Project Report

System Architecture

Login Module

session.php: Checks if the user is logged in. If they aren't logged in, it redirects the user to the login page. Otherwise, they are directed to the page they requested.

login.php: Displays the login page. When the submit button is pressed, it checks the user information against the database. If a matching record is found, their user_name, class, and person_id is stored in a session cookie. Otherwise, an error is displayed and the user may try again.

```
SELECT user_name, class, person_id FROM users WHERE user_name = :username AND
password = :password;
```

changepassword.php: Allows the user to change their password. They must enter their current password, then confirm their password by entering it twice. If the current password and username matches, the password is updated. Otherwise, an error message is displayed and the user may try again.

```
UPDATE users SET password = :newpass WHERE user_name = :username AND password =
:oldpass;
```

editinfo.php: Allows the user to edit their own information. Only the information from their record in the persons table is displayed and may be edited.

Information is fetched and updated according to the person_id recorded in their session cookie. If data is missing, the update will not be completed.

```
SELECT first_name, last_name, address, email, phone FROM persons WHERE
person_id = '". getUserPersonID() ."';

UPDATE persons SET first_name = :firstname, last_name = :lastname, address =
:address, email = :email, phone = :phone WHERE person_id = :person;
```

User Management Module

users.php: Allows an admin user to search for users. This query is created dynamically for each keyword. The requested query is split into at most 5 pieces and compared against each field using the IN operator. For ease of use, the terms and field values are set to uppercase to allow case-insensitive searching.

```
SELECT persons.person_id, user_name, class, date_registered, first_name,
last_name, address, email, phone FROM users, persons WHERE users.person_id =
persons.person_id AND ( FIELD1 IN (term1, term2 ...) OR FIELD2 IN (term1,
term2, ...) ... );
```

register.php: This page allows the admin user to add new users. The admin must fill in a username, password, first name, last name, address, email, and phone number for the user. When the query is submitted, the username and email is checked against the database for uniqueness.

```
SELECT count(*) AS in_use FROM users, persons WHERE users.person_id =
persons.person_id AND (user_name = :user_name OR email = :email);
```

The data is then inserted first into the persons table, then into the users table, rolling back on an error.

```
INSERT INTO persons(person_id, first_name, last_name, address, email, phone)
VALUES(person_id_seq.nextval, :first_name, :last_name, :address, :email,
:phone) RETURNING person_id INTO :person_id
```

```
INSERT INTO users(user_name, password, person_id, class, date_registered)
VALUES (:user_name, :password, :person_id, :class, sysdate)
```

edituser.php: This page allows an admin user to modify any user's information. On submission, all fields that are left blank are ignored. The update query is built up dynamically. First, all entries in the family_doctor table associated with the patient are deleted. Then, all doctor ids that were added in the edituser page are re-added to this table.

```
DELETE FROM family_doctor WHERE patient_id = :patient
INSERT INTO family_doctor(patient_id, doctor_id) VALUES (:patient, :doctor)
```

Second, all data in the person table is updated. If all person entries are blank, this query is skipped.

```
UPDATE persons SET field1 = value1, field2 = value2, ...
```

Finally, all data in the users table is updated. Again, if the data is blank, this query is skipped.

```
UPDATE users SET field1 = value1, field2 = value2, ...
```

Report Generation Module

report_request.php: report_request is linked to from the home page and accepts a diagnosis and a year and posts them both to report_generated.php. However if the user is not an admin an error message appears saying they do not have access to this page with a link back to the home page.

report_generated.php: Accepts the diagnosis and year from report_request.php, printing an error message if one of the values is missing, and generates a report of all the records that include both that year and that diagnosis.

```
SELECT p.first_name, p.last_name, p.address, p.phone, r.test_date FROM persons
p, radiology_record r WHERE p.person_id = r.patient_id AND CONTAINS(diagnosis,
'$diagnosis', 1) > 0 AND '$year' = (EXTRACT (YEAR FROM r.test_date));
```

Uploading Module

uploadRecord.php: This is the html form that allows the user to input all the necessary data for the creation of a record.

_upload.php: Gets the data that was input by the user to create a new radiology record. Record_id's are generated by a sequence that was created in the database on setup.

```
INSERT INTO radiology_record VALUES(record_id_seq.nextval, $patient_id,
$doctor_id, $radiologist_id,'$test_type', to_date('$prescribing_date',
'YYYY-MM-DD'), to_date('$test_date', 'YYYY-MM-DD'), '$diagnosis',
'$description')RETURNING record_id INTO :record_id
```

uploadImage.php: Gives the user the ability to search through their file system for an image to upload. On submit, uploadProcessor.php is called.

uploadProcessor.php: Ensures that the image file provided by the user is an actual image. Takes the image as the full sized image, scales the image to 150x150 for the thumbnail and 400x((400*height of the full sized image)/width of the full sized image) for the regular sized image. These three images are saved to the database by creating three empty blobs with:

```
INSERT INTO pacs_images (record_id, image_id, thumbnail, regular_size,
full_size) VALUES ($record_id, image_id_seq.nextval, EMPTY_BLOB(),
EMPTY_BLOB(), EMPTY_BLOB()) RETURNING thumbnail, regular_size, full_size,
image_id INTO :thumbnail, :regular, :image, :id;
```

and then binding the images to blob handles to save in the images into.   Once the images have been saved, the user is givens some information on the image and told that it was uploaded successfully. The image is displayed to the user through displayImage.php which fetches the images from the database and displays. There is a 'Home' link at the bottom of the page which redirects back to the home page.

