Table of Contents

Project Description

The goal of this project is to allow doctors and patients to communicate effectively and keep essential patient related data in one secure application. This application allows easier organization of data and saves paper. Our group was responsible for a number of new use cases which ranged from viewing cause of death trends, filtering messages and sending appointment reminders to organizing obstetric patient visits. We also came up with a new use case in which the tester role can have the ability to enable or disable buggy features, track application errors, and track server request and response information.

Use Case 41 - Send Reminders

Files Created:

- edu.ncsu.csc.itrust.action.SendReminder.java
- Webroot.auth.admin.sendReminder.jsp

Tests Created:

- Http: edu.ncsu.csc.itrust.Team805.UC41.SendReminderTest.java
- Unit: edu.ncsu.csc.itrust.Team805.UC41.SendReminderActionTest.java

Files Modified:

- Added the function getPatientsForReminders to PatientDAO.java (see javadocs)
- Added the function getAppointmentsForReminders to ApptDAO.java (see javadocs)

Purpose: An admin is able send appointment reminders to patients that have an upcoming appointment in a certain number of days (value is specified by the admin). The patients that have an appointment coming up within the specified number of days will receive an email and a message connected to their account.

Things to know: We created a separate administrator in the database (MID is 9000000089) that should only be used to send appointment reminder messages. Note this admin is different than the admin that chose to send the reminder messages. The reason for this is so it says the message is from “System Reminder”.

Testing: All testing is done for the use case does not require manual testing. The tests are automated and the unittest covers over 80% for the SendReminder class.

UC 20 - View cause-of-death trends report

Files Created: CODTrendsDAO.java, viewCODTrends.jsp
Tests Created:

- Http: edu.ncsu.csc.itrust.Team805.UC20.ViewCODTrendsTest.java
- Unit: edu.ncsu.csc.itrust.Team805.UC20.CODTrendsTest.java
- Files: Various SQL files were created for testing, including fake patients dying from random diseases listed in the icdcodes table. Since we created these new files, TestDataGenerator.launch was modified to include these new files.

Purpose: The goal of implementing UC 20 was to allow authorized HCPs to view certain trends in the deaths among patients.

Things to know: We changed the hcp menu file to be able to navigate to the cause of death trends report page. It is listed in the category “Other” on the hcp’s menu. The page takes into account the fake patients created for this use case. Do not generate test data when finding cause of death trends among actual patients.

Testing: All testing done for this use case does not require any manual testing. The tests are all automated, with over 80% of lines in CODTrendsDAO.java covered under the unit tests. These unit tests test a variety of factors that an HCP can search for. These include testing a search for males, females, all, and for a specific HCP’s id.

In the httptest ViewCODTrendsTest.java, we tested some sample inputs with different possible parameters: female, male and all. We also tested different starting and ending year inputs, and made sure it returns nothing if ending year is earlier than starting year.

UC 14 - Request Biosurveillance

Files Created: EpidemicDAO.java, requestBiosurveillance.jsp
Tests Created:

- Http: edu.ncsu.csc.itrust.Team805.UC14.RequestBiosurveillanceTest.java
- Unit: edu.ncsu.csc.itrust.Team805.UC14.EpidemicDAOTest.java
- Files: Some files already in iTrust were used to create the test data. iTrust had files for creating patients with malaria and influenza, however, these files were not being run when the test data was created. We changed TestDataGenerator.launch to run these files.

Purpose: The goal of implementing UC 14 was to allow authorized HCPs to determine if an epidemic was currently in progress.

Things to know: We changed the hcp menu file to be able to navigate to the request biosurveillance page. It is listed in the category “Other” on the hcp’s menu. Choosing a future date for determining the epidemic is not allowed, as well as choosing an invalid ICD code and an invalid zip code. The table below the search criteria will inform the HCP if any of these are invalid.

Testing: All testing done for this use case does not require any manual testing. The tests are all automated, with over 80% of lines in EpidemicDAO.java covered under the unit tests. The tests for UC14 cover many possible scenarios for the functions in EpidemicDAO.java. For functions that return booleans, tests are written for both possible return values.

In the httptest RequestBiosurveillanceTest.java, we tested if the Malaria and the Influenza with valid zip code and date work as expected for an hcp. We also generated some invalid data to see if proper error messages are displayed.

UC 63 - Obstetrics Patient Initialization

Files Created: ObstetricsBean.java, ObstetricsDAO.java, addOBRecord.jsp, OBRecord.jsp, OBInitialization.jsp
Tests Created:

- Http: edu.ncsu.csc.itrust.Team805.UC63.OBInitializationTest.java
- Unit: edu.ncsu.csc.itrust.Team805.UC63_64. ObstetricsTest.java
- Files: Some SQL files were created for testing purposes. These files are named with the format “obinit…”, and hcpUC63l. A new table, obinit, was also created.

Purpose: The goal of implementing UC 63 was to allow authorized HCPs(with a specialization of “OB/GYN”) to add a new obstetrics patient to the database.

Things to know: We changed the hcp menu file to be able to navigate to the obstetrics patient initialization page. It is listed in the category “Other” on the hcp’s menu. Choosing a male patient on the search page will receive an error message. Only an HCP with the specialty “OB/GYN” can add new initialization records.

