Jordan Letourneau
Jake Brand
Andrew Charles

CMPUT 391 Project Report

* Folders and Contained Classes with Method Descriptions

*** CSS**

*** datepicker.cs**

>Formatting file for the datepicker we used in many of the forms in the website.

*** dataAnalysis**

*** dataAnalysis.jsp**

>This is the query page that the user fills out to generate the report. He or she has the option of querying based on Patient name or test time and by time.

*** OLAPreport.jsp**

>In data analysis we decided to use the rollup function in sql to provide an OLAP report that give the administrator some useful information. There is six queries used in total based on the administrators selection. If the user selects to view the report based on Patient name and year this query is used.

```
SELECT PATIENT_NAME, TEST_TYPE, count(TEST_DATE),
                     extract (year from test_date) " +
                      "FROM RADIOLOGY_RECORD " +
                      "GROUP BY rollup " +
                      "(PATIENT_NAME, TEST_TYPE, extract (year from test_date))"
```

This query is used for month:

```
SELECT PATIENT_NAME, TEST_TYPE, count(TEST_DATE),
                     extract (month from test_date) " +
                      "FROM RADIOLOGY_RECORD " +
                      "GROUP BY rollup " +
                      "(PATIENT_NAME, TEST_TYPE,
                     extract (month from test_date))"
```

This query is used for week:

```
SELECT PATIENT_NAME, TEST_TYPE, count(TEST_DATE),
                     TO_CHAR((TEST_DATE),'WW') " +
                      "FROM RADIOLOGY_RECORD " +
                      "GROUP BY rollup " +
                     "(PATIENT_NAME, TEST_TYPE,
                     to_char((TEST_DATE),'WW'))"
```

If the user selects based on Test type and year this query is used.

```
        "SELECT PATIENT_NAME, TEST_TYPE, count(TEST_DATE),
                                       extract (YEAR from test_date) " +
                                        "FROM RADIOLOGY_RECORD " +
                                        "WHERE patient_name =" + "'" + patientName + "'" +
                                        " GROUP BY rollup " +
                                        "(PATIENT_NAME, TEST_TYPE,
                                        extract (YEAR from test_date))"
```

This query is used for Month:

```
        SELECT PATIENT_NAME, TEST_TYPE, count(TEST_DATE),
                                       extract (Month from test_date) " +
                                        "FROM RADIOLOGY_RECORD " +
                                        "WHERE patient_name =" + "'" + patientName + "'" +
                                        " GROUP BY rollup " +
                                        "(PATIENT_NAME, TEST_TYPE,
                                       extract (Month from test_date))"
```

This query is used for week:

```
        SELECT PATIENT_NAME, TEST_TYPE, count(TEST_DATE),
                                       TO_CHAR((TEST_DATE),'WW') " +
                                        "FROM RADIOLOGY_RECORD " +
                                        "WHERE patient_name =" + "'" + patientName + "'" +
                                        "GROUP BY rollup " +
                                       "(PATIENT_NAME, TEST_TYPE,
                                       to_char((TEST_DATE),'WW'))"
```

This information that is stored in the result set is then printed into a table format that is easy to read and understand. We decided to only show this information because it give the administrator useful information about the system and patients like how many of each test the patient has had at any time interval. This could provide useful to protect against harmful radiation exposure and as a complementary system to the search module.

*** header**

*** header.jsp**

>Used to provide buttons and to the other pages of the site. It has logic built in that will only show the buttons to the people that are allowed to access the pages. For example since the administrator is not restricted access to any module they can see every button, but a patient cannot.

*** login**

*** loggedin.jsp**

>Checks the user password against the password that is stored in the database. If the password is incorrect the user will be notified and returned to the login screen. If successful the user is redirected to the home page and the cookies will be stored with his or her information for further use.

*** login.jsp**

>Contains the form that is used to query the user for his or her user name and password. It also will display whether the password is wrong based on a cookie set by loggedin.jsp.

*** logout.jsp**

>Deletes all cookies and sessions that were created while the user was logged in, this will block access to the pages that were previously allowed by the previous user. Also all the information of the previous user is deleted for safety.

