# Change request log

# Team

Group 3

Susheel Pagadala - xxp155831

Mayuri Banubakode - mxb156530

# Change Request

CR-FEMR-SPMB-137 - Don't require the user editing a user to fill out the "Change User Password" input fields

***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 | We ran the system |  |
| 2 | We created a new patient with ID 12270 | To create a new patient encounter |
| 3 | We created a new medical prescription | To create a new patient/enconter. |
| 4 | Searched for existing patient/encounter | To load existing patient /encounter |
| 5 | We determined the location of the change required is triage index.scala.html | Determine a ui page to be changed |

**Time spent (in minutes):** 28

# Impact Analysis

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | We reviewed the validate method called by EditViewModel | To track the classes that could be impacted by the change. |
| 2 | Determined the required change and impact to overall application | Since this change is localized used for edit user screen from users.js. Impact is minimal and localized to this method caller form users.js |

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

# 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 update the validate method in EditViewModel | Removed the condition when empty passwords are marked as error |

**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: Valid data - Password and verify password are empty  Inputs: Do not edit password and verify password to make those fields to empty  Expected output: User data is saved. | This is the expected behavior.  The test passed. |
| 2 | Test case defined: Invalid data - Password field is empty while verify password is not empty  Inputs: Do not edit password and enter verify password with a text value  Expected output: Error message stating mismatched password | This is the expected behavior.  The test passed. |
| 3 | Test case defined: Invalid data – Password field with verify password is empty  Inputs: Edit password with a text value and Do not enter verify password  Expected output: Error message stating mismatched password | This is the expected behavior.  The test passed. |
| 4 | Test case defined: Valid data - Password and verify password are updated  Inputs: Enter password and verify password with matching text  Expected output: User data is saved. | This is the expected behavior.  The test passed. |

