CS427 T806 Project Documentation

Weize Sun (weizes2), Yuyan Huang (yhung133), Kaiqi Zheng (kaiqi3)
Yan Feng (yanfeng3), Xiangjue Dong (xd5), Dai Teng (daiteng2), Lu Jin (lujin2)

December 2018

Contents

1 Introduction

Design patterns are representation of the best practices used by expert object-oriented developers. In the term project, a new obstetrics and gynecology (OBGYN) module which includes five use cases UC93, UC94, UC95, UC 96 and UC 97 as listed below were designed and implemented by our team to iTrust.

2 UC93: Obstetrics Patient Initialization

2.1 Overview

In user case 93, the main flow implemented is to establish a main obstetric page for patient eligible for obstetric care where the new pregnancy record can be initialized and prior pregnancy records can be added only by OB/GYN HCPs and prior pregnancy records history can be displayed in descending order for any HCP to view. Some details were considered, including the calculation and display of estimated due date (EDD) based on the patient's last menstrual period (LMP) and the number of weeks pregnant on the obstetrics patient initialization data.

2.2 Frontend structure

There are two related frontend *.jsp* files in UC93– *obstetricsHome.jsp* and *addObstetricsRecord.jsp*. The logic of frontend structure is listed as follows:

- The frontend get the pid of the patient input in obstetrics record home page;
- Initialize a *ViewPatientAction* as well as a *PatientBean* to get patient's detailed information;
- Check if this patient is eligible for obstetrics care. (Report an error if the patient is not eligible, e.g. the sex of the patient is male);
- Check if user is in the identity of OB/GYN HCP, which may enable the user to perform more functions, such as initializing and adding obstetric record, not limited in viewing records in this page;

- A new pregnancy record can be initialized by clicking the button of *Initialize Obstetrics Record* on *Obstetrics Record Home* page or through *Initialize Obstetrics Record* interface;
- Guide the user to fill in the item of *Today's Date* and *Last menstrual period*;
- *Weeks-Days pregnant* and *Estimated delivery date* will be generated automatically;
- Check whether the form is filled or not;
- Initialize a error list for error reporting;
- Patient prior pregnancy record can be added by clicking *Add Prior Pregnancy* through *Initialize Obstetrics Record* page ;
- Past pregnancy records includes items, such as year of conception, weeks-days pregnant, hours in labor, weight gained, delivery type, multipregnancy and baby count.
- Initialize a *ObstetricsRecordBean* to record obstetrics records;
- Fill in the fields of the *ObstetricsRecordBean* according to user's input;
- Report error if the error list if not empty;
- Push the *ObstetricsRecordBean* to the backend;

2.3 Backend Structure

Bean file *ObstetricsRecordBean* was created for obstetrics records. Loader file *ObstetricsRecordLoader* was created for get and set data for associated bean files. Corresponding DAO file *ObstetricsRecordDAO* was created for inserting and fetching data from database. Finally, validator file *ObstetricsRecordValidator* was created to validate the input for both tables. Action file *AddObstetricsAction* for adding records was created for action functions. Transaction logs of "Create Initial Obstetric Record" and "View Initial Obstetric Record" were created during the construction of action files.