*** updateUserInfo.jsp**

>In user settings we decided that the user must first be logged in to change thier information. We also decided for more security the user must re-enter their password to be able to change any of their information. Once the submit button is pushed the code will check if the password the user provided is the same as the one in the database, if not then nothing will be updated. It then checks if the user is in the persons table and that their personal information is actually stored. If it is not then it will be inserted using this query:

```
sql = "insert into persons values(\'" + userName + "\', " +
        "\'" + updateFirstName + "\', " +
        "\'" + updateLastName + "\', " +
        "\'" + updateAddress + "\', " +
        "\'" + updateEmail + "\', " +
        "'" + updatePhone + "\')";
```

If the user is in the system already then the code will run a query to update their information.

```
    UpdateInfo = conn.prepareStatement("UPDATE PERSONS SET
FIRST_NAME= ?," +  "LAST_NAME= ?, ADDRESS= ?, EMAIL= ?, PHONE= ?
WHERE USER_NAME = ?");
```

Then finally the password input feilds are checked. If both the fields match and are not empty the password is updated in the users tabel using this query.

"UPDATE USERS SET PASSWORD= ? WHERE USER_NAME = ?"

*** usersettings.jsp**

>Constains the form that is presented to the user so they can update their information, when the submit button is pressed all the information is sent to updateUserInfo.jsp.

*** reportGenerating**

*** reportGenerating.jsp**

> Prepares the page with the title and dropdown menus for each required piece of information to generate a report.

The drop down menus prompt the user to select a diagnosis, starting date and ending date for the report to be generated. Note that the diagnosis dropdown allows you to select any known diagnosis, the start date dropdown allows you to select any time that a patient has had a prescription given, and the end date dropdown allows you to select any time that a patient has had a prescription given. This will allow you to search for all patients with a given diagnosis between the start date, and the end date.

Once input has been selected and the button pressed reportGenerated is called.

SQL STATEMENTS:

SELECT unique diagnosis FROM radiology_record order by upper(diagnosis)
SELECT unique prescribing_date FROM radiology_record order by prescribing_date

*** reportGenerated.jsp**

> Receives the parameters from reportGenerating, queries the database, and presents all valid results to the user in a clear and simple table.

SQL STATEMENTS:

SELECT first_name, last_name, address, phone, prescribing_date FROM persons p, radiology_record r WHERE r.diagnosis = ‘selectedDiagnosis’ AND r.prescribing_date BETWEEN ‘selectedStartDate’ AND ‘selectedEndDate' AND p.first_name = r.patient_name

*** search**

*** search.jsp**

>Implements a search on all of the records in the database and returns the result in a table that is sortable. The search algorithm will create queries based on the number of words inserted. for every word there is three queries that will return a rank. This rank is then used to calculated a score that is used to order the results from most useful to the least. The algorithm used is better than the one used in the example because it will search for each word separately instead of just a string of words, this brings up more relevant results. The queries used to do the search were as follows:

```
        String sql1 = "select record_id, patient_name, diagnosis, description, test_date,";
        String sql2 = " score(1), score(2), score(3)";
        String sql3 = " from radiology_record";
        String sql4 = " where contains(patient_name, '%" + terms[0] + "%', 1) > 0 or" +
                " contains(diagnosis, '%" + terms[0] + "%', 2) > 0 or" +
                " contains(description, '%" + terms[0] + "%', 3) > 0";
        for (int i = 1; i < terms.length; i++)
        {
                int first = (i * 3) + 1;
                int secon = (i * 3) + 2;
                int third = (i * 3) + 3;
                sql2 += ", score(" + Integer.toString (first) +
                                "), score(" + Integer.toString (secon) +
                                "), score(" + Integer.toString (third) + ")";
                sql4 += " or" +
                " contains(patient_name, '%" + terms[i] + "%', " + Integer.toString (first) + ") > 0
or" +
                " contains(diagnosis, '%" + terms[i] + "%', " + Integer.toString (secon) + ") > 0
or" +
                " contains(description, '%" + terms[i] + "%', " + Integer.toString (third) + ") > 0";
        }
        String sql = sql1 + sql2 + sql3 + sql4;
        if (className.equals ("p"))
                sql += " and patient_name='" + user + "'";
        else if (className.equals ("d"))
                sql += " and doctor_name='" + user + "'";
        else if (className.equals ("r"))
                sql += " and radiologist_name='" + user + "'";
```

This query is used to find the rank which is used to calculate the score which is in turn is used to order the results by relevance.

