T812 Documentation

By

Teerapat Pokaprakarn (pokapra2)
Alan Yang (yang161)
Abhishek Chatterjee (chattrj3)
Vincent Del Toral (deltora1)
Wee Lim (weelim2)
Walker Henderson (whender2)
Terrence Katzenbaer (tkatzen)

Contents

Project Overview

The goal of this project is to improve the functionality of the iTrust system. Several use cases have been implemented to allow different users (HCPs, Administrators, Patients, etc) to get more out of the iTrust system.

Aside from full functionality, each use case implementation was well designed and implemented to achieve three goals: seamless user experience and good architectural design (including durability and extensibility).

As a user explores the new iTrust system, they will notice the new use cases make use of already existing buttons, menus, pages, and actions in the iTrust system (where logical). This requires the user to learn almost nothing new in terms of interaction before they can start using the new system as before.

Good architecture is an important point in and of itself. There are also other reasons why we focused on good design. The first was initial construction and debugging. If components of the system are separated, there is much less slow down and confusion as the growing system becomes more complex and develops inevitable bugs (as all software projects do).

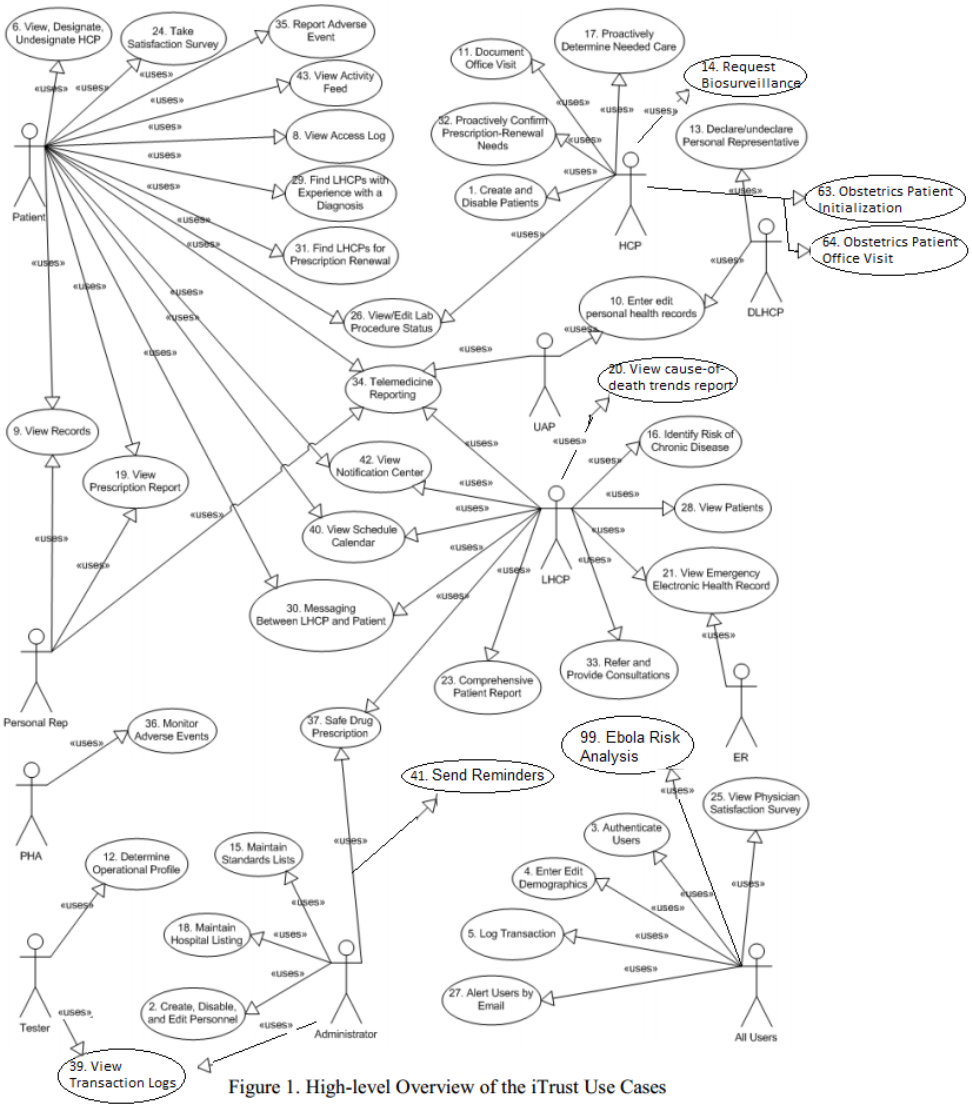

Figure 1. High-level Overview of the iTrust Use Cases

Good design also allows more precise testing. As you will observe at the end of this documentation, extensive tests have been added for all use cases. With poor design, testing would have been much more difficult. (If you make a direct SQL query call in a JavaServer Page, how would you test that?).

Finally, good design allows the system to continue to grow and adapt to client demands. Current use cases can be modified and new uses cases can be added without interference or unnecessary grief caused by our implementations.

UC 39 View Transaction Logs

Brief Description

For this use case, a function was implement for Admins and Testers. Two main functionalities are added to the existing iTrust system.

1. Admins or Testers are able to filter through transactions logs using the parameters: Logged in as, Secondary User, Start Date, End Date and Transaction Type.
2. Users are able to see a summary of the transactions using the above-mentioned parameters. The summary will be shown as bar charts. Four bar charts will be shown
    a. Number of Transaction vs Logged-In Role
    b. Number of Transaction vs Secondary Role
    c. Number of Transaction vs Months
    d. Number of Transaction vs Transaction Type

Both front-end and back-end error checking for the date input have also been implemented.

Architecture & Design

Front-end

/iTrust/WebRoot/auth/admin/menu.jsp
/iTrust/WebRoot/auth/admin/viewTransactionLog.jsp
/iTrust/WebRoot/auth/admin/viewTransactionLogChart.jsp
/iTrust/WebRoot/auth/admin/viewTransactionLogTable.jsp
/iTrust/WebRoot/auth/tester/menu.jsp