2.4 Database

Figure 1: Obstetrics Record Database

#	Field	Schema	Table	Type	Character Set	Display Size	Precision	Scale
1	id	itrust	obstetrics	BIGINT UNSIGNED	binary	20	1	0
2	MID	itrust	obstetrics	BIGINT UNSIGNED	binary	20	1	0
3	initDate	itrust	obstetrics	DATE	binary	10	10	0
4	LMP	itrust	obstetrics	DATE	binary	10	10	0
5	EDD	itrust	obstetrics	DATE	binary	10	10	0
6	weekPregnant	itrust	obstetrics	VARCHAR	utf8	4	4	0
7	concepYear	itrust	obstetrics	INT	binary	4	4	0
8	hrsLabor	itrust	obstetrics	FLOAT	binary	12	-29	31
9	weightGain	itrust	obstetrics	FLOAT	binary	12	-27	31
10	weight	itrust	obstetrics	FLOAT	binary	12	-26	31
11	bloodPressureL	itrust	obstetrics	INT	binary	11	2	0
12	bloodPressureH	itrust	obstetrics	INT	binary	11	3	0
13	FHR	itrust	obstetrics	INT	binary	11	3	0
14	deliveryType	itrust	obstetrics	ENUM	utf8	31	30	0
15	pregnancyStatus	itrust	obstetrics	ENUM	utf8	11	11	0
16	multiPregnancy	itrust	obstetrics	TINYINT	binary	1	1	0
17	babyCount	itrust	obstetrics	INT	binary	2	1	0
18	lyingPlacenta	itrust	obstetrics	TINYINT	binary	1	1	0

2.5 Snapshots

Figure 2: Obstetrics Records Homepage

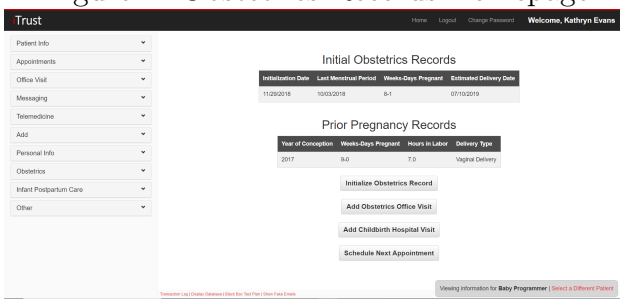


Figure 3: Initialize Pregnancy Records Page

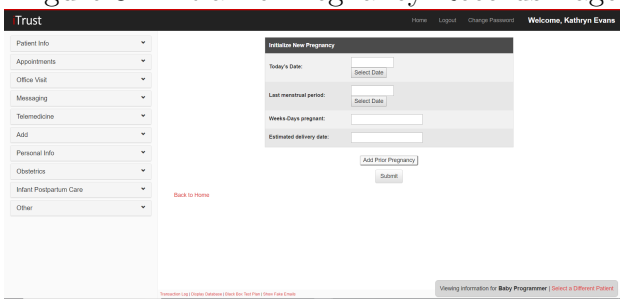
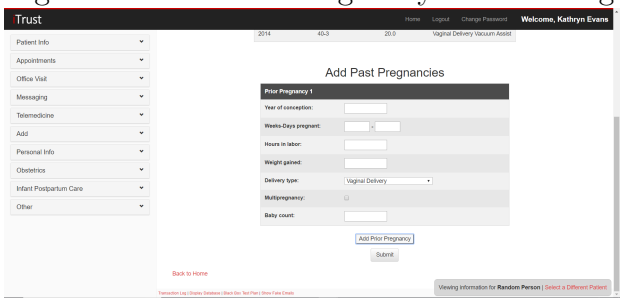


Figure 4: Add Prior Pregnancy Records Page



3 UC94: Obstetrics Office Visit

3.1 Overview

In user case 94, the team implemented a series of functions including: let a HCP to document or edit an obstetrics office visit for a current obstetrics patient; display required alter information if certain conditions are met for the patient; let a HCP to perform an ultrasound during the obstetrics office visit with record creation; let the HCP to add the next appointment with certain restrictions.

