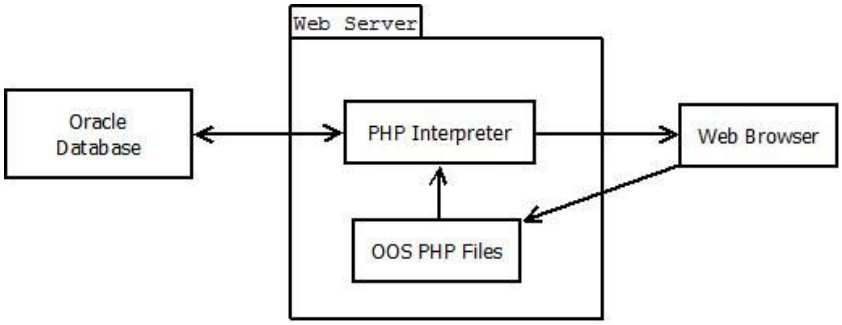
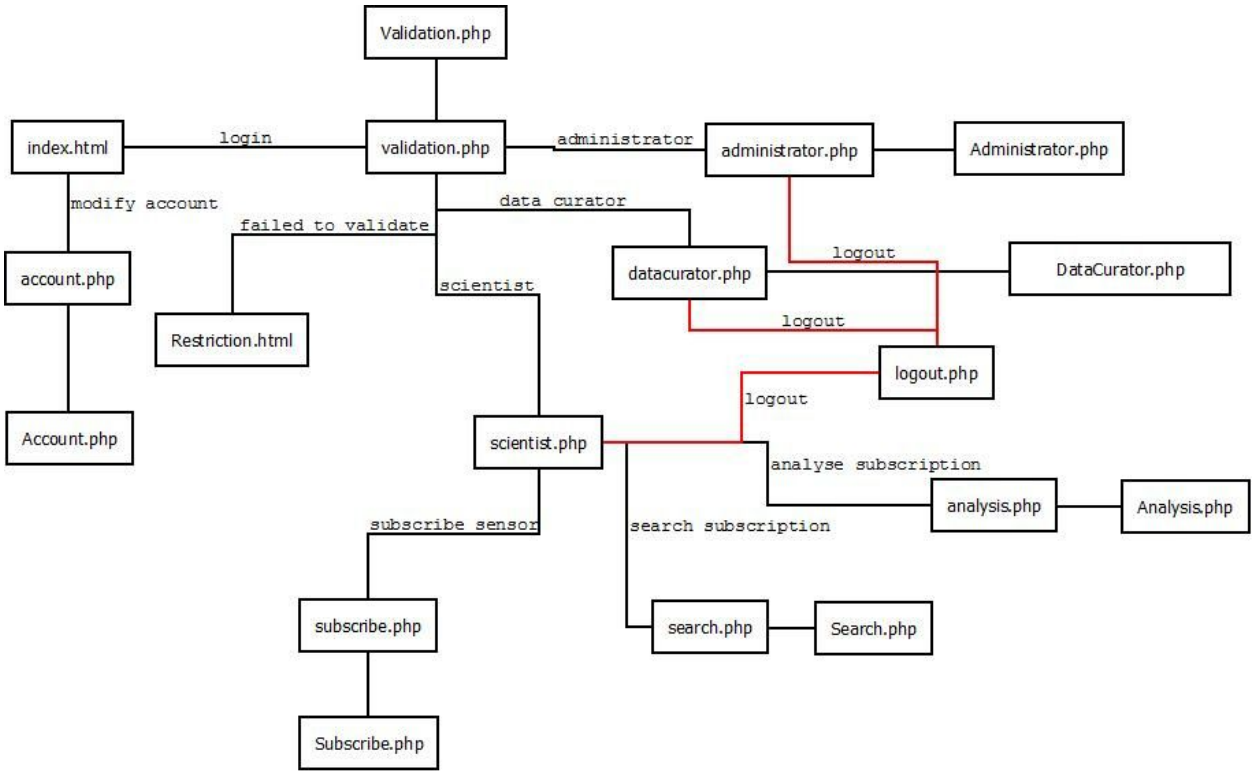
OOS System Architecture



Diagram



PHPconnectionDB.php

This module is used to connect to the database. Username and password are required in order to access the database and run sql statement. This module is used by all modules in this system.

index.html

This module is used as the starting page for the system. It asks user to login before accessing the system. It calls for validation.php (includes Validation.php) to validate user who is trying to login to the system.

restriction.html

This module is used to show a warning message that tells user is not allowed to use the system.

logout.php

This module is used to clear user session that is currently running. After logout, user will be sent to the main page (index.html) and will not be able to access the system. If user attempt to access the system after logout, user will be sent to restriction.html.

account.php / Account.php

This module is used to modify user and his personal information such as password, name, and address. It includes Account.php which performs show account and update user.

