Introduction

Our online radiology system allows users to interact with radiology records online. Users can be classified as administrators, patients, radiologists, or doctors, which determine the access rights of each user class to the databases. Our system is hosted on UofA servers at consort.cs.ualberta.ca, and the databases are hosted on UofA Oracle servers. The system was built using a combination of HTML, CSS, PHP, and JavaScript (more specifically the jQuery library). Notably, our system allows for asynchronous updating of webpages, so that web pages can dynamically update to new data.

General Classes/Files

stylesheets/generalstylesheet.css
All web pages use this CSS file for style

index.php

- Opens /login/loginform.php web page if Login button is clicked

index.php is the first file to be called when starting the client at consort.cs.ualberta.ca/~<ccid>. This page has a button that redirects the user to the login form (/login/loginform.php).

Database Connection files (/database directory)

These files are used to connect to the UofA Oracle Database. Any php file that contains a SQL statement requires these files

dbconnect.php
This file is used to connect to the database using the administrator's Oracle credentials.
executecommand.php

- Holds the function executeCommand($conn, $sql)

This function takes a formatted SQL statement and an Oracle connection as arguments, and executes the statement. This particular function returns a single result from the query in a two-dimensional array. Indices are not based on column names but rather numerical.
executecommand2.php

- Holds the function executeCommand2($conn, $sql)

Returns all results of an SQL statement in a two-dimensional array (results in array[0])
gettableid.php

- Holds function getTableId($conn, $parameter) to return id of next item in table

Login Module (/login directory)

The Login module handles user logins, their privileges, as well as updates to a user's email and password. The main html files are found in /login, while SQL queries are handled in files found in /login/php.

<u>loginform.php</u>

- Opened from index.php
- Also opened if back button is pressed while in another module
- References /login/getUserData.php to retrieve user info for existing sessions
- References /login/login.php to retrieve user information for new sessions
- References /login/logout.php if Logout button is pressed
- Contains buttons to open different modules depending on access rights
- Always shows ‘Update User Email’, ‘Update User Password’ and ‘Search Records’ buttons for all users

loginform.php is the next step after clicking the Login button while on the index.php web page, or going back from another module. If a user has not logged in yet, this module displays forms to enter username and password. If a user has previously logged in, then this module will display buttons to all of the user-specific modules listed in the Permissions section above. Clicking one of the module buttons will open up the corresponding web page, usually called <module-name>form.php.

This file has JavaScript embedded into the html. In particular, AJAX in the jQuery library is leveraged to allow web pages to be updated asynchronously. The loginform.php file loads the html elements first (module buttons or user/password forms depending on session status).

Web page loading sequence:
1. If user is not in a session, username/password forms are created using html.
2. User enters login details and starts a new session.
3. Embedded JavaScript sends an AJAX request to server and retrieves user information
4. JavaScript code clears the username/password forms and displays correct buttons to open modules.

1. If user is logged in previously, buttons are created using html code
2. Embedded JavaScript code does not get used in this case

<u>login/getUserData.php</u>

- Holds the function getUserData($username)

This function is called by loginform.php to retrieve username, password, class, register date, and person_id from the databases if a user is resuming a session. No error checking or password checking is needed because they were checked on log in:

SELECT user_name, password, class, date_registered, person_id FROM users WHERE user_name = '$username'

<u>login/login.php</u>

- Used to query Oracle server for user information asynchronously

This server-side php file is called by loginform.php when a user tries to log in for the first time. Login.php first checks for errors in the entry fields. If no error is found, then the user database is queried, and the results are sent back to the client in JSON format to be parsed back into an array and displayed:

SELECT u.user_name,u.class FROM users u WHERE u.user_name = ‘$username’ AND u.password = '$password';

<u>/login/logout.php</u>

This php file is called when the Logout button is pressed on the loginform.php web page. This file ends the current session and refreshes the loginform.php web page to show the username/password forms.