3.2 Frontend structure

There are three related frontend *.jsp* files in UC94– *addObstetricsOfficeVisit.jsp*, *editObstetricsOfficeVisit.jsp* as well as *scheduleAppt.jsp*. The logic of *addObstetricsOfficeVisit.jsp* is as follows:

- The frontend get the pid of the user;
- Initialize a *ViewPatientAction* as well as a *PatientBean* to get patient's information;
- Check if this patient is eligible for obstetrics care. (Report an error if the patient is not eligible, e.g. the sex of the patient is male);
- Check if this user is in the identity of OB/GYN, which may enable the user for more functionalities in this page;
- Initialize a *AddObstetricsAction*, prepare to fill in the fields;
- Guide the user to fill in the form;
- Check whether the form is filled or not;
- Initialize a error list for error reporting;
- Check if there is any ultra sound record added by the user, if any, initialize and fill in an *UltrasoundBean* and push to backend (this may involve an image-uploading);
- Initialize a *ObstetricsRecordBean* to record current office visit;
- Fill in the fields of the *ObstetricsRecordBean* according to user's input;
- Report error if the error list if not empty;
- Push the *ObstetricsRecordBean* to the backend;

Users are eligible to enter *editObstetricsOfficeVisit.jsp* through the link from *obstetricsHome.jsp* if they are in the identity of OB/GYN. *editObstetricsOfficeVisit.jsp* will go through same procedures of *addObstetricsOfficeVisit.jsp* except those steps related to identity check. *scheduleAppt.jsp* is the page where patients can schedule appointment. This frontend page will follow the following logics:

- Initialize three *Action* objects for further usage– *AddApptAction*, *ViewMyApptsAction* and *EditApptTypeAction*;
- Get the pid of the user;
- Guide the user to select the appointment time and add comments;
- Check if the time user selected is eligible for appointment (Patients cannot make appointments on weekends and non-working time slots unless emergency);
- Push the appointment and comments to the backend;
- Direct the patient the the page for viewing their appointment records.

3.3 Backend Structure

Bean files were created for obstetrics office visit and ultrasound record respectively. Loader files were created for get and set data for associated bean files. Corresponding DAO files were created for inserting and fetching data from database. Finally, validator files were created to validate the input for both tables. Action files for adding records were created for both functions. Additionally, edit action for obstetrics office visit was created. Transaction logs were created during the construction of action files.

3.4 Database

Figure 5: Obstetrics Office Visit Database

#	Field	Schema	Table	Type	Character Set	Display Size	Precision	Scale
1	id	itrust	obstetrics	BIGINT UNSIGNED	binary	20	1	0
2	MID	itrust	obstetrics	BIGINT UNSIGNED	binary	20	1	0
3	initDate	itrust	obstetrics	DATE	binary	10	10	0
4	LMP	itrust	obstetrics	DATE	binary	10	10	0
5	EDD	itrust	obstetrics	DATE	binary	10	10	0
6	weekPregnant	itrust	obstetrics	VARCHAR	utf8	4	4	0
7	concepYear	itrust	obstetrics	INT	binary	4	4	0
8	hrsLabor	itrust	obstetrics	FLOAT	binary	12	-29	31
9	weightGain	itrust	obstetrics	FLOAT	binary	12	-27	31
10	weight	itrust	obstetrics	FLOAT	binary	12	-26	31
11	bloodPressureL	itrust	obstetrics	INT	binary	11	2	0
12	bloodPressureH	itrust	obstetrics	INT	binary	11	3	0
13	FHR	itrust	obstetrics	INT	binary	11	3	0
14	deliveryType	itrust	obstetrics	ENUM	utf8	31	30	0
15	pregnancyStatus	itrust	obstetrics	ENUM	utf8	11	11	0
16	multiPregnancy	itrust	obstetrics	TINYINT	binary	1	1	0
17	babyCount	itrust	obstetrics	INT	binary	2	1	0
18	lyingPlacenta	itrust	obstetrics	TINYINT	binary	1	1	0