Testing: All testing done for this use case does not require any manual testing. The tests are all automated, with over 80% of lines in EpidemicDAO.java covered under the unit tests. These tests also check that errors with the database are properly caught and handled.

In the httptest OBInitializationTest.java, we tested if an hcp with specialization OB/GYIN can add a new OB record, while hcps with other specializations cannot. We also tested if all hcps can read the OB record, and if the patient selected is a male, an error message is displayed.

UC 64 - Obstetrics Patient Office Visit

Files Created: OBVisit.jsp, addOBVisit.jsp
Files Edited: OBInitialization.jsp, ObstetricsBean.java, ObstetricsDAO.java
Tests Modified:

- Http: edu.ncsu.csc.itrust.Team805.UC64.OBVisitTest.java
- Unit: edu.ncsu.csc.itrust.Team805.UC63_64.ObstetricsTest.java
- Files: obvisit1.sql. TestDataGenerator.launch was modified to include this file.

Purpose: The goal of implementing UC 64 was to allow an HCP to choose to add or edit an obstetrics office visit for a current obstetrics patient.

Things to know: To add a visit, an HCP logs in and chooses the category “Other” on the side menu. They then click the “Obstetrics Patient Initialization” link that was created in UC63. When adding or updating the visit, the types of each form field as specified in the use case documentation, should be enforced by HTML5.

Testing: All testing done for this use case does not require any manual testing. The tests are all automated, with over 80% of lines in ObstetricsDAO.java covered under the unit tests. The following functions were added to the existing tests from UC63: testOBVisit(), testOBVisitUpdate(), testNoOBVisitDates().

In the httptest, we tested adding a visit, updating a visit, and the case where the patient initializations fall within a 49 week window as described in the iTrust use case documentation.

Use Case 65 - Application Health

Files Created:

- edu.ncsu.csc.itrust.dao.mysql.FeatureDAO.java
- edu.ncsu.csc.itrust.beans.FeatureBean.java
- edu.ncsu.csc.itrust.beans.loaders.FeatureBeanLoader.java
- Webroot.auth.tester.featureControl.jsp
- edu.ncsu.csc.itrust.server.FeatureControlServlet.java
- edu.ncsu.csc.itrust.dao.mysql.ErrorDAO.java
- edu.ncsu.csc.itrust.beans.ErrorBean.java
- edu.ncsu.csc.itrust.beans.loaders.ErrorBeanLoader.java
- Webroot.auth.testError1.jsp
- Webroot.auth.testError2.jsp
- Webroot.auth.testError3.jsp
- Webroot.auth.testError4.jsp
- Webroot.auth.testError5.jsp
- Webroot.auth.tester.viewErrors.jsp
- edu.ncsu.csc.itrust.dao.mysql.RequestDAO.java
- edu.ncsu.csc.itrust.beans.RequestBean.java
- edu.ncsu.csc.itrust.beans.loaders.RequestBeanLoader.java
- edu.ncsu.csc.itrust.dao.mysql.ResponseDAO.java
- edu.ncsu.csc.itrust.beans.ResponseBean.java
- edu.ncsu.csc.itrust.beans.loaders.ResponseBeanLoader.java
- edu.ncsu.csc.itrust.server.ResponseTrackerServlet.java
- Webroot.auth.tester.viewNetworkInformation.jsp

Tests Created:

- Http: edu.ncsu.csc.itrust.Team805.UC65.FeatureControlTest.java,
  edu.ncsu.csc.itrust.Team805.UC65.ErrorListTest.java,
  edu.ncsu.csc.itrust.Team805.UC65.NetworkInformationTest.java

- Unit: edu.ncsu.csc.itrust.Team805.UC65.RequestDAOTest.java,
  edu.ncsu.csc.itrust.Team805.UC65.ResponseTest.java,
  edu.ncsu.csc.itrust.Team805.UC65.ResponseDAOExceptionTest.java,
  edu.ncsu.csc.itrust.Team805.UC65.ErrorTest.java,
  edu.ncsu.csc.itrust.Team805.UC65.ErrorDAOExceptionTest.java,
  edu.ncsu.csc.itrust.Team805.UC65.FeatureTest.java,
  edu.ncsu.csc.itrust.Team805.UC65.FeatureDAOExceptionTest.java

Files Modified:

- Webroot.header.jsp (added capturing errors and request/response information)
- Webroot.footer.jsp (added request/response information)
- Webroot.global.jsp (added FeatureDAO initialization)

Purpose: Upon logging in as the tester role, the tester will see a “Error List” button, a “Network Information” button, and “Feature Control” button. After clicking on a button it will take the tester to the corresponding page. The “Feature Control” page will show the list of implemented features and their status (either enabled or disabled) and will see buttons next to each feature to enable or disable the feature. The “Error List” page will display a table of information related to errors encountered by the application. Lastly, the “Network Information” page will show statistics regarding request information and will display it in both table and chart form.