<u>/login/updateuseremailform.php</u>

- Opens when ‘Update User Email’ button is clicked on loginform.php web page
- References /login/php/updateemail.php file when updating users table

This web page is displayed so that users can update their email to match the email of an entry in the persons table. When a new email is entered into the form, The embedded jQuery function makes an AJAX HTTP request to query the database (which is handled in /login/php/updateemail.php). This web page will display errors if no email is entered, if the email is too long, or if the email does not exist. Success messages are also displayed.

<u>/login/php/updateemail.php</u>

- Contains function checkEmailExists($conn, $email, $errorcode)

This function queries the Oracle database to check if a certain email exists in the persons table. An error code is returned to updateuseremailform.php if a person with a matching email does not exist. A success message is returned if the database is correctly updated:

SELECT COUNT(*) FROM persons p WHERE p.email ='$email'

- Contains function getPersonId($conn, $email)

This function queries the database to get the person_id of the matching email:

SELECT p.person_id FROM persons p WHERE p.email ='$email'

Overall, this php file is used when a user wishes to update their email. For a user to update their email, a person with the desired email must already exist in the persons table. An error or success code is returned to updateuseremailform.php to notify the user if the update was successful, or failed for any reason.

<u>/login/updateuserpasswordform.php and /login/php/updateuserpassword.php</u>

- Opened if user clicks on Update Password button on loginform.php web page

This web page is similar to updateuseremailform.php except it updates password, and only requires one database query:

```
UPDATE users SET password = '$password' WHERE user_name = '$_SESSION['user_name']'
```

Report Generating Module

<u>/generatereport/generatereportform.php</u>

- Opened if a logged-in administrator clicks on Generate Reports button
- References /generatereport/php/searchrecords.php

This file handles the report generating module to display data about records matching the specified diagnoses. This file follows the same format as other non-login modules. PHP code is used to check if the user is logged in and authorized to access this page. Users cannot go straight to this module by entering the exact url if they are not logged in. If the user is an administrator (and thus authorized to generate reports), then the html code is run, and creates a form to search for a diagnosis. When the administrator submits the search, an AJAX request is sent by the embedded jQuery code to search for matching records/patient, and update the web page to display the results. Like the other modules, SQL processing is done in a separate file (for this case /generatereport/php/searchrecords.php).

<u>/generatereport/php/searchrecords.php</u>

- Contains function findRecords($conn, $diagnosis, $errorcode)
- Requires processfield.php, checkfieldlength.php, and checkfieldempty.php, all of which are in /usermanagement/php
- Requires /database/executecommand2.php to return multiple results

This file takes the specified diagnosis from generatereportform.php and performs an SQL query to return results for the report. Error checking is done by referencing user management files to process the search term, restrict the length of the search term, and check for an empty diagnosis. If there are no errors, the SQL statement is executed and all results are encoded into JSON and sent to the client (generatereportform.php) to be parsed back into an array and displayed. Diagnoses of the results do not have to be an exact match to the search term since administrators may want to see all related diagnoses:

```
SELECT p.first_name, p.address, p.phone, r.test_date FROM persons p, radiology_record r
WHERE r.diagnosis LIKE ‘%$diagnosis%’ AND p.person_id  = r.patient_id
```

Search Module

/search/searchform.php

- Opened if any user clicks on the search button
- Contains function loadrecord(scr) to load an image into a new window when it is clicked
- References /search/php/searchradiologyrecord.php if user wishes to search the records

This module is laid out the same as generatereportform.php, with a few different conditions. The PHP code checks if the user is already logged in before the HTML code is run. The HTML displays forms to enter keywords and/or dates. If a user submits a search, then the embedded JavaScript code sends an AJAX request to perform the SQL query (done by /search/php/searchradiologyrecord.php) and update the web page with the results. Selecting an image calls the loadrecord(scr) function to open the image in a new window and zoom.