Figure 6: Ultrasound Database

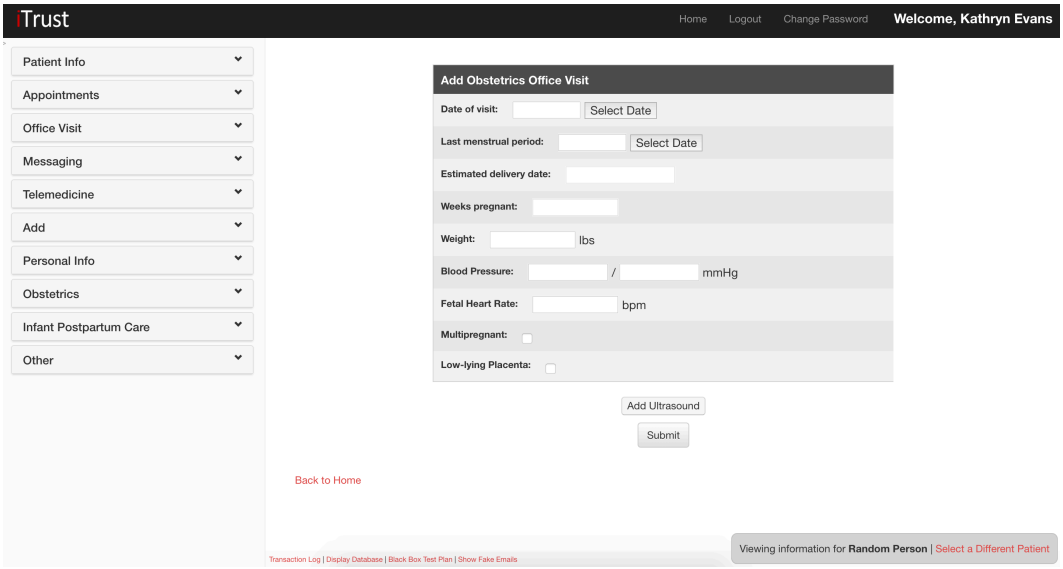
Column	Type	Default Value
id	bigint(20) unsigned	
MID	bigint(20) unsigned	0
fetusID	bigint(20)	
visitDate	date	
crownRumpLength	float	0
biparietalDiameter	float	0
headCircumference	float	0
femurLength	float	0
occipitofrontalDiameter	float	0
abdominalCircumference	float	0
humerusLength	float	0
estimatedFetalWeight	float	0

Figure 7: Ultrasound Image Database

Column	Type	Default Value	Nullable
MID	bigint(20) unsigned	0	NO
fetusID	bigint(20)	0	NO
ultrasoundImage	longblob		YES

3.5 Snapshots

Figure 8: Obstetrics Office Visit Page



4 UC95: Labor and Delivery Report

4.1 Overview

UC95 allows any HCP to generate report containing information about pregnancy, including delivery report, blood type, obstetrics office visit information, pregnancy complication warning flags, pre-existing conditions, and durg allergies.

4.2 Frontend structure

The frontend of UC95 is simple because it is a report page which involves no interaction. There is only one *.jsp* page for UC95– *report.jsp*. The *report.jsp* follows the logic as follows:

- The frontend get the pid of the user;
- Check if this patient is eligible for obstetrics care. (Report an error if the patient is not eligible, e.g. the sex of the patient is male);
- Initialize a *ViewPregnancyReportAction*

- Print out prior pregnant records with the help of *ViewPregnancyReportAction*;
- Get list of *ObstetricsRecordBean* to print out office visits generated;
- Get pregnancy compilation warning flags according to the database;
- Get relevant pre-existing conditions and print out;

4.3 Backend Structure

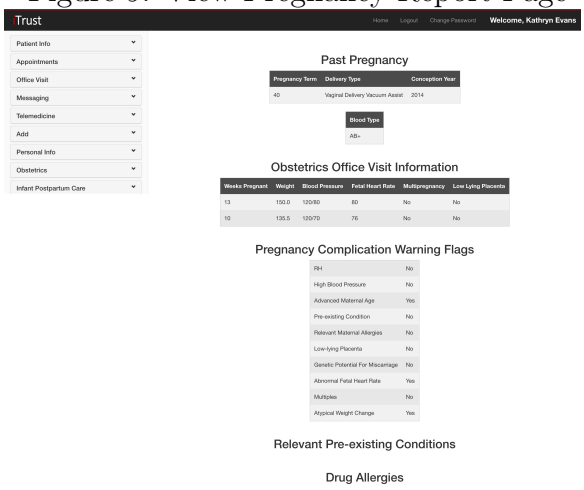
Only action files was generated for this user case. All other backend files were created before UC95 implementation. Existing bean, DAO, loader, validator, and action files were used to complete UC95.