/iTrust/WebRoot/auth/tester/viewTransactionLog.jsp
/iTrust/WebRoot/auth/tester/viewTransactionLogChart.jsp
/iTrust/WebRoot/auth/tester/viewTransactionLogTable.jsp

The *menu.jsp* files were used in order to add a View Transaction Log option in the sidebar menu when a tester or admin logs in.

viewTransactionLog.jsp is responsible for getting the input from users. Various front-end error checks are implemented here. Date input fields must be filled. This was implemented using the html “required” tag. Logical error checking for date is also implemented here. The code checks for invalid month inputs, invalid date format, and also whether the begin date is before the end date. If all inputs are valid, the code will navigate to viewTransactionLogChart.jsp or viewTransactionLogTable.jsp depending on whether the user clicked on the view or summarize submit button. All field data are passed using the GET Method.

viewTransactionLogTable.jsp is responsible for displaying data in table format. It will pass in necessary parameters to viewTransactionBean which will be used in getTransactionView method found inside the ViewTransactionAction class. The method returns a list of viewTransactionBean which will be displayed in table format.

viewTransactionChart.jsp is responsible for showing a summary of the data in bar chart form. GoogleChart Api is used to display graphs. The procedure used in viewTransactionLogTable for retrieving the data is also used here. However, in order to arrange the data to be display on the charts, a function inside action class is called to count and group transaction log in the specified format.

The JSP-files used for admins are also used for testers.

Bean Class

/iTrust/src/edu/ncsu/csc/itrust/beans/ViewTransactionBean

This bean is responsible for storing data from the input form and also storing data when data is being retrieved from the database.

DAO Class

/iTrust/src/edu/ncsu/csc/itrust/beans/mysql/ViewTransactionDAO

This DAO class is responsible for executing the SQL and putting the returned data in a viewTransactionBean List. Two SQL statements are executed. The first one is responsible for getting all transaction logs that are relevant to the logged-in users. The second one is for getting secondary users. After that, a for-loop is used to match the transaction IDs from the primary and secondary users to get the specific transaction as requested by the user.

Action Class

/iTrust/src/edu/ncsu/csc/itrust/action/ViewTransactionAction

Five methods are implemented in this class. The first one is responsible for calling the above DAO class and return the list of Beans back to the front-end. The rest is responsible for counting and grouping the data to show on the graph for the summary function. A hashmap is returned for these functions. For example, in getDistinctLoggedUser method, a Hashmap with the Loggedin role as a key and number of transactions as a value is returned. A simple for-loop through the List of ViewTransactionBean is implemented to count the repeating loggedin Role.

