CMPUT 391 Project Report

Group 7

Yilu Su

Xuping Fang

Anni Dai

March 31, 2014

**Project Report**

**Introduction:**

This is a CMPUT 391 course project. In this project, a web-based database application system, called Radiology Information System, is developed. It is a three tier system: the database server, the web server, and clients are running on different computer systems that are connected via the Internet. JSP is used to interface the website to the database server. We use Oracle in the lab as database server, and the Tomcat as the web server. Clients can access our system using a web browser through the Internet.

**Modules:**

**Login Module:**

This module will be used by all users to login to the system with proper privileges, and to modify their personal information and/or the password.

**Login.html Login.jsp**

The login page prompts for *user name* and *password*. The system will get the correct password from the *user* table in database using the input user name. If the input password does not match the correct password, error message will show up. If the input password match the correct password, the *class* and *person\_id* will be retrieved from the *person* table in the database. After the *class* and *person\_id* are passed into the session, user will be linked to AdminPage.jsp, RadPage.jsp, Doctor.jsp, or PatientPage.jsp according to the class type.

Get password from database: SELECT password FROM users WHERE user\_name="username"

Get class and person\_id: SELECT class,person\_id FROM users WHERE user\_name="userName"

**PersonalManage.jsp ChangeProfileProcess.jsp**

After login, clicking on the person name at the top of the page will jump to PersonalManage.jsp, which allow users to modify their personal information and/or password. Firstly, the person\_id will be retrieved from the *users* table. Then, all the personal information will be retrieved from the *persons* table using the *person\_id*, and displayed. ChangeProfileProcess.jsp will be used to update the personal information to the database after the UPDATE button is clicked. Corresponding page is return to after the update.

Get personal information: SELECT \* FROM persons WHERE person\_id="personId" and getString method.

To update a field: SELECT "tag" FROM persons WHERE person\_id="person\_id" FOR UPDATE (tag is what to update) Also use updateString and updateRow methods.

invalidate any cookies when the user logs out.

**User Management Module:**

This module allows a system administrator to manage (to enter or update) the user information, i.e., the information stored in tables *users,persons, family\_doctor*.

**ResetAccount.jsp**

ResetAccount.jsp is directed from AdminPage.jsp. It is used to update the *users* table. It displays the current information and uptake inputs, and then updates the user information to the database after the UPDATE button is clicked. AdminPage.jsp will be returned after update.

To update user information: SELECT user\_name,password,class FROM users WHERE user\_name="old\_user\_name" FOR UPDATE". Also use updateString and updateRow methods.

**ManagePerson.jsp CommitManagePersonInfo.jsp**

ManagePerson.jsp is directed from AdminPage.jsp. It is used to update the *persons* table. It displays the current information and uptake inputs. CommitManagePersonInfo.jsp will be used to update the personal information to the database after the UPDATE button is clicked. AdminPage.jsp will be returned after update.

To update person information: SELECT "tag" FROM persons WHERE person\_id="person\_id" FOR UPDATE (tag is what to update) Also use updateString and updateRow methods.

**AddUser.jsp CommitAddUser.jsp AddPerson.jsp CommitAddPerson.jsp**

AddUser.jsp or AddPerson.jsp is directed from AdminPage.jsp. They are used to create new user. AddUser.jsp and AddPerson.jsp take the input and CommitAddUser.jsp and CommitAddPerson.jsp insert the new data to the *users* or *persons* table in the database, respectively. AdminPage.jsp will be returned after insert.

Get patient information: SELECT person\_id,first\_name,last\_name FROM persons

alter SESSION set NLS\_DATE\_FORMAT = 'YYYY-MM-DD'

Get the max person\_id number: SELECT MAX(person\_id) FROM persons

Insert data to *users* table: INSERT INTO users VALUES("userName", "password", "userClass", "id", "current")

Insert data to *persons* table: INSERT INTO users VALUES("personId", "firstName", "lastName", "address", "email", "phone")

**ManageFamilyDoctor.jsp CommitUpdateFamilyDoctor.jsp**

These two files are used to add/remove a family doctor to/from a patient. Operation type and patient id will be passed as parameter to ManageFamilyDoctor.jsp. Firstly, it check the operation type. Then use CommitUpdateFamilyDoctor.jsp to update *family\_doctor* table to add a family doctor to the patient or remove a family doctor from the patient, according to the operation type. AdminPage.jsp will be returned after operation.

