Change request log

Team

Team - 7			
Name	**NetID**	**Email**	**Roles**
Md Samiul Aftad Chowdhury	mac151830	mac151830@utdallas.edu	Coding
Kenechukwu Chidozie Nwankwo	kcn062000	kcn062000@utdallas.edu	Documentation

Change Request

FEMR-137:
-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
1	We ran the system	
2	We interacted with the system: after logging in we entered in to the Triage screen.	In order to get familiar with some of the features of the system, and identify the screens or graphical elements we had to change.
3	We checked the route file to find out which controller files is called for /Triage path	In the play framework Route file contains the path to function call mapping
4	From the route path found that it was calling TriageController.indexGet	
5	We inspected the class TriageController. We went to this class using the "jump to implementation " short cut of the IDE editor.	We noticed that the method indexGet calling HTML template of index.scala.html to show the data in the GUI
6	We tried to find out which function is called when the "Birth Date" filed is changed	We decided that if the user enters the date of birth in the Birth Date filed then user is certain the Date of Birth(DOB) is real And if the DOB is taken from integer Age then it is fake.
7	We inspect the **triage.js** file and found that $('#age').change function is called when "Birth date" field is changed.	We now know that from here we can set a data that can identify the DOB is real/fake.
8	Now we inspect the route file to find out which method is called when a POST request is made in the TriageController class.	We need to make sure how the user data is stored in the database and which classes are used for that as we need to store an extra filed in DB to store the Real/Fake flag.
9	We found that POST data converted into a object of class IndexViewModelPost there we found that it holds all the variable used for the POST data	We confirmed this class had to be modified. We need to add a string filed to hold the dobCertainty
10	Next populatePatientItem method use this object of IndexViewModelPost to create PatientItem	We inspect this PatientItem to check if we need to add dobCertainty variable there too

	object	
11	We found that patientService.createPatient is called to store the patient data	patientService.createPatient calles dataModelMapper.createPatient to store data
12	DatamodelMapper uses IPatient interface to store data. We marked createPatient method needs to be change for pass one extra parameter for dobCertainity.	It uses createPatient method to sore that data object.
13	Then we found that we need to make change in “Patient” class which uses interface IPatient.	We identified we need to change createPatientItem method of ItemModelMapper class
14	To store the data we found that we need to make change in the “Patient” table in database(DB)	We need to add a varchar type dobCertainty filed in DB

Time spent (in minutes): 60

Impact Analysis

Step #	Description	Rationale
1	We made a list of methods called by TriageController.indexPost	To track the classes that could be impacted by the change.
2	We inspected the class PatientItem, TriageController, and few other classes that uses createPatientItem method . Such classes was marked as "to change" as well	We realized this class had to be changed because the method createPatient uses the patientService.createPatient again which calls dataModelMapper.createPatient finally this calls createPatientItem of PatientItem class to store data.
3	We made sure that all the classes calling createPatientItem method is updated with the new dobCertain parater.	We found that from 8 locations createPatientItem method is called and updated all 8 locations.

Time spent (in minutes): 30

Prefactoring (optional)

Step #	Description	Rationale
1	We added a variable String dobCertainty in the Patient class and PatientItem class.	We used Intellij IDEs getter, setter method creation tool to set the method for dobCertainty variable.
2	If the value is Certain then we showed this value in the history data of user	We showed this value beside YO/MO of age
3	We committed our changes with git.	Just in case we need to revert our changes.

Time spent (in minutes): 15

Actualization

Step #	Description	Rationale
1	We added String variable to the following classes, PatientItem, Patient, IndexViewModelPost,	We also have created getter/setter methods for that variable in all three classes.
2	In the dataModelMapper.createPatient Method and IPatient createPatient method we have added one extra parameter to pass the dobCertainty.	As we have added extra variable in the Patient class for that reason we had to pass those values to create function.
2	In the triage.js javascript file we have added a check if "Birthdate" filed is changed then it will set a hidden variable in the html form to "Certain"	We have added hidden input tag/variable which indicates a DOB is Certain=Real, Uncertain=Fake, this hidden variable dobCertain can be automatically set to "Certain" if the user enters age by "Birth Date" box.
3	We manually performed functional testing. We also ran the existing test cases.	To make sure everything works.

Time spent (in minutes): 35

Postfactoring (optional)

Step #	Description	Rationale
1	We added String variable to the following classes, PatientItem, Patient, IndexViewModelPost,	We also changed some interfaces to add the dobCertainty variable like IItemModelMapper, DataModelMapper
2	We manually performed functional testing. We also ran the existing test cases.	We tested everything was working as before, after the refactoring.
3	We committed and pushed our changes with git.	Just in case we need to revert our changes.