UC 14 Request Biosurveillance

Brief Description

The purpose of this use case is twofold. The first is to give the Health Care Providers (HCP) a way to easily determine if there is a malaria or influenza epidemic within two weeks of a certain date and in a specific region.

1. For malaria, there is an epidemic if the percentile of the average of the number of cases obtained in the same week of all years other than the one under consideration is greater or equal to the percentage threshold in which the HCP enters.
2. For influenza, there is an epidemic if the defined threshold is exceeded when the number of incidents in a week exceeds the threshold given by a specific formula.

For more information on the threshold algorithms, please refer to the link: https://wiki.cites.illinois.edu/wiki/display/cs427fa14/UC+Request+Biosurveillance.

The second purpose of this use case is to show the trend and frequency of malaria and influenza within eight weeks of a given date in a region, state, and nationwide on a bar chart.
Front-end and back-end error checking have also been implemented for the inputs. Also each action that the HCP user specifically prompts (analyzing if there is an epidemic or viewing the trend of a certain disease) is logged.

Architecture & Design

Front-end

/iTrust/WebRoot/auth/hcp/menu.jsp
/iTrust/WebRoot/auth/hcp/requestBiosurveillance.jsp
/iTrust/WebRoot/auth/hcp/requestSurveillanceTrendResult.jsp

menu.jsp was used in order to add a Request Biosurveillance option in the sidebar menu when an hcp logs in.

When Request Biosurveillance is selected in the side menu, the HCP is redirected to a new page which is setup by the *requestBiosurveillance.jsp* file. This page contains 4 inputs for the Epidemic Analysis Request. one being radio buttons to choose between malaria and influenza, and three text fields for the HCP to enter zip code, date, and threshold. These inputs are then followed by a submit button. On the bottom of the page is another form for Epidemic Trend Request which contains 3 input text fields for diagnosis code, zip code, and date. There is a front end check implemented on this file as well that makes sure all inputs must be filled in for their respective form before the form can be submitted. The code also checks for logic errors in the inputs such as if the date input has a month value that is over the number 12 or a day value larger than the number of days in the month.

When the Request Trend Request form is submitted, the user is redirected to the *requestSurveillanceTrendResult.jsp* page. This page is use to show a bar chart of the trend and frequency of malaria or influenza eight weeks before the input date. Each week shown in the chart contains 3 bars, one for region as determined by the first 3 digits of the zip code, one for state as determined by the first 2 digits of the zip code, and finally one for the entire nation. The GoogleChart Api is used to display these graphs. The information necessary to create this chart is retrieved by the *RequestBioSurveillanceTrendAction.java* file.

Bean Class

/iTrust/src/edu/ncsu/csc/itrust/beans/BioSurveillanceBean.java

This bean is responsible for storing data from the input form and also storing data when data is being retrieved from the database. When the Analysis and Trend forms are submitted, the inputs are passed to the *BioSurveillanceBean.java*.

Action Class

/iTrust/src/edu/ncsu/csc/itrust/action/RequestBioSurveillanceAnalysisAction.java
/iTrust/src/edu/ncsu/csc/itrust/action/RequestBioSurveillanceTrendAction.java

RequestBioSurveillanceAnalysisAction.java is used to calculate the percentile threshold and determine if there is an epidemic for both malaria and influenza with the Bean passed in from the form. The malaria and influenza functions return a boolean telling the HCP if there is an epidemic based on the input information. For specific algorithm information please look at the “Brief Description” section.

RequestBioSurveillanceTrendAction.java is used to retrieve the information necessary to create the trend chart in *requestSurveillanceTrendResult.jsp*. The input bean information is passed from the requestBiosurveillance.jsp to this action class while the the database information is passed in with by the *ovDAO*, *OfficeVisitBean*, *patientDAO,* and *patientBean*. First all of the office visits with the given Diagnosis code were stored in a list of *OfficeVisitBean*s. Then from that list of beans, we compare the

patient ids with the patient beans and determine if a specific patient with malaria or influenza has a zipcode within a specific region or state. A list of 3 integers is returned based on the counters of the number of patients with the disease in the state, region, or nationwide in a given week.