Get personal information of doctors that are not the family doctor of a patient: (SELECT \* FROM persons p WHERE p.person\_id = ANY(SELECT u.person\_id FROM users u WHERE u.class='d')) MINUS (SELECT \* FROM persons p2 WHERE p2.person\_id = ANY(SELECT fd.doctor\_id FROM family\_doctor fd WHERE fd.patient\_id="patient\_id"))

Get personal information of doctors that are the family doctor of a patient: SELECT \* FROM persons p2 WHERE p2.person\_id = ANY(SELECT fd.doctor\_id FROM family\_doctor fd WHERE fd.patient\_id="patient\_id")

Insert data to *family\_doctor* table: INSERT INTO family\_doctor VALUES ("doctor\_id"," patient\_id")

Delete data from *family\_doctor* table: DELETE FROM family\_doctor WHERE doctor\_id="doctor\_id" AND patient\_id="patient\_id"

**Report Generating Module:**

This module will be used by a system administrator to get the list of all patients with a specified diagnosis for a given time period.

**Report.jsp**

This page is directed from AdminPage.jsp. It provides the function for administrators to input diagnosis and time period, search, and generate report. “Cancel” button will direct the administrator back to the **AdminPage.jsp.**

To search data from the database based on the input condition: SELECT first\_name, last\_name, address, phone, min(test\_date) FROM persons p, radiology\_record r WHERE r.patient\_id = p.person\_id AND r.diagnosis = ? AND r.test\_date >= to\_date(?,'MM/DD/YYYY') AND r.test\_date <= to\_date(?,'MM/DD/YYYY') Group by patient\_id, first\_name, last\_name, address, phone

**Uploading Module:**

This module will be used by radiologists to first enter a radiology record, and then to upload medical images into the radiology record.

**Upload.jsp Upload\_Processor.jsp**

These two files are used to upload a radiology record. Upload.jsp is directed from RadPage.jsp. It prompt radiologist for a radiology record. After the input is taken, Upload.jsp passes *patientId*, *doctorId*, *testType*, *pDate*, *tDate*, *diagnosis*, and *description* as parameters to Upload\_Processor.jsp, which is used to upload the record. After Upload\_Processor.jsp get the parameters, it finds the *radId* and generate a *recordId*. The new record is then inserted to the *radiology\_record* table in the database. It will direct the radiologist to UploadPic.jsp to upload medical images after the record is uploaded.

To get personal information of all patients: SELECT \* FROM persons p WHERE p.person\_id = ANY(SELECT u.person\_id FROM users u WHERE u.class='p')

To get personal information of all doctors: SELECT \* FROM persons p WHERE p.person\_id = ANY(SELECT u.person\_id FROM users u WHERE u.class='d')

To generate a *recordId*: SELECT MAX(record\_id) FROM radiology\_record then add 1 to the number.

To insert data to *radiology\_record* table: INSERT INTO radiology\_record VALUES("recordId", "patientId", "doctorId", "radId", "testType", "prescribing\_date", "test\_date", "diagnosis", "description")

**UploadPic.jsp UploadPic\_Processor.jsp**

These two files are adapted from the example code on the course website. They are used upload medical images into the radiology record. UploadPic.jsp is directed from Upload\_Processor.jsp. It displays buttons for choosing files and direct to UploadPic\_Processor.jsp. UploadPic\_Processor.jsp take the images from local file system and store them in *BufferedImage* object. It also create the normal sized and thumbnails of those images, which are stored in other *BufferedImage* objects. After generating an image\_id, a row is prepped into the pacs\_images table. Then the BLOB stream is got, and images in different size are written to their corresponding BLOB using ImageIO.write method. All the streams are closed after the upload is done, and it will direct the page to RadPage.jsp.

To get the maximum of image\_id: SELECT MAX(image\_id) FROM pacs\_images WHERE record\_id="recordId”

To insert a row in the pacs\_images table: INSERT INTO pacs\_images VALUES("recordId", "(maxImgId+1)", empty\_blob(), empty\_blob(), empty\_blob())

To get the BLOB stream: SELECT \* FROM pacs\_images WHERE record\_id="recordId" AND image\_id="(maxImgId+1)" FOR UPDATE

External library provided by in the sample: [commons-fileupload-1.0.jar](http://luscar.cs.ualberta.ca:8080/yuan/commons-fileupload-1.0.jar)

**Search Module:**

This module will be used by all the registered users to search the database for a list of relevant radiology records and to view medical images with the zoom-in facility.