Time spent (in minutes): 15

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: If user enters Age through Birth date field, it automatically change the value to "Certain" hidden input field Expected output: Checked Patient table if the value of dobCertainty field is set to "Certain". In the history page beside the Age YO it was showing Certain.	This is the regular expected behavior. The test passed.
2	Test case defined: Inputs: If user enters Age through Years, Months field, it automatically selects Uncertain radio button Expected output: Checked Patient table if the value of dobCertainty field is set to "Uncertain". In the history page beside the Age YO it was showing UnCertain	This is the regular expected behavior. The test passed.

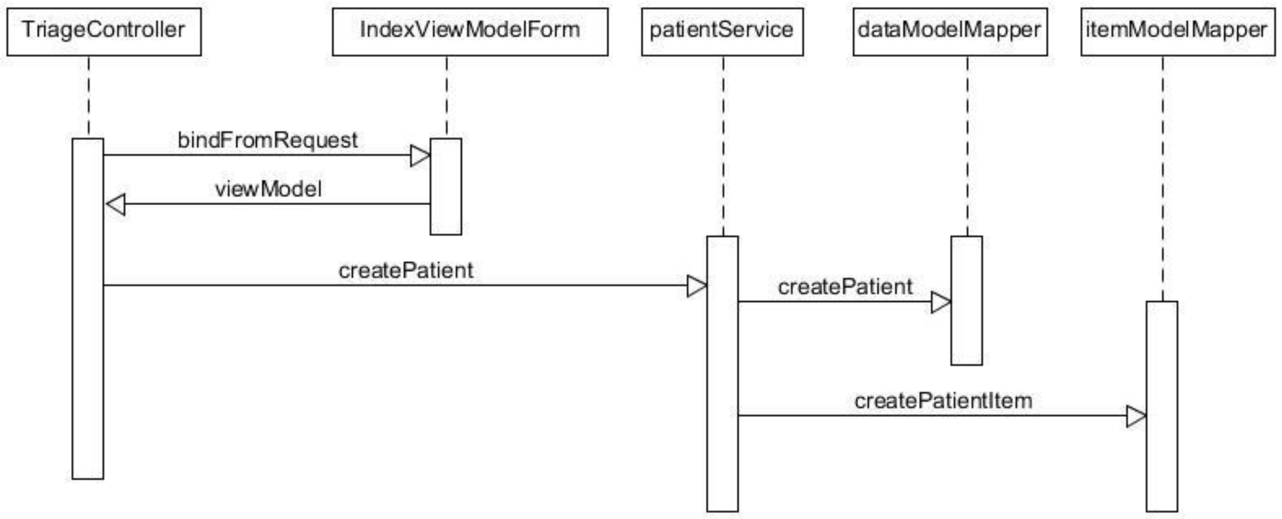
Time spent (in minutes): 25

Timing

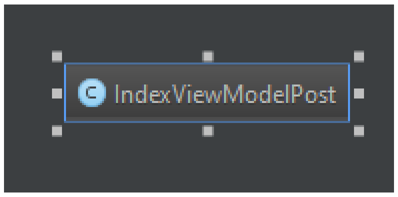
Summarize the time spent on each phase.

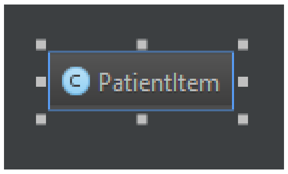
Phase Name	Time (in minutes)
Concept location	60
Impact Analysis	30
Prefactoring	15
Actualization	35
Postfactoring	15
Verification	25
Total	165