Validation Class

/iTrust/src/edu/ncsu/csc/itrust/validate/ValidationFormat.java
/iTrust/src/edu/ncsu/csc/itrust/validate/RequestBioSurveillanceValidator.java

There is also backend error checking implemented for both trend and analysis with *RequestBioSurveillanceValidator.java* which calls on the regex checking from the file *ValidationFormat.java*. We added an extra regex for the Threshold input so that if the HCP chose malaria from their radio buttons, they must input a valid number Threshold in that field. Other regex formats for date, zip code, and diagnosis code were provided in *ValidationFormat.java* and used accordingly.

UC 63 Obstetrics Patient Initialization

Brief Description

This use case allows an obstetrics/gynecology HCP (ob/gyn) to create new obstetrics initialization record (OIR) as well as create new prior pregnancies. In addition, all HCPs (ob/gyn or otherwise) will be able to access prior obstetrics initialization records in view only mode. This use case is tied to UC64, in that a patient must have an active obstetrics initialization record in order for an obstetrics office visit to be recorded.

For more information on this use case, please visit:
http://agile.csc.ncsu.edu/iTrust/wiki/doku.php?id=requirements:uc63

Architecture & Design

Front-end

/iTrust/WebRoot/auth/hcp/OIRList.jsp
/iTrust/WebRoot/auth/hcp/OIRPage.jsp
/iTrust/WebRoot/auth/hcp/priorPregnancy.jsp

The flow on the client side is as follows: HCP logs in -> Obstetrics sidebar link -> OIRList -> OIRPage -> priorPregnancy.

OIRList lists all obstetrics initialization records and provides a link to create a new record (if the HCP is an ob/gyn). OIRPage has a view mode and a create mode. The view mode of the OIRPage shows the details of a previously created obstetrics initialization record, along with all prior pregnancies for that patient. The create mode of the OIRPage allows an ob/gyn to create a new obstetrics initialization record, as well as a link to create a new prior pregnancy. The priorPregnancy page allows an ob/gyn to create a new prior pregnancy for that patient. Please note that prior pregnancies are only tied to the patient, and are not tied to any specific obstetrics initialization record.

Action Class

/iTrust/src/edu/ncsu/csc/itrust/action/OIRAction.java
/iTrust/src/edu/ncsu/csc/itrust/action/RequestBioSurveillanceTrendAction.java

OIRAction has three current functions, allowing the user to get an OIRBean list of all OIRs for a specific patient, to get a specific OIR, and to create a new OIR. For creation, the creation date is set as the current system date at the time of creation. This class is coupled with the OIRBean class.
PregnancyAction has two current and active functions, allowing the user to get a PregnancyBean list of all prior pregnancies for a specific patient, and also to create a new prior pregnancy. This class is coupled with the PriorPregnancyBean class.

Bean Class

/iTrust/src/edu/ncsu/csc/itrust/beans/OIRBean.java
/iTrust/src/edu/ncsu/csc/itrust/beans/PregnancyBean.java

These are standard beans. Each bean instance corresponds to an OIR or a prior pregnancy respectively. They are used both in the retreiving and the storage of data.

Validation Class

/iTrust/src/edu/ncsu/csc/itrust/validate/OIRValidator.java
/iTrust/src/edu/ncsu/csc/itrust/validate/PregnancyValidator.java

These two validator classes are fairly self explanatory. Each will throw an exception if the provided bean does not meet certain criteria. For example, an exception will be thrown if the program attempts to validate a PregnancyBean where the year of conception is greater than the current year. Each class is coupled with it's respective Bean class.

Data Access Object Class

/iTrust/src/edu/ncsu/csc/itrust/dao/mysql/OIRDAO.java
/iTrust/src/edu/ncsu/csc/itrust/dao/mysql/PregnancyDAO.java

These are standard DAOs that interact directly with their respective database tables. OIRDAO interfaces with the obstetricsinitializationrecords table and PregnancyDAO interfaces with the priorpregnancies table. These DAOs are similar to their respective Action classes in that each method in the action class calls a unique method in the DAO class.