**Search.jsp**

This page can be directed from AdminPage.jsp, RadPage.jsp, DoctorPage.jsp or PatientPage.jsp. It displays a form that prompt for the search conditions. The user can enter a comma-seperated list of keywords, optionally choose a desired time period, and select an order option to display the results. An initial sql query is created and optional conditions will be appended to the end of the string.

Initial sql query if no search kays are entered: SELECT DISTINCT r.record\_id,r.patient\_id, r.doctor\_id, r.radiologist\_id, r.test\_type, r.test\_date, r.prescribing\_date,r.diagnosis, r.description FROM radiology\_record r WHERE

Initial sql query if search keys are entered: SELECT s.rank, r.record\_id, r.patient\_id, r.doctor\_id, r.radiologist\_id, r.test\_type, r.test\_date, r.prescribing\_date, r.diagnosis, r.description FROM radiology\_record r, (SELECT DISTINCT max(score(1)\*6 + score(2)\*3 + score(3)) as rank, r1.record\_id FROM radiology\_record r1, fullname f WHERE CONTAINS(f.full\_name, "search\_key", 1)>0 or CONTAINS(r1.diagnosis, "search\_key", 2) >0 or CONTAINS(r1.description, "search\_key", 3)>0 AND f.person\_id = r1.patient\_id GROUP BY r1.record\_id) s WHERE r.record\_id = s.record\_id

Class type of the user will be appended:

If the user is a patient, add AND r.patient\_id = "personID"

If the user is a doctor, add AND r.doctor\_id = "personID"

If the user is a radiologist, add AND r.doctor\_id = "personID"

Note that AND will not be appended if no search keys are entered.

Time period will also be appended if specified:

If a From date is specified, add AND r.test\_date >= to\_date("from", 'MM/DD/YYYY')

If a To date is specified, add AND r.test\_date <= to\_date("to", 'MM/DD/YYYY')

Order option will also be appended:

If order by default: ORDER BY rank desc

If order by most-recent-first: ORDER BY test\_date desc

If order by most-recent-last: ORDER BY test\_date

Note that order option will not be appended if no search keys are entered.

After the sql query fully created, Search.jsp prepare for the query execution.

Drop the *fullname* table: DROP TABLE fullname

Create a new *fullname* table using the information from *person* table: CREATE TABLE fullname AS SELECT person\_id, CONCAT(CONCAT(first\_name, ' '),last\_name) as full\_name FROM persons)

CreateIndexName: CREATE INDEX name ON fullname(full\_name) INDEXTYPE IS CTXSYS.CONTEXT

After the query is executed, result set is placed in a table. One record information on each raw, along with the attached medical images.

To get image\_id: select image\_id from pacs\_images where record\_id = ?

**ZoomPic.jsp GetOnePic.java**

**GetOnePic.java is adapted from the example code on the course website. Those two files are used to retrieve the image size, record\_id, and image\_id, and display the specified images.**

To get thumbnail: SELECT thumbnail FROM pacs\_images WHERE record\_id="recordID" AND image\_id="imageID”

To get full size: SELECT full\_size FROM pacs\_images WHERE record\_id="recordID" AND image\_id="imageID”

To get regular size: SELECT regular\_size FROM pacs\_images WHERE record\_id="recordID" AND image\_id="imageID”

**Data Analysis Module:**

This module will be used by the system administrator to generate and display an OLAP report for data analysis. A user of this module may choose to display the number of images for each patient , test type, and/or period of time.

**Analysis.jsp**

Analysis.jsp is directed from AdminPage.jsp. It prompt for conditions, including patient, Test Type, Time period, and group options for the data analysis. It passes the conditions as parameters and direct to ConfirmAnalysis.jsp. The "cancel" button return page back to AdminPage.jsp.

To get all patient for administrator to choose from: SELECT \* FROM persons p WHERE p.person\_id=ANY(SELECT u.person\_id FROM users u WHERE u.class='p')

**ConfirmAnalysis.jsp**

ConfirmAnalysis.jsp is used to retrieve data from the database and display results of the data analysis. If a patient is specified, the full name should be retrieved from the database: SELECT p.first\_name,p.last\_name FROM persons p WHERE p.person\_id="patientID”

An initial sql query is created according to the group option and optional conditions will be appended to the end of the string.