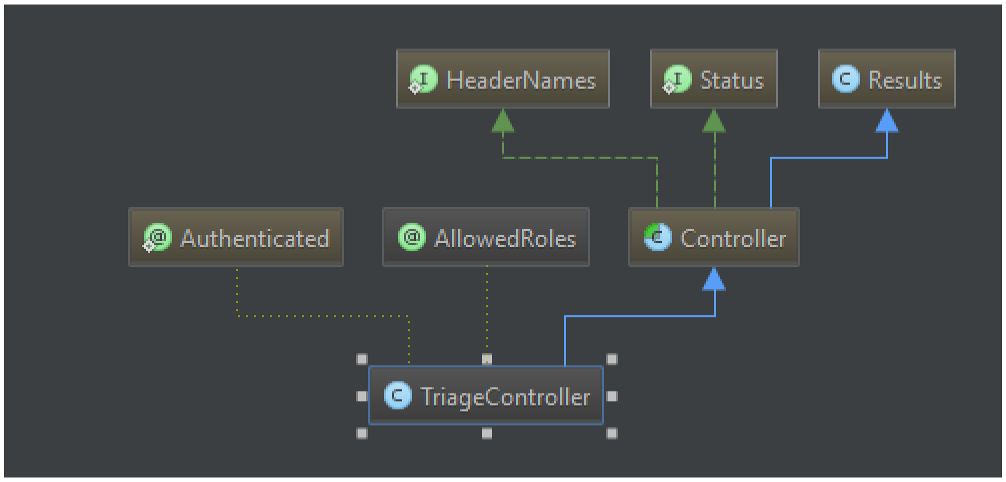
Reverse engineering

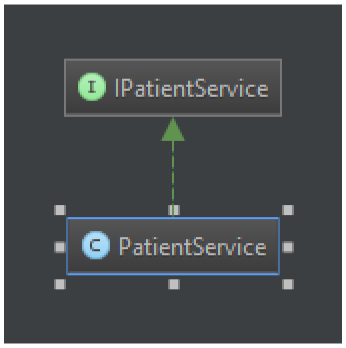


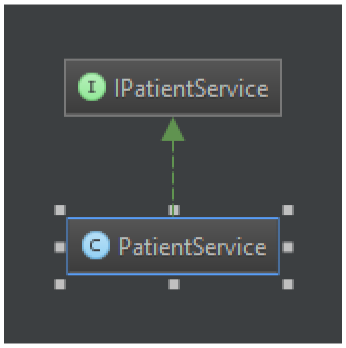
Following classes we have visited for the second part of the change requests.

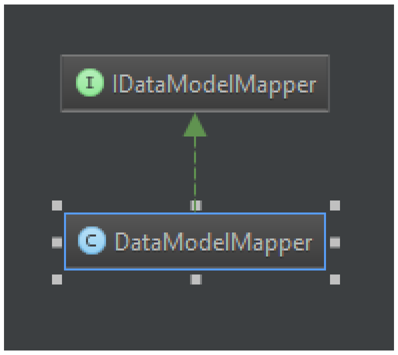


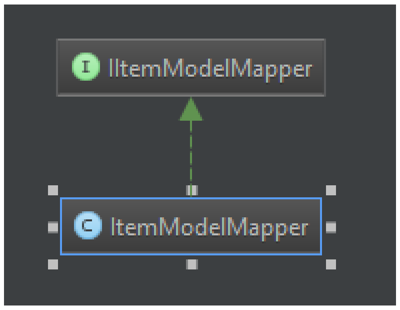


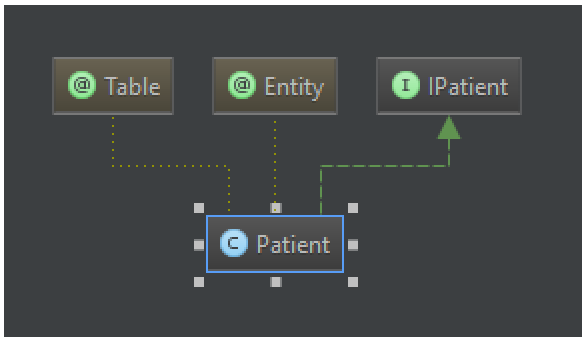












Conclusions

It took us some time to figure out how to Date of birth fake or real can be implemented. We decided to use javascript to detect if the date is entered through the "Birth Date" filed then we are certain that it is real date of birth otherwise it is fake. We identified the Real dob as "Certain" and fake as "Uncertain". Then we decided to store this dob type in data base with the name of dobCertainty. Then we tried to figure out how we can propagate this dobCertainty value from Radio buttons to the classes that stores the data in Database. We visited several classes to find out where new variable needs to added and which methods needs to be changed to store this data.

For this change, concept location was relatively easy because the system is small and its architecture and code are not complicated.

Classes and methods changed:

- modified: app/femr/business/services/system/PatientService.java
  - method: createPatient
- modified: app/femr/business/services/system/SearchService.java
  - method: retrievePatientItemByEncounterId
- modified: app/femr/common/IItemModelMapper.java
  - method: createPatientItem
- modified: app/femr/common/ItemModelMapper.java
  - method: createPatientItem
- modified: app/femr/common/models/PatientItem.java
  - class PatientItem
  - getDobCertainty, setDobCertainty,

- modified: app/femr/data/DataModelMapper.java
    - method: createPatient
- modified: app/femr/data/IDataModelMapper.java
    - method: createPatient
- modified: app/femr/data/models/core/IPatient.java
    - Interface IPatient
    - getDobCertainty, setDobCertainty
- modified: app/femr/data/models/mysql/Patient.java
    - class Patient
    - getDobCertainty, setDobCertainty
- modified: app/femr/ui/controllers/TriageController.java
    - method: populatePatientItem
- modified: app/femr/ui/models/triage/IndexViewModelPost.java
    - getDobCertainty, setDobCertainty
- modified: app/femr/ui/views/triage/index.scala.html
    - added radio buttion
- modified: public/js/triage/triage.js
    - method : $('#age').change(function ())