Search Module

search.php: Displays a keyword search box and a date range search box for the user to enter their search queries.

_search.php: Gets the user’s search query and checks if it is a key word search or a date range search. Date ranges are searched for by:

```
SELECT i.image_id, r.record_id, r.patient_id, p.first_name, p.last_name,
r.doctor_id, r.radiologist_id, r.test_type, r.prescribing_date, r.test_date,
r.diagnosis, r.description    FROM family_doctor fd, radiology_record r,
persons p, pacs_images i WHERE r.patient_id = p.person_id AND r.record_id =
i.record_id AND ((prescribing_date BETWEEN to_date('$startDate', 'YYYY-MM-DD')
AND to_date('$endDate', 'YYYY-MM-DD')) OR (test_date BETWEEN
to_date('$startDate', 'YYYY-MM-DD') AND to_date('$endDate', 'YYYY-MM-DD')))
```

This gets the user all of the information about all records. Depending on the type of user one of:

```
AND p.person_id = $personId
AND fd.doctor_id = $personId AND fd.doctor_id = r.doctor_id
AND r.radiologist_id = $personId
```

will be added to the query. The first for patients, second for doctors and the third for radiologist. There is no special access restrictions for admin users. Date range search applies to both the prescribing_date column and the test_date column.

Keyword searches are done using indexes created on the test_type, diagnosis, and description columns in the radiology_record as well as the first_name and last_name columns in the persons table. The following query is used to select the records that contain any of the key words that were searched for:

```
SELECT DISTINCT image_id, record_id, patient_id, first_name, last_name,
doctor_id, radiologist_id, test_type, prescribing_date, test_date, diagnosis,
description FROM
      (SELECT i.image_id, r.record_id, r.patient_id, p.person_id, p.first_name,
p.last_name, r.doctor_id, r.radiologist_id, r.test_type, r.prescribing_date,
r.test_date, r.diagnosis, r.description
      FROM radiology_record r, persons p, family_doctor fd, pacs_images i WHERE
      r.patient_id = p.person_id AND r.record_id = i.record_id AND Select
      thumbnail from pacs_images where image_id = :id;
```

Again checking for access rights is added to this query with the same additions as before. The keywords are searched for by adding the following contain statements on previously created indexes in a loop searching for multiple keywords if necessary:

```
CONTAINS(r.description, '$word', $i) > 0 OR
CONTAINS(r.diagnosis, '$word', $i) > 0 OR
CONTAINS(p.first_name, '$word', $i) > 0 OR
CONTAINS(p.last_name, '$word', $i) > 0 OR
CONTAINS(r.test_type, '$word', $i) > 0 OR
```

Both the date range and keyword search display the thumbnails of images that were uploaded for the record. This is done by using the image_id's that were selected in the previous select statements send to displayImage, along with the size id of 0 to specify the thumbnail size.

Data Analysis

OLAP_report.php: An an error message is generated if the user does not have the requisite admin privileges, otherwise it allows the user to give in any combination of patient, test type and diagnosis and returns a data analysis for that combination. This query selects all of the distinct test types in the radiology_record table:

```
SELECT distinct test_type FROM radiology_record;
```

This query selects all of the distinct patient names (first and last) in the persons table:

```
SELECT distinct first_name, last_name, person_id FROM persons;
```

A table is created with the person_id's, test_type, image_ids and test_date to enable a select of the count of the distinct number of images that each record has.

```
CREATE TABLE joined AS (SELECT p.person_id AS pid, r.test_type AS test,
i.image_id AS image, r.test_date AS tdate FROM persons p, radiology_record r,
pacs_images i WHERE p.person_id = r.patient_id AND r.record_id = i.record_id);
```

This selects the number of images each record has on the time period specified by the user.

```
SELECT COUNT(DISTINCT image) FROM joined WHERE pid = '$PatientIds[$int]' AND
test = '$test' AND EXTRACT (DAY FROM tdate) <= ('$w' + 1)*7 AND EXTRACT(DAY
FROM tdate) > '$w'*7 AND EXTRACT(MONTH FROM tdate) = '$month' AND EXTRACT (YEAR
FROM tdate) = '$year';
```

Variations of this query exist with fewer conditions depending on the combination of input values