/search/php/searchradiologyrecord.php

- Requires dbconnect.php, executecommand.php, and executecommand2.php in the /database directory
- Contains function getUserData($conn, $username) to get a specified user's data (for access control)
- Contains function getRecordImages($conn, $r_id) to retrieve the images for a specified radiology record

This file handles the processing of record searches. Since a user can leave a search term blank, the SQL statement must be customized depending on which search fields are entered (keywords and date). This forms an SQL statement that queries all records that match the keyword and/or date. Next, the user's permissions are queried to restrict the results to that user's profession (eg. doctors can only see the records of their patients, and radiologists only see the tests they've performed). Next, the SQL statement is modified to change the order of the results based on what the user specifies (eg. recent first, or group by ranking algorithm). Once the SQL statement is properly constructed, it is executed, and the results are passed back to the web page encoded in JSON.

Base SQL Statement (all queries start with this statement):

```
SELECT r2.record_id, p.first_name, p.last_name, d.first_name, d.last_name, r.first_name,
r.last_name, r2.test_type, r2.prescribing_date, r2.test_date, r2.diagnosis, r2.description,
p2.full_size
FROM radiology_record r2, persons p, persons d, persons r, pacs_images p2
WHERE r2.patient_id = p.person_id
AND r2.doctor_id = d.person_id AND r2.radiologist_id = r.person_id AND p2.record_id =
r2.record_id AND p2.image_id = 1
```

Conditions are concatenated to this statement based on permissions and search terms:

- If a keyword is specified

```
AND((contains(p.first_name, '$keywords', '.'1'. ') > 0)
OR (contains(p.last_name, '$keywords, '.'2'. ') > 0)
```

OR (contains(r2.diagnosis, '$keywords', '.'3'. ') > 0)
OR (contains(r2.description,'$keywords','.'4'. ') > 0));

- If a date is specified

AND (r2.test_date BETWEEN TO_DATE('$s_date', 'YYYY-MM-DD') AND TO_DATE('$f_date', 'YYYY-MM-DD'))

- Depending on permissions

AND r2.patient_id = $person_id --if user is a patient
AND r2.doctor_id = $person_id --if user is a doctor
AND r2.radiologist_id = $person_id --if user is a radiologist

- depending on ranking type

ORDER BY r2.test_date ASC -- if user specifies most recent last
ORDER BY (RANK() OVER (ORDER BY(6*(SCORE(1)+SCORE(2)) + 3*SCORE(3) + SCORE(4)))) DESC -- if user doesn`t choose a specific order

Uploading Module

/uploading/insertradiologyrecordform.php

- Opens if a radiologist clicks Insert Radiology Record button on loginform.php page
- References /uploading/php/insertradiologyrecord.php if an image is submitted

This file is organized the same way as the other user-specific modules. PHP code checks if the user is signed in as a radiologist, then displays HTML elements to fill in the record details, as well as upload an image. When the radiologist finishes filling in the fields and submits, the embedded JavaScript code sends an AJAX request to update the tables (uses /uploading/php/insertradiologyrecord.php) and receive a success message.

/uploading/php/insertradiologyrecord.php

- References /usermanagement/php/ files to process the entries and error check
- References database files

This file processes the inputs and checks for any errors before performing an insert to radiology_record and pacs_images.

SQL Statement for radiology_record insertion:
INSERT INTO radiology_record (record_id, patient_id, doctor_id, radiologist_id, test_type, prescribing_date, test_date, diagnosis, description) VALUES ('$record_id','$id2','$id','$person_id.','$test_type',TO_DATE('$p_date','YYYY-MM-DD'),TO_DATE ('.$test_date.','YYYY-MM-DD'),'$diagnosis','$desc')

SQL Statement for upload Image function:
insert into pacs_images (record_id, image_id, thumbnail, regular_size, full_size) values(:recordid, :imageid, empty_blob(), empty_blob(), empty_blob()) returning thumbnail, regular_size, full_size into :thumbnail, :regularsize, :fullsize

User Management Module
All of these modules are very similar. The user is checked to be signed in as an administrator in the PHP code, and HTML forms are displayed for the inputs. The insert<row>form.php files have forms for all of the attributes, and the row is inserted by the corresponding file (/usermanagement/php/insert<row>.php). The update<row>form.php files follow a similar process, but have an additional, originally hidden, form. The first form is used to search for matching results, and the second form is used to update the matched row. Each update web page references two files for the SQL query (search<row>.php and update<row>.php. All SQL queries are performed asynchronously in an AJAX request.

