Project report

1. Login module

(loginpage.php) First, it connects to the database. After establishing connection, we get every input information. By using the first sql statement, we get the correct password according to the input user name (through the users table).

The second and third sql statements are asked to get all other information from users table. While getting the correct password, we compare the correct password to the input password. If they are match then login successfully, elsewise it shows ‘invalid password’ and can not login.

(updateinfo.php) while login, the system get person\_id (through second sql). Because person\_id is the foreign key of person table, the system can get all the corresponding information in ‘persons’ table. By using the forth sql statement, the required information will be updated. The system can get all the information in ‘persons’ table after updating using fifth sql.

(updatepword.php) In order to update password, the system need to check if the user know the old password. The system will get correct password by the 6th sql and then compare to the old password. If they are match, the system will update password in users table by 7th sql, otherwise, the sysem will not allow you to update password.

(logout.php) It will process log out. It will destroy all of the data associated with the current session.

Sql statements:

1. SELECT password FROM users WHERE user\_name=’$uname’(input user name)
2. SELECT person\_id FROM users where user\_name=’$uname’(input user name)
3. SELECT role FROM users where user\_name=’$uname’(input user name)
4. UPDATE persons SET first\_name=’$\_POST[first\_name]’ WHERE person\_id=’$personID’
5. SELECT \* FROM persons WHERE person\_id=’$personID’
6. SELECT psaaword FROM users WHERE user\_name=’$UserName’
7. UPDATE users SET pawweord=’$\_POST[new\_pw]’ WHERE user\_name=’$UserName’
8. Sensor and user management module

(welcomAdim.php) after login as administrator, it will goes to this page and then user can modify sensor and user accounts.

(createSensor.php) First it will check if the user login or not by check if the input user name exists or not. It will also check if the user is administrator on not. If the user is administrator then it can continue. Then, it will check the input sensor id has already existed in database or not using 1st sql. If the sensor id exists,

the sql will return something and the system will report ‘Sensor ID already exist’. If the sensor id does not exist, the sql will return nothing and it will excute 2nd sql to insert new sensor.

(removeSensoe.php) First it will check if the user login or not by check if the input user name exists or not. It will also check if the user is administrator on not. If the user is administrator, then it can continue. The system will ask user to input sensor id. By using 3rd sql, it will return sensor id if the delete id is in the database, if the delete id is not in the database, it will return nothing, and the system will not allow users to delete. When the delete id is in the database, the sensor need to be deleted from subscription because the sensor id is part of key of table subscription using 4th sql. Also, the sensor need to be deleted from sensors because the sensor is key of table sensors using 5th sql.

(enterUser.php) First it will check if the user login or not by check if the input user name exists or not. It will also check if the user is administrator on not. If the user is administrator, then it can continue. The system will ask user to input new user name. By using 6th sql, it will return user name if the user name is in the database, if the user name is not in the database, it will return nothing. By using 7th sql, it will check if the person id is already in database or not. If they are both not in database, the system will allow to insert because they are key ans key cannot be duplicate. The system will insert new user using 8th sql.

(removeUser.php) First it will check if the user login or not by check if the input user name exists or not. It will also check if the user is administrator on not. If the user is administrator then it can continue. It will check if the delete user is in database or not. If the user is in the database, then the system will allow to delete using 9th sql.

(updateUser.php) First it will check if the user login or not by check if the input user name exists or not. It will also check if then user is administrator on not. If the user is administrator the it can continue. It will check if the user updated is in database or not using 10th sql.

(updateUser2.php) If the user exists, according to user name then update role using 11th sql.