4.4 Database

There was no new data base created for UC95.

4.5 Snapshots

Figure 9: View Pregnancy Report Page



5 UC96: Childbirth Hospital Visit

5.1 Overview

In user case 96, the team implemented a series of functions including: create childbirth hospital visit through pre-scheduled appointment or emergency room; display history of the patient's obstetrics care; let HCP specify patient's preferred childbirth method; display drug usage conditions; and record baby's condition as new patient at the time of delivery.

5.2 Frontend structure

There are three related frontend *.jsp* files in UC94– *addChildbirthHospitalVisit.jsp* as well as *editChildbirthHospitalVisit.jsp*. The logic of *addChildbirthHospitalVisit.jsp* is as follows:

- The frontend get the pid of the user;
- Initialize a *ViewPatientAction* as well as a *PatientBean* to get patient's information;
- Check if this patient is eligible for obstetrics care. (Report an error if the patient is not eligible, e.g. the sex of the patient is male);
- Check if this patient is in the identity of OB/GYN, which may enable the user for more functionalities in this page;
- Initialize a *AddChildbirthAction*, prepare to fill in the fields;
- Guide the user to fill in the form;
- Check whether the form is filled or not;
- Initialize a error list for error reporting;
- Check if there is any child born, if any, initialize and fill in an *PatientBean* and push to backend (We utilize a built-in class *AddPatientAction* to add this new born baby as a patient into the database);
- Initialize a *ChildbirthRecordBean* to record current office visit;
- Fill in the fields of the *ChildbirthRecordBean* according to user's input;
- Report error if the error list if not empty;
- Push the *ChildbirthRecordBean* to the backend;

Users are eligible to enter *editChildbirthHospitalVisit.jsp* through the link from *addChildbirthHospitalVisit.jsp* if they are in the identity of OB/GYN. *editChildbirthHospitalVisit.jsp* will go through same procedures of *addChildbirthHospitalVisit.jsp* except those steps related to identity check.

5.3 Backend Structure

Bean file was created for childbirth hospital visit. Loader files was created for get and set data for associated bean file. Corresponding DAO file was created for inserting and fetching data from database. Finally, validator file was created to validate the input. Action files for adding and editing record were created. Transaction logs were created during the construction of action files.