Database Schema

obstetricsinitializationrecords
priorpregnancies

Both of these tables are defined in createTables.sql (as well as their destruction in dropTables.sql). They also have sample data generation (OIR.sql and priorPregnancy.sql). You will notice that neither one is tied to the other, in fact both are tied to the patient.

Also, note in priorpregnancies that the length in time of the pregnancy is represented by two columns: numberOfWeeksPregnant and numberOfDaysPregnant. Both should be non-negative integers, with numberOfDaysPregnant never exceeding 6. Keep in mind that these constraints are enforced at the Bean level. A direct sql insertion will allow non-sensical data to be saved.

UC 64 Obstetrics Patient Office Visit

Brief Description

This use case allows the creation, editing, and viewing of obstetrics office visit documentation by HCPs. It is somewhat integrated into the existing method for documenting an office visit. Only OB/GYN HCPs may add or edit office visits, and patients must have an existing obstetrics initialization record (UC63) for a visit to be recorded.

For more information on this use case, please visit:
http://agile.csc.ncsu.edu/iTrust/wiki/doku.php?id=requirements:uc64

Architecture & Design

Front-end

/iTrust/WebRoot/auth/hcp-uap/documentOfficeVisit.jsp
/iTrust/WebRoot/auth/hcp-uap/editObstetricsVisit.jsp

The documentOfficeVisit page was updated to include links relating to obstetrics visit documentation. This includes a button for creating a new obstetrics visit as well as a listing of links to past obstetrics visits. Both the button and the links bring the user to the editObstetricsVisit page, which contains form fields for relevant obstetrics visit information. OB/GYN HCPs are allowed to create new visits and update old ones, while non-OB/GYN HCPs can only view the visits.

Bean Classes

/iTrust/src/edu/ncsu/csc/itrust/beans/ObstetricsVisitBean.java
/iTrust/src/edu/ncsu/csc/itrust/beans/forms/EditObstetricsVisitForm.java
/iTrust/src/edu/ncsu/csc/itrust/beans/loaders/ObstetricsVisitLoader.java

The ObstetricsVisitBean class is the model used to represent an obstetrics visit. The EditObstetricsVisitForm class is used for storing user input when the form on the editObstetricsVisit page is submitted. The ObstetricsVisitLoader class is used to generate an ObsetricsVisitBean from an SQL query result.

DAO Class

/iTrust/src/edu/ncsu/csc/itrust/dao/mysql/ObstetricsVisitDAO.java

This is a basic DAO used for retrieving obstetrics visits from the database. It has functionality for getting specific visits, listing visits for a particular patient, updating and adding new visits, and checking for the existence of a visit.

Action Classes

/iTrust/src/edu/ncsu/csc/itrust/action/base/ObstetricsVisitBaseAction.java
/iTrust/src/edu/ncsu/csc/itrust/action/AddObstetricsVisitAction.java
/iTrust/src/edu/ncsu/csc/itrust/action/EditObstetricsVisitAction.java

The ObstetricsVisitBaseAction class is extended by all of the action classes relating to obstetrics visits. The AddObstetricsVisitAction and EditObstetricsVisitAction classes are fairly self-explanatory, they are used to perform insertion and updates of obstetrics visits, respectively.

Validator Class

/iTrust/src/edu/ncsu/csc/itrust/validate/EditObstetricsVisitValidator.java

/iTrust/src/edu/ncsu/csc/itrust/validate/ValidationFormat.java

The EditObstetricsVisitValidator class is used to verify the validity of form input when a user makes a submission on the editObstetricsVisitPage. The rules used for validation are stored in the ValidationFormat class.

Database Schema

obstetricsofficevisits

There is a single table added to support this use case, and that is the obstetricsofficevisits table. This table contains all the necessary information for a given visit, including the patient ID and the HCP ID. Each visit is referenced using a unique ID.

UC 99 Ebola Risk Analysis

Brief Description

This user case is created in response to Ebola crisis as hospitals are not prepared to properly handle Ebola patients. Various failures to detect the risk of infecting Ebola have proved to be devastating and caused inconvenience to general population. This function calculates based on the travel history of patients, the risk of individuals infected by Ebola. Countries are classified into three categories according to the WHO Ebola-risk analysis. There are two ways for patients to input their travel history :

1. For registered patients, travel history is required by default when users are requesting an appointment. The doctor is warned about the Ebola-risk as part of the comment message on the approve request page. The transaction event is also logged.
2. For any patients, hospital staffs can run a quick analysis tool right from the iTrust Main Page. This subflow is especially designed to minimize the interaction between health care workers and possible Ebola patients. The transaction event is also logged with an additional field stating the patient is not registered.