/usermanagement/insertuserform.php
- References /usermangement/php/insertuser.php

/usermanagement/php/insertuser.php
- References processfield.php, checkfieldlength.php, and checkfieldempty.php in /usermanagement/php/ for error checking

Checks for errors in any of the fields and performs the insertion

SQL Statement:
INSERT INTO users (user_name, password, class, person_id, date_registered) VALUES (\''.$username.'\',\''.$password.'\',\''.$class.'\',\''.$id.'\',sysdate)

/usermangement/updateuserform.php
- References /usermanagement/php/searchuser.php and /php/updateuser.php

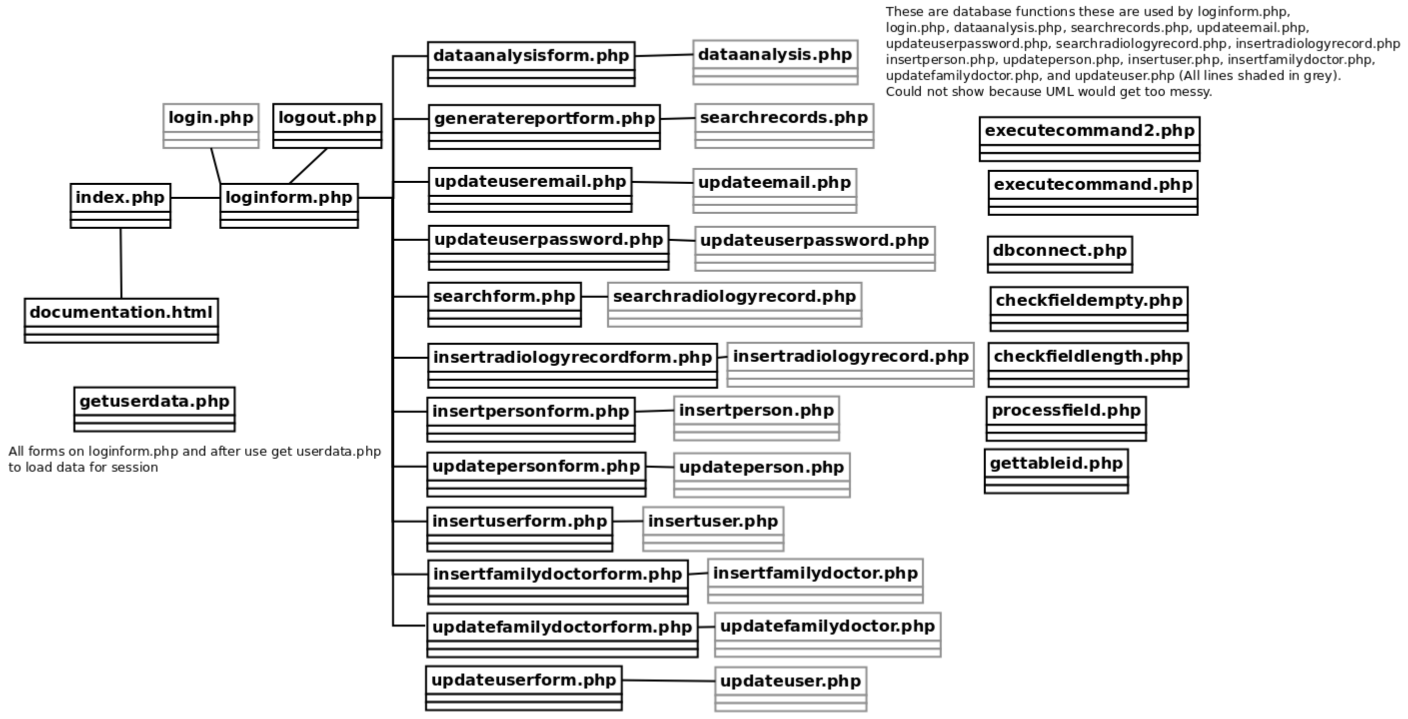
/usermanagement/php/searchuser.php
Checks for errors in any of the fields and queries database to find a matching user:
SELECT u.user_name, u.password, u.class, u.person_id, u.date_registered FROM users u WHERE u.user_name ='$username'
/usermanagement/php/updateuser.php
Takes inputs and updates user table:
UPDATE users SET class = '$class’ WHERE user_name = '$username'