*** uploading**

*** uploading.jsp**

> Prepares the page with all required dropdown menus and text fields to upload a new radiology record. If all text fields have been filled correctly and the test date is not prior to the prescription date commitUpload.jsp is called.

SQL STATEMENTS:

SELECT user_name, class FROM users u WHERE u.class ='r' OR u.class='d' OR u.class='p'

*** commitUpload.jsp**

> Receives the parameters from uploading and commits the radiology record to the database (as requirements specify). It then confirms the users commit and prepares the page to upload a file for this radiology record. Once a file has been located and the upload button is pressed upload.jsp is called.

SQL STATEMENT

```
SELECT MAX(record_id) FROM radiology_record
```

*** upload.jsp**

> Receives the parameters from commitUpload.jsp and if the recieved file type is correct uploads it as a BLOB to the database for future querying.

SQL STATEMENT

```
SELECT MAX(image_id) FROM pacs_images
INSERT INTO pacs_images VALUES ('" + record_id + "', '" + image_id + "', ?, ?, )
        preparedStatement.setBinaryStream (1, (InputStream) fis, (int) (image.length ()));
        preparedStatement.setBinaryStream (2, (InputStream) fis, (int) (image.length ()));
        preparedStatement.setBinaryStream (3, (InputStream) fis, (int) (image.length ()));
INSERT INTO radiology_record (record_id, patient_name, doctor_name, radiologist_name,
test_type, prescribing_date, test_date, diagnosis, description) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
        pst.setInt(1, recordId);
        pst.setString(2, patientName);
        pst.setString(3, doctorName);
        pst.setString(4, radiologistName);
        pst.setString(5, testType);
        pst.setString(6, prescribingDate);
        pst.setString(7, testDate);
        pst.setString(8, diagnosis);
        pst.setString(9, description);
```

*** userManagement**

*** userManagement.jsp**

>Used to update or insert information on a specified user of the RIS system. This user is an administrator. He or she can insert a new user into the system by input a user name and password into the fields provided. To insert the new user this query is used.

```
insert into users values ('" + newUser + "', '" + pass + "', '" + uClass + "', '" + date)
```

Once inserted the user can then select the new user in the drop down menu and press select. This will reload the page with a new form that asks to input personal information. Once the information is filled the user will press submit. Since the user is a new user this query is used to insert the person into the person table and to insert the family doctor into the family doctor table.

```
insert into family_doctor values('" + Doctor + "', " +
                               "'" + user + "')

insert into persons values(\'" + user + "\', " +
                               "\'" + first + "\', " +
                               "\'" + last + "\', " +
                               "\'" + address + "\', " +
                               "\'" + email + "\', " +
                               "'" + phone + "\')
```

If the user happens to already be in the persons table the information will be updated using these queries.

```
update family_doctor set doctor_name='" + Doctor + "' " +
                                         "where patient_name='" + user

update persons set first_name=\'" + first + "\', " +
                               "last_name=\'" + last + "\', " +
                               "address=\'" + address + "\', " +
                               "email=\'" + email + "\', " +
                               "phone=\'" + phone + "\' " +
                               "where user_name=\'" + user

update users set password=\'" + pass + "\', " +
                               "class=\'" + uClass + "\', " +
                               "date_registered=\'" + formattedDate + "\' " +
                               "where user_name=\'" + user
```

To check if the person exists in the tables simple queries are used and then the result set is checked if it is empty or not, if it is empty then insert and if not update.

*** WEB-INF/classes/connect**

*** CheckClass.java**

>Used to validate the user upon accessing a page. This class is called and returns a boolean whether or not the user is able to access the page. The user_class and moduleName is passed as a parameter to the checkClass method in this class.

***connect.java**

>connectDb() used to connect to the database and reduce the amount of code that is needed in the other modules. The method setLogin(CCID, PASS) is used to set the CCID and PASS that will be used to access the database. The getDateStringFromDateString (String date)

method is used to convert a date string into a format that can be stored into the database.

*** WEB-INF/lib**

- **CheckClass.jar**
  >compiled jar file of CheckClass.java class.
- **commons-fileupload-1.2.2.jar**
  >compiled jar used in the search module, provided by the instructor.
- **commons-io-2.2.jar**
  >compiled jar used in the search module, provided by the instructor.
- **connect.jar**
  >compiled jar file of the connect.java class.