Architecture & Design

Front-end

/iTrust/WebRoot/auth/patient/appointmentRequest.jsp
/iTrust/WebRoot/addon/ebolacheck.jsp

In *appointmentRequest.jsp*, a new text input form is added into the original iTrust file. This box is responsible for allowing users to input their travel history. Users are required to put comma between each country they travel to. Back-end checking is implemented to do error-checking.The method found inside the action class will be called when the users submit the form. The return value, which is the percentage risk, will be included in the comment field when submitted to the doctor. Logging method is also called from here.

ebolacheck.jsp is used to provide quick access to the Ebola risk analysis tool. The page is very similar to the above-menton jsp file without the requesting appointment functionality. The main difference is that this page will show the risk percentage on the top of the page after the method found inside Action class runs. Logging method is also called from here.

Action Class
/iTrust/src/edu/ncsu/csc/itrust/action/CheckEbolaAction.java

This class contains a method used to return the percentage of Ebola-risk. The parameter passed into the method is a String of countries that the patients travel to. The method will run a match-check between the input countries and the list of countries that have Ebola. Each country has a different value. The higher the value means the patient has a higher risk of contracting Ebola. This percentage is returned as an integer to the jsp files.

UC 20 View cause-of-death trends report

Brief Description
For this use case, a function was implement for LHCP. Two main functionalities are added to the existing iTrust system.

1. LHCP can login to search for a time period.
2. LHCP will then be able to see a report which provides a sorted list of the “top 2” most common causes of death for all patients within that searched time period LHCP are able to see either the report for all patients, all male patients only or all female patients only.

Both front-end and back-end error checking for the date input for searching of time period have also been implemented.

Architecture & Design

Front-end
iTrust/WebRoot/auth/hcp/menu.jsp
iTrust/WebRoot/auth/hcp-uap/viewCauseOfDeathTrendsReport.jsp
iTrust/WebRoot/auth/hcp-uap/viewListOfDeceasedPatients.jsp

menu.jsp files were used in order to add both “View Cause Of Death Trends Report” and “View List Of Deceased Patients” option in the sidebar menu when a LHCP logs in.

viewCauseOfDeathTrendsReport.jsp is responsible for getting the input from users. Various front-end error checks are implemented here. Date input fields must be filled. This was implemented using the html “required” tag. Logical error checking for date is also implemented here. The code checks for invalid month inputs, invalid date format, and also whether the begin date is before the end date. If all inputs are valid, the code will collect the data and pass it over to *viewListOfDeceasedPatients.jsp*. All field data are passed using the GET Method. The action class *ViewCauseOfDeathTrendsReportAction.java* will implement the search for collection of data.

viewListOfDeceasedPatients.jsp is responsible for displaying data in table format. It will display data collected from the database based on the search of time period inputted by the user in *viewCauseOfDeathTrendsReport.jsp*. The action class *ViewListOfDeceasedPatientsAction.java* will then implement the display details.

Action Class
iTrust/src/edu/ncsu/csc/itrust/action/ViewCauseOfDeathTrendsReportAction.java
iTrust/src/edu/ncsu/csc/itrust/action/ViewListOfDeceasedPatientsAction.java

Methods in *ViewCauseOfDeathTrendsReportAction.java* are responsible for implementing logical error checking for data inputs as well as searching the database for the filtered data.

Methods in *ViewListOfDeceasedPatientsAction.java* are responsible for displaying data based on the filtered data according to the user input.

UC 30 Message Displaying Filter

Brief Description

This use case is an extension of the basic messaging feature that already existed. The purpose of this use case is to make the messaging feature of the iTrust website more powerful and user-friendly by giving users (Health Care Providers as well as Patients) the ability to quickly find relevant information among their messages by means of filters, as well as to save preferred filters between logins.