Sql statements:

1. SELECT sensor\_id FROM sensors WHERE sensor\_id=’$\_POST[new\_id]’
2. INSERT INTO sensors VALUES (‘$\_POST[new\_id]’, ‘$\_POST[new\_local]’, ‘$\_POST[new\_type]’, ‘$\_POST[new\_disc]’)
3. SELECT sensor\_id From sensors WHERE sensor\_id=’$\_POST[delete\_id]’
4. DELETE FROM subscriptions WHERE subscriptions.sensor\_id=’$\_POST[delete\_id]’
5. DELETE FROM sensors WHERE sensor.sensor\_id=’$\_POST[delete\_id]’
6. SELECT user\_name FROM users WHERE user\_name='$\_POST[new\_un]'
7. SELECT \* FROM persons WHERE person\_id='$\_POST[new\_pid]'
8. "INSERT INTO users VALUES ('$\_POST[new\_un]', '$\_POST[new\_pw]', '$\_POST[new\_role]', '$\_POST[new\_pid]', TO\_DATE('$date', 'YYYY/MM/DD'))
9. DELETE FROM users WHERE user\_name = '$\_POST[delete\_un]
10. SELECT role FROM users WHERE user\_name='$\_POST[change\_un]'
11. UPDATE users SET role = '$\_POST[role]' WHERE user\_name='$UserUpdate'
12. Subscribe module

(viewSensor.php) First it will check if the user login or not by check if the input user name exists or not. It will also check if the user is administrator on not. If the user is scientist, then it can continue. Then it will show all information in table sensors on the web page using 1st sql. To add subscription, it will check the input sensor id has already in the table sensors or not using 2nd sql. If the sensor id exists,

the 3rd sql will check if the sensor has already subscribed or not. If the 1st sql return something and 2rd sql return nothing, the system will allow to add new subscription using 4th sql.

To delete subscription, it will check the input sensor id has already in the table sensors or not. If the sensor id exists, the 5th sql will check if the sensor has already subscribed or not. If the 5th sql also return something, the system will allow to delete subscription using 6th sql. The using 7th sql, the sensor information will update on the web page.

Sql statements:

1. SELECT \* FROM sensors
2. SELECT sensor\_id FROM sensors WHERE sensor\_id='$\_POST[add\_id]'
3. SELECT sensor\_id FROM subscriptions WHERE sensor\_id='$\_POST[add\_id]' AND person\_id='$PersonID'
4. INSERT INTO subscriptions VALUES ('$\_POST[add\_id]', '$PersonID')
5. SELECT sensor\_id FROM subscriptions WHERE sensor\_id='$\_POST[delete\_id]' AND person\_id='$PersonID'
6. DELETE FROM subscriptions WHERE person\_id='$PersonID' AND sensor\_id='$\_POST[delete\_id]'
7. SELECT s.sensor\_id, s.location, s.sensor\_type, s.description FROM sensors s, subscriptions su WHERE su.person\_id='$PersonID' AND s.sensor\_id=su.sensor\_id
8. Upload module

(uploadFile.php) First it will check if the user login or not by check if the input user name exists or not. It will also check if the user is data curator or not. If the user is data curator, then it can continue. If the upload file is image, the system will get the image id using 1st sql. After the user enter some information, it will insert the new image info into table images using 2nd sql. If the upload file is .wav, the system will get the recording id using 3rd sql. After the user enter some information, it will insert the new recording info into table audio\_recording using 4th sql. If the upload file is .csv, it will insert the new info into table scalar\_data using 5nd sql.

Sql statements:

1. SELECT image\_id FROM images
2. INSERT INTO images VALUES($count1, $\_POST[sensorId], TO\_DATE('$date\_created', 'DD/MM/YYYY'), '$\_POST[description]', EMPTY\_BLOB(), EMPTY\_BLOB()) RETURNING thumbnail, recoreded\_data INTO :tn, :rdata
3. SELECT recording\_id FROM audio\_recordings
4. INSERT INTO audio\_recordings VALUES($count2, $\_POST[sensorId], TO\_DATE('$date\_created', 'DD/MM/YYYY'), $\_POST[length], '$\_POST[description]', EMPTY\_BLOB()) RETURNING recorded\_data INTO :rdata
5. INSERT INTO scalar\_data VALUES ($vid, $sensorId, TO\_DATE('$dateCreated', 'DD/MM/YYYY HH24:MI:SS'), $value)
6. Search module

(Search.php) First it will check if the user login or not by check if the input user name exists or not. It will also check if the user is scientist or not. If the user is scientist, then it can continue. Then using 1st sql the user can get the image id under the user has according to the keyword (keyword is in the description of table sensors and the can get image information by foreign key. Using the second sql the user can get the image id under the user has according to the keyword.