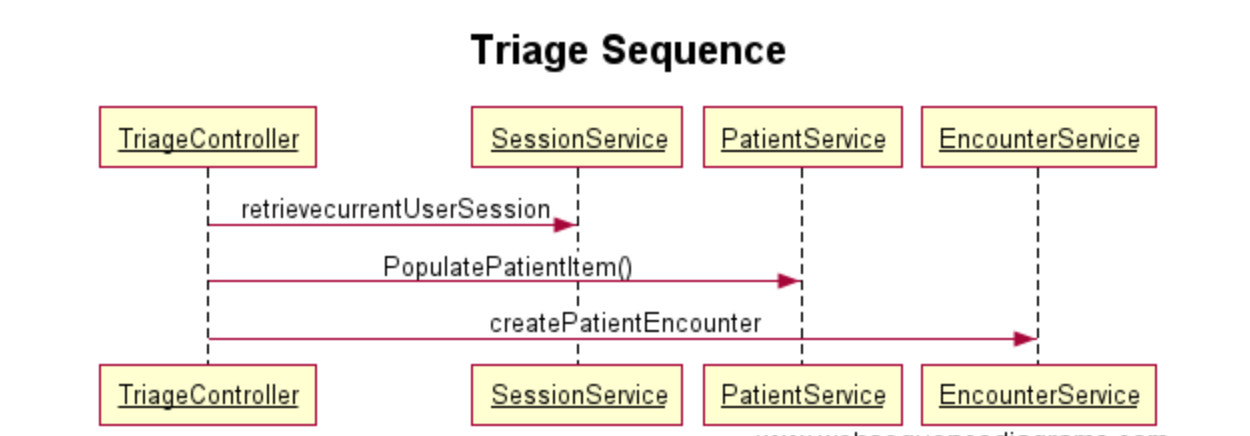
**Time spent (in minutes):** 15

# Timing

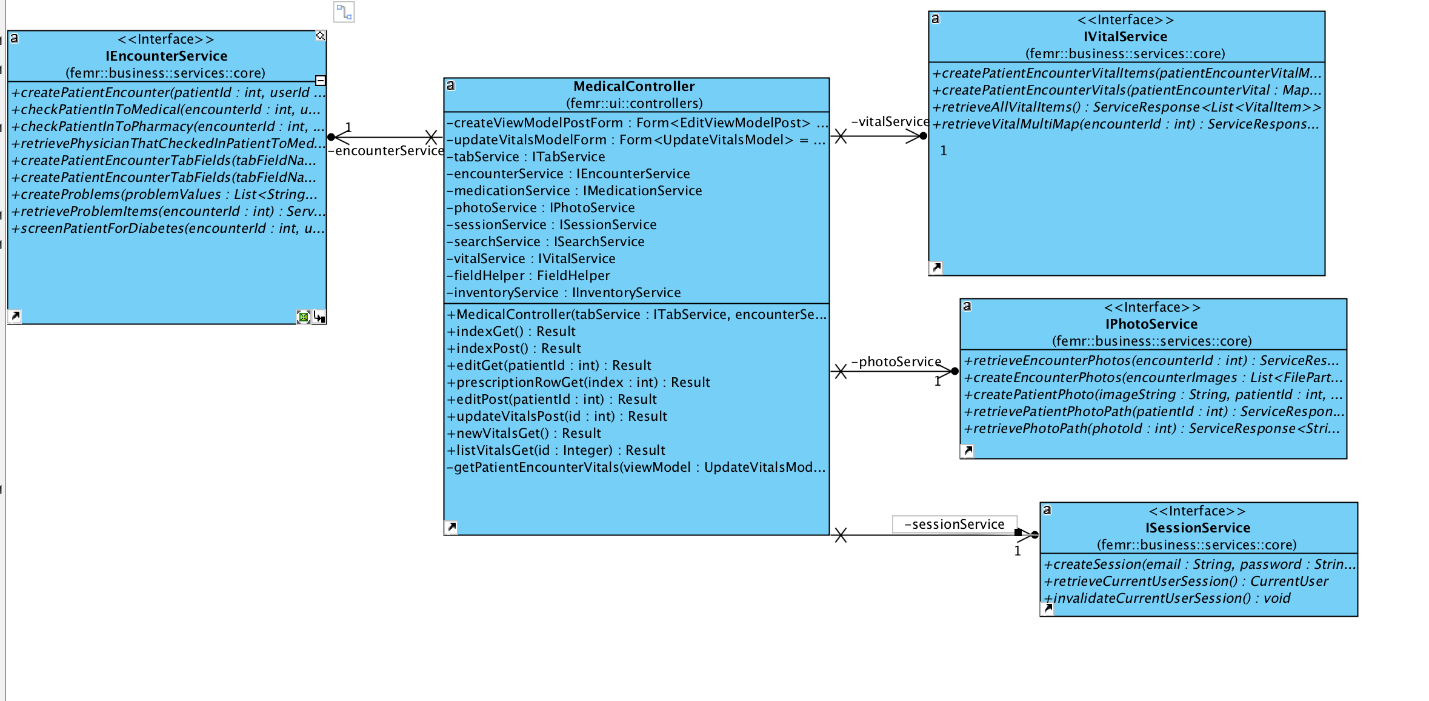
Summarize the time spent on each phase.

|  |  |
| --- | --- |
| Phase Name | Time (in minutes) |
| Concept location | 180 |
| Impact Analysis | 60 |
| Prefactoring | 45 |
| Actualization | 240 |
| Postfactoring | 45 |
| Verification | 120 |
| Total | **690** |

# Reverse engineering



**Class Diagram**



# Conclusions

This was complex change as it required an understanding of current overall frameworks implemented and also how the application flows. Really had to struggle a lot to triage the required changes tracing back each component at a time. Also form a functional usage perspective had to discuss a lot on UI implementation and corresponding updates to classes and eventually storing in DB. There could have been multiple ways to implement changes using check box, radio button, fixed buttons etc. picked a solution that provided faster implementation and simpler usage impact.

Classes and methods changed:

package femr.business.services.system;

public class PatientService implements IPatientService

public ServiceResponse<PatientItem> updateSex(int id, String sex)

public ServiceResponse<PatientItem> createPatient(PatientItem patient)

package femr.business.services.system;

public class SearchService implements ISearchService

public ServiceResponse<PatientItem> retrievePatientItemByPatientId(int patientId)

public ServiceResponse<PatientItem> retrievePatientItemByEncounterId(int encounterId)

package femr.common.models;

public class PatientItem

private Integer fakeBDFlag;

public Integer getFakeBDFlag()

public void setFakeBDFlag(Integer fakeBDFlag)

package femr.common;

public interface IItemModelMapper

Method public PatientItem createPatientItem

package femr.common;

public class ItemModelMapper implements IItemModelMapper

public PatientItem createPatientItem

package femr.data.models.core

public interface IPatient

Integer getFakeBDFlag()

void setFakeBDFlag(Integer fakeBDFlag)

package femr.data

public class DataModelMapper implements IDataModelMapper

public IPatient createPatient

package femr.data

public interface IDataModelMapper

IPatient createPatient

package femr.ui.controllers

public class TriageController extends Controller

private PatientItem populatePatientItem

**package** femr.data.models.mysql;  
**public class** Patient **implements** IPatient

**public** Integer getFakeBDFlag()

**public void** setFakeBDFlag(Integer fakeBDFlag)

ui\views\triage\index.scala.html

Create new radio buttons for Birthday flag fake generated or real entered