Method	SQL Statement
Show Account (Information)	`$sql = "SELECT u.user_name, u.password, p.first_name, p.last_name, p.address FROM users u, persons p WHERE u.person_id = p.person_id AND u.user_name = '".$username."'";`

Update User (and/or Personal information)	`// update password`
	`$sql ="UPDATE users SET password = '".$new_password."' WHERE user_name = '".$username."'";`
	
	`// update name and address`
	`$sql = "UPDATE (SELECT u.user_name, u.password, p.first_name, p.last_name, p.address FROM users u, persons p WHERE u.person_id = p.person_id AND u.user_name = '".$username."') SET first_name = '".$new_fname."', last_name = '".$new_lname."', address = '".$new_address."'";`

validation.php / Validation.php

This module is used to validate user login when the user attempts to login to the system. It checks the user role (a = administrator, d = data curator, s = scientist), and then it sends the user to his panel with proper privileges (ie. administrator will see Administrator panel and have administrator privileges). If the user does not exist in the database, the user will not be able to access the system and will be sent to restriction.html.

Method	SQL Statement
Validate	`$sql = "SELECT * FROM users WHERE user_name = '".$username."' AND password = '".$password."'";`

administrator.php / Administrator.php

This module is used by administrator to create and remove sensors, manage (enter, update, or remove) user accounts (persons and users). It includes Administrator.php which performs add person, add user, add sensor, delete user, delete person, delete sensor, update person and update user.

Method	SQL Statement
Add Person	`$sql = "INSERT INTO persons VALUES (".$person_id.", '".$firstname."', '".$lastname."', '".$address."', '".$email."', '".$phone."')";`
Add User	`$sql = "INSERT INTO users VALUES ('".$username."', '".$password."', '".$roles."', ".$user_id.", TO_DATE('".$date."', 'YY-MM-DD'))";`
Add Sensor	`$sql = "INSERT INTO sensors VALUES (".$sensor_id.", '".$location."', '".$types."', '".$description."')";`
Delete User	`$sql = "DELETE FROM users WHERE user_name='".$del_username."'";`
Delete Person	`$sql = "DELETE FROM persons WHERE person_id=".$del_person;`
Delete Subscriptions (in case person is a scientist)	`$sql2 = "DELETE FROM subscriptions WHERE person_id=".$del_person;`
Delete Sensor	`$sql = "DELETE FROM sensors WHERE sensor_id=".$del_sensor;`
Delete Audio Recordings	`$sql = "DELETE FROM audio_recordings WHERE sensor_id=".$del_sensor;`
Delete Image	`$sql = "DELETE FROM images WHERE sensor_id=".$del_sensor;`
Delete Scalar Data	`$sql = "DELETE FROM scalar_data WHERE sensor_id=".$del_sensor;`
Update Person	`$sql = "UPDATE persons SET first_name='".$edit_firstname."', last_name='".$edit_lastname."', address='".$edit_address."', email='".$edit_email."', phone='".$edit_phone."' WHERE person_id=".$dis_personid;`
Update User	`$sql = "UPDATE users SET password='".$user_password."', role='".$user_role. "', person_id=".$user_id." WHERE user_name='".$user_username."'";`
Search Person	`$sql = "SELECT * FROM persons WHERE person_id=".$edit_personid;`
Search User	`$sql = "SELECT * FROM users WHERE user_name='".$search_username."'";`

datacurator.php / DatacuratorFunction.php

This Model is used by scientist to upload audio recordings, images and scalar data. It includes datacurarotFunction.php which performs checking sensor id exists in database, upload an image to database, resize an image to thumbnail, generate an unique id for audio/image.