Initial sql query with no group option specified: SELECT count(\*) AS CNT FROM pacs\_images pi,radiology\_record rr WHERE pi.record\_id=rr.record\_id

Initial sql query for group by patient: SELECT count(\*) AS CNT,p.first\_name,p.last\_name,rr.patient\_id FROM pacs\_images pi,radiology\_record rr,persons p WHERE pi.record\_id=rr.record\_id AND rr.patient\_id=p.person\_id "+extra+" GROUP BY p.first\_name,p.last\_name,rr.patient\_id

Initial sql query for group by testType: SELECT count(\*) AS CNT,rr.test\_type FROM pacs\_images pi,radiology\_record rr WHERE rr.record\_id=pi.record\_id "+extra+" GROUP BY rr.test\_type

Initial sql query for group by patient and testType: SELECT count(\*) AS CNT, p.first\_name, p.last\_name, rr.patient\_id,rr.test\_type FROM persons p,pacs\_images pi,radiology\_record rr WHERE p.person\_id=rr.patient\_id AND rr.record\_id=pi.record\_id "+extra+" GROUP BY p.first\_name, p.last\_name, rr.patient\_id, rr.test\_type

Initial sql query for group by year: SELECT count(\*) AS CNT,EXTRACT(YEAR FROM rr.test\_date) AS Y FROM pacs\_images pi, radiology\_record rr WHERE rr.record\_id=pi.record\_id "+extra+" GROUP BY EXTRACT(YEAR FROM rr.test\_date)

Initial sql query for group by month: SELECT count(\*) AS CNT,EXTRACT(YEAR FROM rr.test\_date) AS Y,EXTRACT(MONTH FROM rr.test\_date) AS M FROM pacs\_images pi,radiology\_record rr WHERE rr.record\_id=pi.record\_id "+extra+" GROUP BY EXTRACT(YEAR FROM rr.test\_date), EXTRACT(MONTH FROM rr.test\_date)

Initial sql query for group by week: SELECT count(\*) AS CNT,EXTRACT(YEAR FROM rr.test\_date) AS Y,EXTRACT(MONTH FROM rr.test\_date) AS M,to\_char(rr.test\_date,'w') AS W FROM pacs\_images pi, radiology\_record rr WHERE rr.record\_id=pi.record\_id "+extra+" GROUP BY EXTRACT(YEAR FROM rr.test\_date), EXTRACT(MONTH FROM rr.test\_date), to\_char(rr.test\_date, 'w')

Initial sql query for group by year and patient: SELECT count(\*) AS CNT,EXTRACT(YEAR FROM rr.test\_date) AS Y,p.first\_name,p.last\_name,p.person\_id FROM persons p,radiology\_record rr,pacs\_images pi WHERE rr.record\_id=pi.record\_id AND rr.patient\_id=p.person\_id "+extra+" GROUP BY EXTRACT(YEAR FROM rr.test\_date),p.first\_name,p.last\_name,p.person\_id

Note that initial sql query for group by month and patient and week and patient are omitted.

Initial sql query for group by time and testType: SELECT count(\*) AS CNT,EXTRACT(YEAR FROM rr.test\_date) AS Y,rr.test\_type FROM pacs\_images pi,radiology\_record rr WHERE rr.record\_id = pi.record\_id "+extra+" GROUP BY EXTRACT(YEAR FROM rr.test\_date),rr.test\_type

Note that initial sql query for group by month and testType and week and testType are omitted.

Initial sql query for group by all options: SELECT count(\*) AS CNT,EXTRACT(YEAR FROM rr.test\_date) AS Y,p.first\_name,p.last\_name,p.person\_id,rr.test\_type FROM persons p,radiology\_record rr,pacs\_images pi WHERE rr.record\_id=pi.record\_id AND rr.patient\_id=p.person\_id "+extra+" GROUP BY EXTRACT(YEAR FROM rr.test\_date),p.first\_name,p.last\_name,p.person\_id,rr.test\_type

Note that initial sql query for group all options for month and week are omitted.

If specified, Patient id, Test Type, From date, and To data will be inserted into the initial sql query at the position labeled as "+extra+" above. A sample with all conditions specified look like this: AND rr.patient\_id="patientID" AND rr.test\_type="testType" AND rr.test\_date>="fDate" AND rr.test\_date<="tDate"

After the query is executed, the returned count will be displayed.