University of Alberta

Term Project

RaySyS

Steven Maschmeyer
Camden Narzt
Oscar Ramirez

Professor:
Dr. Li-Yan Yuan

April 4, 2011

RaySys

RaySys consists of several modules that allow different types of users to manage and interact a Radiological Database System. With this application users are divided into four different categories: administrators, patients, doctors, and radiologists each having different privileges. In order to summarize the system architecture and the major modules used in the system, along with their relationships and main interactions with the database this document will first describe those sections that are common to all users, and then it will be divided into specific user sections. Each section will contain a listing of the relevant **.jsp* and **.java* files for the module, along with the initial *URL* which can be used to access it.

Login

The login module is the first place the user will interact with RaySys. The user will first encounter this when accessing the system for the first time. At this point a login screen requesting a `username` and a `password` is shown. If the user enters information for a valid account a *session variable* `userName` will be set be a String representing the user's account type. Also as expected if the information is invalid the user will be notified appropriately. For added security sessions are set to expire after 5 minutes, therefore if the user is inactive for a period of time any action will redirect the user to the login screen.

The action of logging in utilizes a the servlet `LoginManager.java` which is located in the `loginModule` package. Through this server a prepared statement is created with the form `select password, class from users where user_name = ?` and the name is set to equal the username provided. The result set that is generated from executing this query is then inspected and if a password was returned it is compared to the provided one by the user.

URL: `/radiologydb/login.jsp`
**.jsp:* `login.jsp`
**.java:* `LoginManager.java`

Profile

Another part of the login module is the profile. This provides an interface for the user to update most of his information. All users are allowed to change: *first name, last name, password, address, email,* and *phone*. This section of the login module takes advantage of a servlet that is also used for updating user information by admins so the specifications of the servlet used can be found in the *User Update* section.

URL: `/radiologydb/profile.jsp`
**.jsp:* `profile.jsp`
**.java:* `UserUpdate.java`
Note: Further information found in *User Update* section

Connections

Connections to the database are managed by `ConnectionManager`, which provide a static methods for opening pooled connections from an `OracleConnectionPoolDataSource`. (Note that since the server loads outdated JDBC drivers from `classes12.jar`, the connection cache was implemented using the deprecated class `OracleConnectionCacheImpl` instead of using the implicit connection caching mechanism provided by `OracleConnectionCacheManag`

`ConnectionManager` also provides a nested class `ConnectionKit` to help ease the execution of queries, and to ensure that resources are properly closed.

**.java:* `ConnectionManager.java`

Search Records

The home page of RaySys is the Search Records page, which allows users to find records over the columns *patient_name*, *diagnosis*, and or *description* defined in the *radiology_record* table. The user may specify start or end dates to limit the range of the search. By default, the results are returned by decreasing rank, where the rank of a given result record is defined as *6*frequency(patient_name) + 3*frequency(diagnosis) + frequency(description)*. Alternatively, the user can use the 'Order By' combo box to order the results by increasing or decreasing *test_date*. The resulting records are displayed in a table, with the list of thumbnails dispalyed in a `ContentFlow` (see 3rd party libraries). The user can scroll through the images in the `ContentFlow` and then click on one to see the full size image in a separate browser. Record search is available to all users, however the results are filtered depending on the user class (defined in the session).

Records searches are initiated by */search.jsp* which creates a new `RecordsQuery` to get a connection and carry out the actual query. *search.jsp* passes in relevant parameters – such as the search input – and `RecordsQuery` wraps the resulting `ResultSet` to reduce the amount of back-end work done by the *jsp* page. The exact query that is executed depends on the arguments that are passed to `RecordsQuery`. In particualar, the query will change slightly depending on how the user chooses to order the results, or if the user has selected to limit the results to a particular range of dates.

The resulting query has the following form, where square brackets denote segments that change depending on user input:

```
select (6*score(1) + 3*score(2) + score(3)) as rank, record_id,
patient_name,doctor_name, radiologist_name, test_type,
prescribing_date, test_date, diagnosis, description

from radiology_record where (contains(patient_name, [KEYWORDS], 1) > 0 or
contains(diagnosis, [KEYWORDS], 2) > 0 or contains(description, [KEYWORDS],
3) > 0)
```

$if user class is patient$ `and patient_name = [USERNAME]`

$if user class is radiologist$ `and radiologist_name = [USERNAME]`

$if user class is doctor$ `AND (doctor_name = [USER_NAME] OR patient_name in (select patient_name from family_doctor where doctor_name = [USERNAME]))`

```
 [and test_date >= to_date([STARTDATE], 'dd/MM/yyyy')] [and test_date <=
```

`to_date([ENDDATE], 'dd/MM/yyyy')] [order by test_date asc] [order by test_date desc] [order by myscore desc]`

The query is executed in a prepared statement to help protect against SQL injection.