Method	SQL Statement
Check Sensor Id	`$sql = "SELECT * FROM sensors WHERE sensor_id = ".$sensorId;`
Upload Image	`$sql = "INSERT INTO images (image_id, sensor_id, date_created, description, thumbnail, recoreded_data) VALUES(".$image_id.", ".$sensor_id.", TO_DATE('".$date_created."', 'DD/MM/YYYY hh24:mi:ss'), '".$description."', empty_blob(), empty_blob()) RETURNING thumbnail, recoreded_data INTO :thumbnail, :recoreded_data";`
Upload Audio	`$sql = "INSERT INTO audio_recordings(recording_id, sensor_id, date_created, length, description, recorded_data) VALUES(".$audio_id.",".$sensor_id.", TO_DATE('".$date_created."', 'DD/MM/YYYY hh24:mi:ss'),".$length.",'".$description."',empty_blob()) RETURNING recorded_data INTO :recorded_data";`
Upload Scalar	`$sql = "INSERT INTO scalar_data VALUES (".$scalar_id.", ".$sensor_id.", TO_DATE('".$date."', 'DD/MM/YYYY hh24:mi:ss'), ".$value.")";`

Generate Id	`$sql = "SELECT * FROM IMAGES WHERE image_id = ".$id;`
	`$sql = "SELECT * FROM AUDIO_RECORDINGS WHERE recording_id = ".$id;`
	`$sql = "SELECT * FROM SCALAR_DATA WHERE id = ".$id;`

* The upload image/audio needs to encode the jpg/wav into a string and store it in the blob; this is done by php function called base64_encode.
* The upload scalar need to open the csv file, and read it line by line, and store each line into an array before insert it to the database.

scientist.php

This module is used by users who log in with the role of scientist to select from Sensor Subscription, Search Records and Data Analysis.

subscribe.php / Subscribe.php

This module is used by users who log in with the role of scientist to view all the existing sensors, subscribe to the existing sensors (with the correct corresponding Sensor ID), as well as to show the sensors that the user has already subscribed to, and unsubscribe to the sensors that the user has subscribed to by Sensor ID.

Method	SQL Statement

Display All Sensors	`$sql = "SELECT * FROM sensors";`

Display Subscribed Sensors	`$sql = "SELECT * FROM subscriptions WHERE PERSON_ID = ".$_SESSION['person_id'];`
	
	`$sql = "select * from sensors where SENSOR_ID = ".$sensor_ids[$i2];`

Subscribe	`$sql = "select * from sensors where SENSOR_ID = ".$sub_sensorid;`
	
	`$sql = "select * from subscriptions where SENSOR_ID = ".$sub_sensorid." and PERSON_ID = ".$_SESSION["person_id"];`
	
	`$sql = "Insert into subscriptions values(".$sub_sensorid.",".$_SESSION["person_id"].")";`

Unsubscribe	`$sql = "select * from sensors where SENSOR_ID = ".$unsub_sensorid;`
	
	`$sql = "select * from subscriptions where SENSOR_ID = ".$unsub_sensorid." and PERSON_ID = ".$_SESSION["person_id"];`
	
	`$sql = "delete from subscriptions where SENSOR_ID = ".$unsub_sensorid." and "."PERSON_ID = ".$_SESSION["person_id"] ;`

*$_SESSION['person_id'] means the person id of the user who currently log in
*$sensor_ids[$i2] means the IDs of the sensors the user is subscribed to

search.php / Search.php

This module is used by users who log in with the role of scientist to search by typing in keywords(optional), sensor location(optional), and date range(required) and selecting sensor type(all sensor types, image, aduio or scalar). Then it returns and displays the corresponding image thumbnails, audios and scalar data and also allows the user to download original image, audio and scalar data in csv file format.

Method	SQL Statement

Search Sensor	`$sql = "SELECT * FROM subscriptions NATURAL JOIN sensors WHERE PERSON_ID = ".$_SESSION['person_id'];`
	
	`$sql = $sql." AND SENSOR_TYPE = '".$_POST['type']."'";`
	
	`$sql = $sql." AND LOCATION = '".$_POST['location']."'";`
	
	`$sql = $sql." AND description like '%".$_POST['description']."%'";`

Search Image	`$search_string_image = "select thumbnail, recoreded_data from images where SENSOR_ID = ".$sensor_ids[$i2]." AND date_created between TO_DATE('".$_POST['from']."','DD/MM/YYYY hh24:mi:ss') and TO_DATE('".$_POST['to']."','DD/MM/YYYY hh24:mi:ss')";`
	