Using 3rd sql the user can get the image id under the user has according to sensor location. (while getting the input location, we can get information from table sensors, then we can get image information according to the foreign key). Using 4th sql the user can get the recording id under the user has according to sensor location.

Using 5th sql the user can get the image id under the user has according to time range. (while getting the time range, we can get information from table images). Using 6th sql the user can get the recording id under the user has according to time range. Using 7th sql the user can get the data id under the user has according to time range.

Using 8th sql the user can get the recording id under the user has according to sensor type. (while getting the sensor type, we can get information from table sensor then we can get image information according to the foreign key). Using 9th sql the user can get the data id under the user has according to type.

While download images, the system will get the image id the user enters and according to the image id to get all image information (image id is the key of table images.) by using 10th sql.

While download audio recording, the system will get the recording id the user enters and according to the recording id to get all audio information (recording id is the key of table images.) by using 11th sql.

While download csv file, the system will get the file name the user enters and according to the combined tables to get all data information by using 12th sql.

sql statements:

1. SELECT images.image\_id FROM sensors , subscriptions , images WHERE subscriptions.person\_id='$PersonID' AND subscriptions.sensor\_id=sensors.sensor\_id AND sensors.description LIKE '%".$\_POST[search\_key]."%' AND images.sensor\_id=sensors.sensor\_id
2. SELECT audio\_recordings.recording\_id FROM sensors , subscriptions , audio\_recordings WHERE subscriptions.person\_id='$PersonID' AND subscriptions.sensor\_id=sensors.sensor\_id AND sensors.description LIKE '%".$\_POST[search\_key]."%' AND audio\_recordings.sensor\_id=sensors.sensor\_id
3. SELECT images.image\_id FROM sensors , subscriptions , images WHERE subscriptions.person\_id='$PersonID' AND subscriptions.sensor\_id=sensors.sensor\_id AND sensors.location LIKE '%".$\_POST[search\_key]."%' AND images.sensor\_id=sensors.sensor\_id
4. SELECT audio\_recordings.recording\_id FROM sensors , subscriptions , audio\_recordings WHERE subscriptions.person\_id='$PersonID' AND subscriptions.sensor\_id=sensors.sensor\_id AND sensors.location LIKE '%".$\_POST[search\_key]."%' AND audio\_recordings.sensor\_id=sensors.sensor\_id
5. SELECT images.image\_id FROM subscriptions , images WHERE subscriptions.person\_id='$PersonID' AND subscriptions.sensor\_id=images.sensor\_id AND images.date\_created>=TO\_DATE('$\_POST[search\_time1] 00:00:00', 'DD/MM/YYYY HH24:MI:SS') AND images.date\_created<=TO\_DATE('$\_POST[search\_time2] 23:59:59', 'DD/MM/YYYY HH24:MI:SS')
6. SELECT audio\_recordings.recording\_id FROM subscriptions , audio\_recordings WHERE subscriptions.person\_id='$PersonID' AND subscriptions.sensor\_id=audio\_recordings.sensor\_id AND audio\_recordings.date\_created>=TO\_DATE('$\_POST[search\_time1] 00:00:00', 'DD/MM/YYYY HH24:MI:SS') AND audio\_recordings.date\_created<=TO\_DATE('$\_POST[search\_time2] 23:59:59', 'DD/MM/YYYY HH24:MI:SS')
7. SELECT scalar\_data.id FROM subscriptions , scalar\_data WHERE subscriptions.person\_id='$PersonID' AND subscriptions.sensor\_id=scalar\_data.sensor\_id AND scalar\_data.date\_created>=TO\_DATE('$\_POST[search\_time1] 00:00:00', 'DD/MM/YYYY HH24:MI:SS') AND scalar\_data.date\_created<=TO\_DATE('$\_POST[search\_time2] 23:59:59', 'DD/MM/YYYY HH24:MI:SS')
8. SELECT audio\_recordings.recording\_id FROM sensors , subscriptions , audio\_recordings WHERE subscriptions.person\_id='$PersonID' AND subscriptions.sensor\_id=sensors.sensor\_id AND sensors.sensor\_type='$\_POST[search\_t]' AND audio\_recordings.sensor\_id=sensors.sensor\_id AND audio\_recordings.date\_created>=TO\_DATE('$\_POST[search\_time1] 00:00:00', 'DD/MM/YYYY HH24:MI:SS') AND audio\_recordings.date\_created<=TO\_DATE('$\_POST[search\_time2] 23:59:59', 'DD/MM/YYYY HH24:MI:SS')
9. SELECT scalar\_data.id FROM sensors, subscriptions , scalar\_data WHERE subscriptions.person\_id='$PersonID' AND subscriptions.sensor\_id=scalar\_data.sensor\_id AND subscriptions.sensor\_id=sensors.sensor\_id AND sensors.sensor\_type='$\_POST[search\_t]' AND scalar\_data.date\_created>=TO\_DATE('$\_POST[search\_time1] 00:00:00', 'DD/MM/YYYY HH24:MI:SS') AND scalar\_data.date\_created<=TO\_DATE('$\_POST[search\_time2] 23:59:59', 'DD/MM/YYYY HH24:MI:SS')
10. SELECT images.recoreded\_data FROM images WHERE images.image\_id='$\_POST[d\_imageid]'
11. SELECT audio\_recordings.recorded\_data FROM audio\_recordings WHERE audio\_recordings.recording\_id='$\_POST[d\_audioid]'
12. SELECT audio\_recordings.recording\_id FROM sensors , subscriptions , audio\_recordings WHERE subscriptions.person\_id='$PersonID' AND subscriptions.sensor\_id=sensors.sensor\_id AND sensors.description LIKE '%".$\_POST[search\_key]."%' AND audio\_recordings.sensor\_id=sensors.sensor\_id
13. Data analysis