Architecture & Design

Front-end

/iTrust/WebRoot/auth/hcp-patient/mailbox.jsp
/iTrust/WebRoot/auth/hcp-patient/messageInbox.jsp
/iTrust/WebRoot/auth/hcp-patient/messageOutbox.jsp

Users can navigate to their Inbox or Outbox from the side menu, which takes them either to messageInbox.jsp or messageOutbox.jsp, both of which make use of mailbox to display messages. The filtering functionality is added to mailbox to complement its existing capabilities.

These steps describe how to make use of the filtering feature:

1. The filter can be accessed by clicking *Edit Filter* in the messaging page.
2. The displayed filtering fields may be filled out or left blank.
3. Click *Test Filter* to apply the filter. This will filter the messages in the inbox temporarily, and the filter is not saved between sessions.
4. Click *Cancel* to return to the previously saved filter (default: unfiltered).
5. Click *Save* to save the filter. The filter will be saved between sessions, and messages will be displayed according to this filter between login sessions, until the user decides to change the filter.

Bean Class

/iTrust/src/edu/ncsu/csc/itrust/beans/PatientBean.java
/iTrust/src/edu/ncsu/csc/itrust/beans/PersonnelBean.java

These beans are extended to store a user's saved message filter.

Action Class

/iTrust/src/edu/ncsu/csc/itrust/action/ViewMyMessagesAction.java
/iTrust/src/edu/ncsu/csc/itrust/action/EditPatientAction.java
/iTrust/src/edu/ncsu/csc/itrust/action/EditPersonnelAction.java

EditPatientAction.java and *EditPersonnelAction.java* are extended to implement editing and saving patients' and HCP's filtering preferences respectively. These classes interact directly with the database to achieve this.

ViewMyMessagesAction.java is extended to support filtering as well as sorting messages, and is responsible for retrieving messages from the database according to the filtering criteria specified by the user.

Database Schema

A column named *Filter* has been added to the *patients* table and the *personnel* table, as well as to the *HistoryPatients* table. The preferred filtering criteria for each user is stored along with other information about the user to facilitate customizing each user's mailbox according to his or her preferences.

UC 41 Send Reminders

Brief Description

This use case implements a tool for mass appointment reminder distribution and a tool to view sent reminders.

Architecture & Design

Rather than implement a new reminders table that would ultimately duplicate the existing functionality of the messages table, the decision was made to program System Reminders as a special MID 0 user. This approach is intuitive, simple to implement, and least disrupts the existing design; for example, we can ignore questions such as, "Should System Reminder be considered an HCP?"

Front-end

Admins can navigate to either /auth/admin/sendReminders.jsp or /auth/admin/remindersOutbox.jsp through the side menu defined in /auth/admin/menu.jsp.

/auth/admin/sendReminders.jsp contains a form containing a textfield that accepts a numeric value. /auth/admin/remindersOutbox.jsp contains a table of sent reminders where each row includes the message subject, name of the recipient, timestamp, and a "Read" link. The messages are ordered by timestamp, the most recent first. Each "Read" link navigates to a detailed view containing the message subject, name of the recipient, timestamp, and message content.

DAO Classes

/iTrust/src/edu/ncsu/csc/itrust/dao/mysql/ApptDAO.java

A method was added to the ApptDAO to retrieve all upcoming appointments within a given number of days.

Action Classes

/iTrust/src/edu/ncsu/csc/itrust/action/ViewMyMessagesAction.java
/iTrust/src/edu/ncsu/csc/itrust/action/SendRemindersAction.java

ViewMyMessagesAction.getName(mid) has been modified to return a static string, "System Reminder", when the MID parameter is 0.

A new action, SendRemindersAction, accompanies the Admin front-end page and invokes the ApptDAO to retrieve upcoming appointments and then invokes the MessageDAO to send messages to the associated patients with the appropriate reminder content.

Enums

iTrust/src/edu/ncsu/csc/itrust/enums/TransactionType.java

New enums were added to TransactionType to facilitate logging of Admins that utilize the Send Reminders tool.