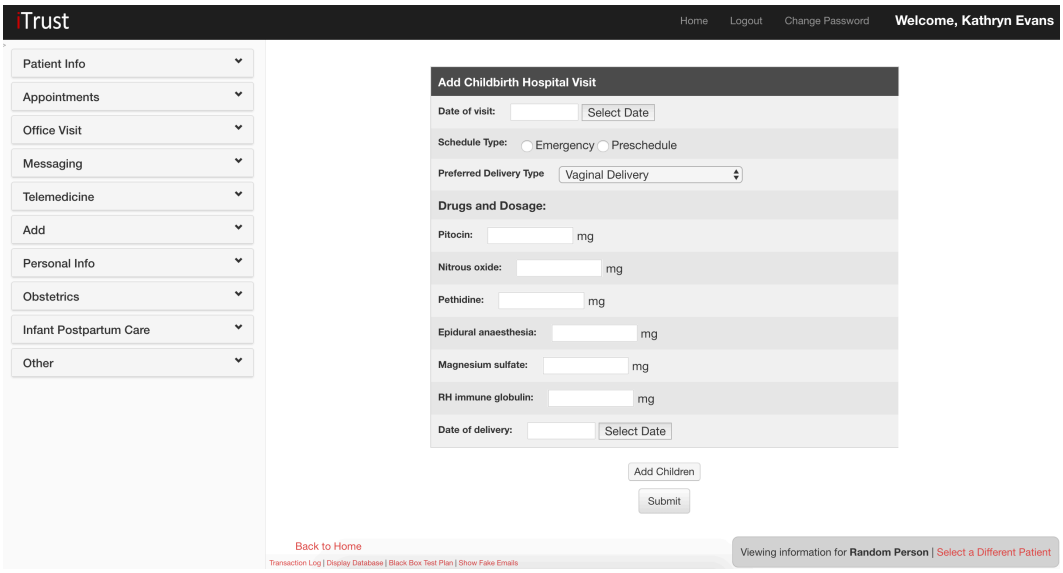
5.4 Database

Figure 10: Childbirth Hospital Visit Database

Column	Type	Default Value
id	bigint(20) unsigned	
MID	bigint(20) unsigned	0
visitDate	date	
visitType	enum('Pre-scheduled','ER')	Pre-scheduled
deliveryType	enum('Vaginal Delivery','Vaginal Delivery Vacuum Assist','Vaginal Delivery Forceps Assist','Caesarean Section','Miscarriage')	Vaginal Delivery
pitocin	tinyint(1)	0
pitocinDosage	float	0
nitrousOxide	tinyint(1)	0
nitrousOxideDosage	float	0
pethidine	tinyint(1)	0
pethidineDosage	float	0
epiduralAnaesthesia	tinyint(1)	0
epiduralAnaesthesiaDosage	float	0
magnesiumSulfate	tinyint(1)	0
magnesiumSulfateDosage	float	0
rhImmuneGlobulin	tinyint(1)	0
rhImmuneGlobulinDosage	float	0
babyID	int(2)	1
deliveryTime	date	
babySex	enum('Male','Female')	

5.5 Snapshots

Figure 11: Childbirth Hospital Visit Page



6 UC97: Infant Postpartum Care

6.1 Overview

In UC 97, the team implemented a series of functions including: let a HCP to document or edit an infant postpartum care visit record for an infant patient; record every important health indexes for an infant; present a list of possible vaccines information for any infant that meet the requirements to take them. Our new functionality can keep track of the health of the new infants and remind of the doctors and its parents to accept vaccine injection in time.

6.2 Frontend structure

There are three related frontend *.jsp* files in UC97– *infantVisitHome.jsp*, *addInfantVisitRecord.jsp* as well as *editInfantVisitRecord.jsp*.
The logic of *infantVisitHome.jsp* and *addInfantVisitRecord.jsp* is as follows:

- The frontend get the pid of the user;
- Initialize a *ViewPostpartumCareAction* as well as a *InfantVisitRecordBean* to get infant's information;
- Check if this patient is eligible for postpartum care. (Report an error if the patient is not an infant, which means that all infants should be born in 365 days);
- Check if this user is in the identity of OB/GYN, which may enable the user for more functionalities (be able to add/edit records) in this page;
- Initialize a *addInfantVisitRecord*, prepare to fill in the fields;
- Guide the user to fill in the form;
- Check whether the form is filled or not;
- Initialize a error list for error reporting;
- Initialize a *InfantVisitRecordBean* to record current office visit;
- Fill in the fields of the *InfantVisitRecordBean* according to user's input;
- Report error if the error list if not empty;
- Push the *InfantVisitRecordBean* to the backend;

Users are eligible to enter *editInfantVisitRecord.jsp* through the link of *infantVisitHome.jsp* if they are in the identity of OB/GYN. *editInfantVisitRecord.jsp* will go through same procedures of *addInfantVisitRecord.jsp* except those steps related to identity check.
There is also a table shows the recommendation dates for five different kinds of vaccines in the page of *infantVisitHome.jsp*. The dates in this table was collected and calculated from

the date of birth of the infant. As there is no column storing this data in *infanVisit* table, we need to call *PatientBean* for the chosen patient, and run corresponding function to fetch date of birth information from *patients* table.