(dataAnalysis.php) First it will check if the user login or not by check if the input user name exists or not. It will also check if the user is scientist or not. If the user is scientist, then it can continue. Then, the system will show the senor and data information form three table base on data cube information using 1st sql. To make sure no view temp before using 2nd sql and make a new table contains all the information we need. After the user enter the level, the system will check it and according to the input to perform generalization(roll up) and specialization(drill down) on daily weekly, monthly, quarterly, and yearly using 4th,5th, 6th, 7th and 8th sql.

Sql statements:

1. select s.sensor\_id, s.location, d.date\_created, avg(d.value), min(d.value), max(d.value) from scalar\_data d, sensors s, subscriptions sub where sub.person\_id = $PersonID and s.sensor\_id = sub.sensor\_id and s.sensor\_type = 's' and d.sensor\_id = s.sensor\_id group by cube (s.sensor\_id, s.location, d.date\_created)
2. drop view temp
3. create view temp as select s.sensor\_id sensor\_id, s.location location, d.value val, to\_char(d.date\_created,'YYYY') year, to\_char(d.date\_created,'Q') quarter, to\_char(d.date\_created,'MM') month, to\_char(d.date\_created,'IW') week, to\_char(d.date\_created,'ddd') day from scalar\_data d, sensors s, subscriptions sub where sub.person\_id = $PersonID and s.sensor\_id = sub.sensor\_id and s.sensor\_type = 's' and d.sensor\_id = s.sensor\_id
4. select sensor\_id, location, year, avg(val), min(val), max(val) from temp group by rollup (sensor\_id, location, year)
5. select sensor\_id, location, year, quarter, avg(val), min(val), max(val) from temp group by rollup (sensor\_id, location, year, quarter)";
6. select sensor\_id, location, year, month, avg(val), min(val), max(val) from temp group by rollup (sensor\_id, location, year, month)
7. select sensor\_id, location, year, week, avg(val), min(val), max(val) from temp group by rollup (sensor\_id, location, year, week)
8. select sensor\_id, location, year, day, avg(val), min(val), max(val) from temp group by rollup (sensor\_id, location, year, day)