Appendix A: Unit Tests & HTTP Tests

UC 63 Obstetrics Patient Initialization Tests

1. iTrust/unittests/edu/ncsu/csc/itrust/action/OIRActionTest.java
2. iTrust/unittests/edu/ncsu/csc/itrust/action/PregnancyActionTest.java
3. iTrust/unittests/edu/ncsu/csc/itrust/bean/OIRBeanTest.java
4. iTrust/unittests/edu/ncsu/csc/itrust/bean/PregnancyBeanTest.java
5. iTrust/unittests/edu/ncsu/csc/itrust/dao/pregnancy/OIRDAOTest.java
6. iTrust/unittests/edu/ncsu/csc/itrust/dao/pregnancy/PregnancyDAOTest.java
7. iTrust/httptests/edu/ncsu/cs/itrust/CS427_TEAM1/OIRPageTest.java
8. iTrust/httptests/edu/ncsu/cs/itrust/CS427_TEAM1/PregnancyPageTest.java

UC 64 Obstetrics Patient Office Visit Tests

1. iTrust/unittests/edu/ncsu/csc/itrust/action/EditObstetricsVisitActionTest.java
2. iTrust/unittests/edu/ncsu/csc/itrust/action/AddObstetricsVisitActionTest.java
3. iTrust/unittests/edu/ncsu/csc/itrust/bean/ObstetricsVisitBeanTest.java
4. iTrust/unittests/edu/ncsu/csc/itrust/dao/ObstetricsVisitDAOTest.java
5. iTrust/httptests/edu/ncsu/cs/itrust/CS427_TEAM1/ObstetricsVisitTest.java

UC 39 View Transaction Logs Tests

1. iTrust/unittests/edu/ncsu/csc/itrust/action/ViewTransactionActionTest.java
2. iTrust/unittests/edu/ncsu/csc/itrust/dao/viewtransaction/GetSelectedTransactionsTest.java
3. iTrust/unittests/edu/ncsu/csc/itrust/bean/ViewTransactionBeanTest.java
4. iTrust/httptests/edu/ncsu/cs/iTrust/CS427_TEAM2/ViewTransactionLogTEST.java
5. iTrust/httptests/edu/ncsu/cs/iTrust/CS427_TEAM2/ViewTransactionLogForTester.java

UC 14 Request Biosurveillance Tests

1. iTrust/unittests/edu/ncsu/csc/itrust/action/RequestBiosurveillanceTrendActionTest.java
2. iTrust/unittests/edu/ncsu/csc/itrust/action/RequestBiosurveillanceAnalysisActionTest.java
3. iTrust/unittests/edu/ncsu/csc/itrust/bean/BiosurveillanceBeanTest.java
4. iTrust/unittests/edu/ncsu/csc/itrust/validate/regex/ThresholdValidatorTest.java
5. iTrust/httptests/edu/ncsu/cs/iTrust/CS427_TEAM2/BioSurveillanceTest.java

UC 99 Ebola Risk Analysis Tests

1. iTrust/unittests/edu/ncsu/csc/itrust/action/CheckEbolaActionTest.java
2. iTrust/httptests/edu/ncsu/cs/iTrust/CS427_TEAM2/EbolaCheckTest.java

UC 20 View cause-of-death trends report Tests

1. iTrust/httptests/edu/ncsu/cs/iTrust/CS427_TEAM3/DeathSearchTest.java
2. iTrust/httptests/edu/ncsu/cs/iTrust/CS427_TEAM3/DeceasedPatientsTest.java

UC 30 Message Displaying Filter

1. iTrust/unittests/edu/ncsu/csc/itrust/action/EditPatientActionTest.java
2. iTrust/unittests/edu/ncsu/csc/itrust/action/EditPersonnelActionTest.java

3. iTrust/unittests/edu/ncsu/csc/itrust/action/ViewMyMessagesActionTest.java
4. iTrust/httptests/edu/ncsu/cs/itrust/CS427_TEAM3/MessageFilterTest.java

UC 41 Send Reminders

1. iTrust/unittests/edu/ncsu/csc/itrust/action/SendRemindersActionTest.java
2. iTrust/unittests/edu/ncsu/csc/itrust/action/ViewMyMessagesActionTest.java
3. iTrust/unittests/edu/ncsu/csc/itrust/dao/appointment/ApptDAOTest.java
4. iTrust/httptests/edu/ncsu/cs/itrust/CS427_TEAM3/SendRemindersTest.java

Manual Tests

Please refer to *https://wiki.cites.illinois.edu/wiki/display/cs427fa14/T812+Manual+Testing* for step-by-step guide.(Only tested with Google Chrome and Mozilla Firefox.)