Things to know: The servlet files are only used for ajax related calls (see footer.jsp and featureControl.jsp for the ajax calls). The testError jsp files are used for manually triggering some common exceptions for testing purposes. Note, if exceptions occur before login, the default mid for the error will be 1.

Testing: All testing is done for the use case does not require manual testing. The tests are automated and the unittest covers over 80% for the the following java classes: ErrorDAO, FeatureDAO, RequestDAO, and ResponseDAO.

In the http tests, we tested enabling/disabling features, viewing a list of errors, and viewing various network information as described in the use case description.

UC 30 - Message Filter

Files Modified:

- WebRoot.auth.hcp.messageInbox.jsp
- WebRoot.auth.patient.messageInbox.jsp
- src.edu.ncsu.csc.itrust.action.ViewMyMessagesAction.java (validateAndCreateFilter)
- src.edu.ncsu.csc.itrust.action.EditPatientAction.java (editMessageFilter)
- src.edu.ncsu.csc.itrust.action.EditPersonnelAction.java (editMessageFilter)
- src.edu.ncsu.csc.itrust.beans.PatientBean.java (getMessageFilter, setMessageFilter)
- src.edu.ncsu.csc.itrust.beans.PersonnelBean.java (getMessageFilter, setMessageFilter)
- src.edu.ncsu.csc.itrust.dao.mysql.PatientDAO.java (editPatient)
- src.edu.ncsu.csc.itrust.dao.mysql.PersonnelDAO.java (editPersonnel)
- src.edu.ncsu.csc.itrust.beans.loaders.PatientLoader.java
- src.edu.ncsu.csc.itrust.beans.loaders.PersonnelLoader.java

Tests Created:

- Http:
    - edu.ncsu.csc.itrust.Team805.UC30.MessagingFilterTest.java
- Unit:
    - edu.ncsu.csc.itrust.Team805.UC30.EditPatientActionFilterTest.java
    - edu.ncsu.csc.itrust.Team805.UC30.EditPatientFilterTest.java
    - edu.ncsu.csc.itrust.Team805.UC30.EditPersonnelActionFilterTest.java
    - edu.ncsu.csc.itrust.Team805.UC30.EditPersonnelFilterTest.java
    - edu.ncsu.csc.itrust.Team805.UC30.ViewMyMessagesFilterTest.java

Purpose: The goal of implementing UC 30 S7 was to allow HCP's and Patients to create/edit and save a filter for their message inbox.

Things to know: To create/edit a filter, an HCP or patient logs in and chooses the category "Messaging" and select "Message Inbox" on the side menu. They then click the "Edit Filter" category that was created in UC30. After writing the filter criterias the "Test Filter" button can be clicked to test if the filter results are as expected, then the filter preferences can be saved. The "Cancel" button discards any changes to the filter and uses the old filter. Filters are kept between login sessions.

Testing: All testing done for this use case does not require any manual testing. The tests are all automated, and covers functionality that was changed. All functions created and added to existing files as described on page 10 (in parentheses next to the file names) were tested for 80% coverage. In the httptest, we tested applying a message filter and saving a message filter as described in the iTrust use case documentation.

Use Case 39 - View Transaction Logs

Files Created:

- WebRoot.auth.admin.transactionLog.jsp
- WebRoot.auth.tester.transactionLog.jsp

Tests Created:

- httptests.edu.ncsu.csc.itrust.Team805.UC39.TransactionlogTest.java

Files Modified:

- WebRoot.auth.admin.menu.jsp
- WebRoot.auth.tester.menu.jsp

Purpose: Upon logging in as a tester/admin role, the tester/admin will see a “Transaction Log” button under the “Other”/“View” Menu. The “Transaction Log” page has a search form for viewing transactions. After filling out the form, the user has a choice between the “View” button or the “Summarize” button. The “View” button will return a list of transactions satisfying the search criteria. The “Summarize” button will display charts that satisfy the search criteria along with the list.

Things to know: The bottom of the page contains a “Transaction Code Reference”.

Testing: All testing is done for the use case does not require manual testing. The tests are all automated.

The http tests covers the expected behaviors of the webpage. We tested searching for many roles such as ER, HCP and PATIENT.

Appendix

New use case detailed description (UC 65):
https://wiki.cites.illinois.edu/wiki/display/cs427fa14/T805+Use+Case+65?src=contextnavchildmode

Testing:

All unit tests can be found in the package “edu.ncsu.csc.iTrust.Team805” under the “unittests” source folder. All tests in the package can be run at the same time by running All805UnitTests.java. There should be subpackages pertaining to each use case (package edu.ncsu.csc.iTrust.Team805.UCXX).

All http tests can be found in the package “edu.ncsu.csc.iTrust.Team805” under the “httptests” source folder. All tests in the package can be run at the same time by running All805HttpTests.java. There should be subpackages pertaining to each use case (package edu.ncsu.csc.iTrust.Team805.UCXX).

Notes on unit tests:

- UC39 does not require any unit tests
- For UC41, the function getPatientsForReminders was added to PatientDAO.java and the function getAppointmentsForReminders was added to ApptDAO.java. No additional files were needed because both of these functions get 80% coverage while running SendReminderTest.java.