`RecordsQuery` creates a PicsQuery for each record to fetch the image URLs for that record. The image URLs are generated by the `GetOnePic` servlet as demonstrated in the example files on the project page. The exact query executed for each record, image, and style is:

`select [STYLE] from pacs_images where record_id = [RECORDID] and image_id = [IMAGEID].`

> *URL:* `/radiologydb/index.jsp`
> **.jsp:* `index.jsp`

Adding Users

Adding users is only available to admins and the interface is very simple. Admins can add users by only setting the *username, password,* and *class*. This will add an entry to the `users` table in the database by utilizing the prepared statement `insert into users values (?,?,?,?)` where the values correspond to the ones listed before hand, and the fourth value is set to the registration time which is generated when the user is added.

If the administrator decides to fill the remaining fields *first name, last name, address, email, phone* an entry will be created in the persons entry with the supplied information.

> *URL:* `/radiologydb/new-uesr.jsp`
> **.jsp:* `new-user.jsp`
> **.java:* `NewUser.java`

User Update

The user update module's main purpose is to be utilized by administrators. This module provides an interface for administrators to update all information for any user type. The interface for this begins by providing the administrator a search for users where he can search by username, name, or last name and can then select what user to edit from the displayed results.

The interface for changing basic information is very similar to the one used to add users so no explanation should be needed. If the user to be modified is a patient or a doctor this module will display a list of all the doctors or patients available for choosing and a second list with the ones that are currently the patient/doctor's patients/doctors.

In order to edit user information two prepared statements, similar to those for adding new users, are used. The first prepared stament used is `update users set password=?, class = cast(? as char(1)) where user_name =?` and `update persons set first_ name =?, last_name=?, address=?, email=?, phone =? where user_name=?`.

In order to display the available doctors the statement `select UNIQUE patient_name from family_doctor where doctor_name = '" + username + "'` is used. In order to make a change to the *family_doctor* table we will either add a new row with `insert into family_doctor values('" + username + "', '" + uname + "')` or remove a row with `delete from family_doctor where doctor_name = '" + username + "' and patient_name = '" + uname + "'`. Very similar queries are used for getting, adding and deleting patient rows.

URL: `/radiologydb/update/index.jsp`
**.jsp:* `/update/index.jsp /index/user-search.jsp /index/update.jsp`
**.java:* `UserUpdate.java PersonSearch.java`

Report Generation

Report generation presents the user with the now very familiar search layout. This search is available to administrators and it allows them to search for a specific diagnosis in a time range. The results will then display the user information for those with the searched diagnosis along with the date of the first diagnosis for the searched type.

URL: `/radiologydb/report/index.jsp`
**.jsp:* `/report/index.jsp /report/search.jsp`
**.java:* `DiagnosisSearch.java`

Record Creation

The record creation is done in two parts, first a simple form that takes in all relevant data for a Radiology Record, then a servlet to send that data to oracle using the sql statements: texttt"SELECT rec_id_sequence.nextval from dual" and `"insert into radiology_record(record_ id,patient_name,doctor_name, radiologist_name , test_type,prescribing_date,test_date,diagnosis,description) values (?,?, ?, ?,?, ?, ?  ,?,?)"` where the '?"s are replaced with the relevant data using *setOBJTYPE()* style methods. The status of the creation is flashed to the screen after the user submits and upon success the page forwards to an image upload page with the record id already filled in, and upon failure return the user to the record create page, data intact to try and fix the data or figure out their error.

**.jsp:* `/radiologydb/create-record.jsp`
**.java:* `CreateRecord.java`

Image Upload

The image uploading is done in two parts the image path on the users computer and record id are chosen on one page, and then upon submit the data is passed to a servlet and the image is read in and resized to the three relevant sizes, then stored in the database using the SQL statements:

```
ResultSet rset1 = stmt.executeQuery("SELECT pic_id_sequence.nextval
from dual");
```

```
stmt.execute( "insert into pacs_images (record_id,image_id,thumbnail,
regular_size, full_size) values ("+record_id+","+pic_id+",
empty_blob(),empty_blob(),empty_blob())");
```

again if the upload is successful the page forwards you to the upload page with a success message, otherwise it flashes an error message and returns you to the upload page to try and fix the data.

Further Work

The current implementation of RaySys is a great starting point for any type of medical database system where multiple types of user must be supported. In order to improve upon this implementation several key points would have to be polished. The first thing that would need some work would be to standardize the use of ConnectionKit amongst all modules. This would provide a very cohesive and centralized interface for managing the connections and would provide a very easy way to upgrade when the database drivers are updated.

Another element that could benefit the user greatly would be support for multiple image uploads. This would simplify and speed up the process of inserting long records into the database. By handling multiple image uploads at the same time several optimizations could be made to make the process faster.

Acknowledgements

Calendar Popup http://www.mattkruse.com/javascript/calendarpopup/index.html

ContentFlow http://www.jacksasylum.eu/ContentFlow/

Selection Box http://www.web-savant.com/users/kathi/asp