	`$search_string_image = "select thumbnail, recoreded_data from images where SENSOR_ID = ".$sensor_ids[$i2]." AND date_created between TO_DATE('".$_POST['from']."','DD/MM/YYYY hh24:mi:ss') and TO_DATE('".$_POST['to']."','DD/MM/YYYY hh24:mi:ss")";`

Search Audio	`$search_string_audio = "select recorded_data from audio_recordings where SENSOR_ID = ".$sensor_ids[$i2]." AND date_created between TO_DATE('".$_POST['from']."','DD/MM/YYYY hh24:mi:ss') and TO_DATE('".$_POST['to']."','DD/MM/YYYY hh24:mi:ss')";` `$search_string_audio = "select recorded_data from audio_recordings where SENSOR_ID = ".$sensor_ids[$i2]." AND date_created between TO_DATE('".$_POST['from']."','DD/MM/YYYY hh24:mi:ss') and TO_DATE('".$_POST['to']."','DD/MM/YYYY hh24:mi:ss')";`
Search Scalar	`$search_string_scalar = "select SENSOR_ID, TO_CHAR(DATE_CREATED, 'DD/MM/YYYY HH24:MI:SS') as DATE_CREATED, VALUE from scalar_data where SENSOR_ID = ".$sensor_ids[$i2]." AND date_created between TO_DATE('".$_POST['from']."','DD/MM/YYYY hh24:mi:ss') and TO_DATE('".$_POST['to']."','DD/MM/YYYY hh24:mi:ss")";`

* The search image/audio will use php function base64_decode to display the image/audio and the download link for it image/audio.
* The search scalar will use php function fputs to store the data to the csv file.

analysis.php/Analysis.php

This module is used by scientist to generate and display OLAP report for the analysis of scalar data that they are currently subscribed. This module creates a fact table to make the OLAP performance faster. It performs CUBE, roll up, and drill up on five levels of time hierarchies (daily, weekly, monthly, quarterly, yearly) for the report.

```
CREATE TABLE fact_table1
AS SELECT *
FROM
(
  SELECT s.sensor_id, s.location,
          TO_CHAR(d.date_created, 'YYYY') as year,
          TO_CHAR(date_created, 'Q') as quarter,
          TO_CHAR(date_created, 'Mon') as month,
          TO_CHAR(date_created, 'W') as week,
          TO_CHAR(date_created, 'D') as day,
          d.value
  FROM sensors s, scalar_data d
  WHERE s.sensor_id = d.sensor_id
```

```
  AND s.sensor_type = 's'
);
```

Above statement creates a fact table with the combinations of sensor id, location, day, week (of month), month, quarter (of year), year (date created), and value (scalar data). The fact table helps to speed up the OLAP operations.

Method	SQL Statement

Show yearly result	`SELECT sensor_id, location, year, AVG(value), MIN(value), MAX(value)`
	`FROM fact_table1`
	`GROUP BY CUBE(sensor_id, location, year)`

To show result yearly, we group the value by year.	
Show quarterly result	`SELECT sensor_id, location, quarter, year, AVG(value), MIN(value), MAX(value) FROM fact_table1` `GROUP BY CUBE(sensor_id, location, quarter, year)`
To show result quarterly, we group the value by quarter and year.	

Show monthly result	`SELECT sensor_id, location, month, quarter, year, AVG(value), MIN(value), MAX(value)`
	`FROM fact_table1`
	`GROUP BY CUBE(sensor_id, location, month, quarter, year)`

To show result monthly, we group the value by month, quarter and year.	
Show weekly (of month) result	`SELECT sensor_id, location, week, month,quarter, year, AVG(value), MIN(value), MAX(value) FROM fact_table1` `GROUP BY CUBE(sensor_id, location, week, quarter, month, year)`
To show result weekly, we group the value by week, month, quarter and year.	

Show daily (of week) result	`SELECT sensor_id, location, day, week, month, quarter, year, AVG(value), MIN(value), MAX(value)`
	`FROM fact_table1`
	`GROUP BY CUBE(sensor_id, location, day, week, month, quarter, year)`

To show result daily, we group the value by day, week, month, quarter and year.	