6.3 Backend Structure

Bean files were created for infants postpartum office visit. Loader files were created for get and set data for associated bean files. Corresponding DAO files were created for inserting and fetching data from database. Finally, validator files were created to validate the input for both tables. Action files for adding records were created for both functions. Additionally, edit action for infants postpartum office visit was created. Transaction logs were created during the construction of action files.

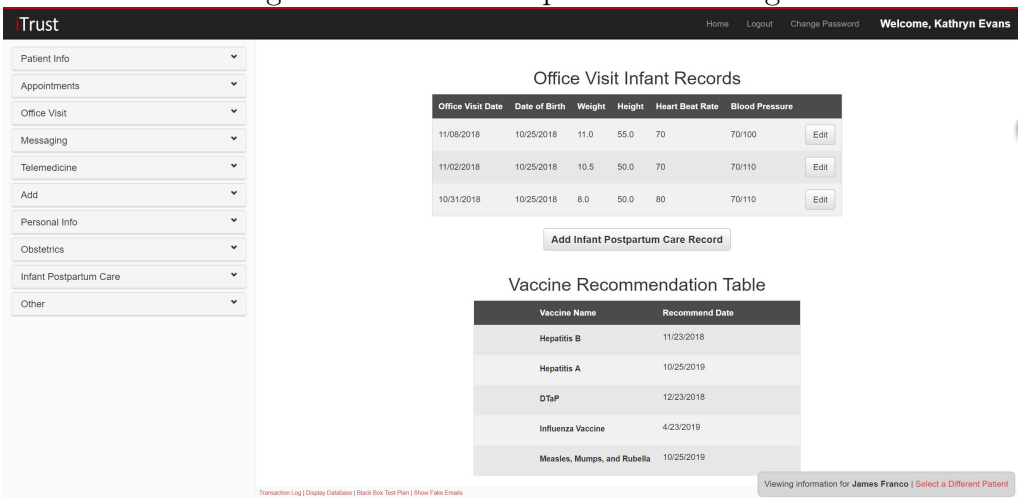
6.4 Database

Figure 12: UC97 Database Schema

Column	Type	Default Value	Nullable
id	bigint(20) unsigned		NO
MID	bigint(20) unsigned	0	NO
visitDate	date		YES
weight	float	0	YES
height	float	0	YES
heartbeatRate	int(11)	0	YES
bloodPressureL	int(11)	0	YES
bloodPressureH	int(11)	0	YES

6.5 Snapshots

Figure 13: Infant Postpartum Visit Page



7 Test cases

Unit test is conducted for each Bean, DAO, Action, Loader, Validator files we have created. For example, for each Validator file created, corresponding test files have been created, including *ChildbithRecordValidatorTest.java*, *ObstetricsRecordValidatorTest.java*, and *UltrasoundRecordValidatorTest.java*.

Let me use *UltrasoundRecordValidatorTest.java* as an example to further illustrate this. *UltrasoundRecordValidator* basically validate that a *UltrasoundRecord* is in correct format. In the test, we first initialize a *UltrasoundRecordBean* with all its attributes in invalid format. For instance, set Visit Date to be null, and set Estimated Fatal Weight to be -5. Then the Validator would print out some error message showing that "Visit Date is a required field" and "Estimated fatal weight should be positive". Then we assert the error message generated is the same as we expected by applying assertEquals() function. Then we could make sure the Validator works in the way we expected.

Finally, we generated 29 test files, every file has a more than 80 percent line coverage. This demonstrate that our Bean, DAO, Action, Loader, Validator file can work as the way we expected.