Data Analysis Module
/dataanalysis/dataanalysisform.php
This web page checks if user is logged in as an administrator, then asks for patient id, test type, and/or time period. When the user clicks ‘Generate Report’, /php/dataanalysis.php is called to query the database

Sample SQL Statement with all inputs filled and time period set to week:
SELECT g.patient_id, g.test_type, to_char(g.test_date,'IW'), COUNT(g.image_id) FROM g_records g GROUP BY g.patient_id, g,test_type, to_char(g.test_date,'IW')

Diagrams

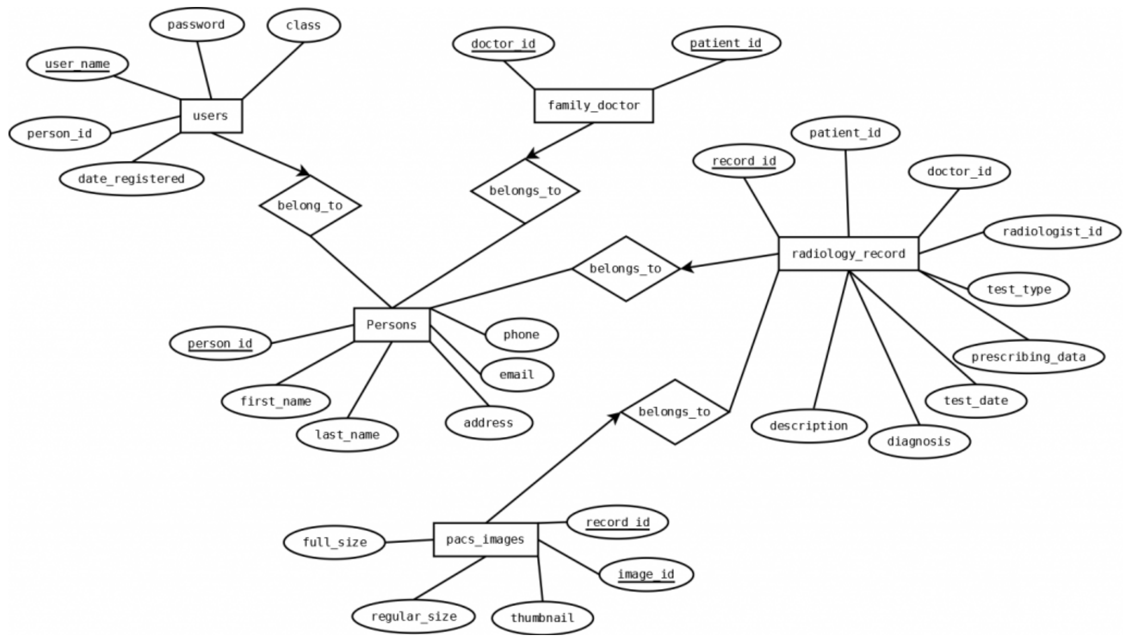
UML Diagram



These are database functions these are used by loginform.php, login.php, dataanalysis.php, searchrecords.php, updateemail.php, updateuserpassword.php, searchradiologyrecord.php, insertradiologyrecord.php insertperson.php, updateperson.php, insertuser.php, insertfamilydoctor.php, updatefamilydoctor.php, and updateuser.php (All lines shaded in grey). Could not show because UML would get too messy.

All forms on loginform.php and after use get userdata.php to load data for session

ER Diagram


password
class
user_name
users
person_id
date_registered
belong_to
doctor_id
patient_id
family_doctor
belongs_to
patient_id
record_id
doctor_id
radiologist_id
radiology_record
belongs_to
test_type
Persons
person_id
phone
email
first_name
last_name
address
prescribing_data
test_date
belongs_to
description
diagnosis
full_size
pacs_images
record_id
image_id
regular_size
thumbnail