

OnlinePhotoGallery Project Report

V.1



Bo Zhou

Baihong Qi

Yueran Sun

Group 11

2016-Mar-31

1. Overall: 2

2. User Management Module: 2

3. Security Module: 3

4. Uploading Module: 4

5. Display Module: 4

6. Search Module: x

7. Data Analysis Module x

8. Other tables and files x

**Overall:**

This is an online photo sharing system called Online Photo Gallery.

You can get access to this system via [consort.cs.ualberta.ca/~bzhou2/](consort.cs.ualberta.ca/~bzhou2) if you have connected to the network of University of Alberta. You can share photo to other users and create group for sharing convenience.

Index Page: index.php:

This is the first page you will see when you get access to our system.

This page provides all inputs for login, and all inputs for sign up.

The form of login will be submit to login.php, and the from of sign up will be submit to signup.php.

**User Management Module:**

Related page:

index.php

login.php

signup.php

logout.php

main\_page.php

This module can be separated to three sub module, login, sign up, and log out.

1. Login module:

This module is implemented in login.php:

Use sql " SELECT user\_name, password, date\_registered FROM users WHERE users.user\_name='".$user\_name."' " to find the data from database and compare it with your input. If the user name exists, then compare the password. If the password is matched, load user information by “SELECT \* FROM persons WHERE user\_name = '".$user\_name."'” and go to main\_page.php with session opened. If user name doesn’t exist or password doesn’t matched, return to index.php and print error message.

2. Signup module:

This module is implemented in signup.php:

Use sql SELECT user\_name FROM users to find all user names in database then check if the input user name in the database.

Use sql SELECT email FROM persons to check if the email of input exist or not.

If constraints are matched, insert other information into table persons, then go to main\_page.php with session opened.

3. Log out module:

This module is implemented in logout.php.

It kills all session then go to index.php.

User will stay at main\_page.php until log out. The main\_page.php has top bar to log out if needed and left bar to select other module. The main\_page.php is based on bootstrap library. All other module will be insert into the iframe tag in main\_page.php .

**Security Module:**

Related page:

groups.php

add-group.php

select-show.php

add-friend.php

edit-friend.php

delete-friend.php

This module is implemented in group.php. It includes group creation and group edit.

1. Group Creation:

It has a form to let user input a new group name to create.

Form will be submitted to add-group.php.

- Use SELECT group\_id, group\_name FROM groups WHERE user\_name='".$user\_name."' to check if the group name exist or not.

- Use SELECT group\_id, group\_name FROM groups WHERE user\_name IS NULL to check if the new group name has conflict with system added group (public or private).

If all constraints are matched, insert group information into table groups, also use INSERT INTO group\_lists VALUES ('".$id\_guess."', 'admin', '".$now\_date."', 'system added') to insert admin as a friend in the group list of this group.

1. Group Edit:

When user select an exist group, it will call select-show.php via jquery and ajax.

In select-show.php, it returns all information about the group you select by SELECT user\_name FROM users WHERE user\_name<>'admin' AND user\_name<>'".$user\_name."', SELECT friend\_id, notice FROM group\_lists WHERE group\_id ='".$q."' AND friend\_id<>'admin' and SELECT group\_name FROM groups WHERE group\_id = '".$q."'. Then return these information back to group.php.

Part 1 is used to let user choose all possible users which can be added to the group. The form will be submitted to add-friend.php.

In add-friend.php:

Use INSERT INTO group\_lists (group\_id, friend\_id, date\_added, notice ) VALUES ('".$group\_id."', '".$name."', '".$now\_date."', '".$notice."' ) to insert all information into table group\_lists.

Part 2 is used to let user delete or update the notice of exist friends in the group.

When user submits a form to delete friends or update the notice of the friends in the group list, the form will be submitted to edit-friend.php.

In edit-friend.php:

If the delete from friends button is clicked, it will use DELETE FROM group\_lists WHERE group\_id='" . $group\_id . "' AND friend\_id='" . $name . "' to delete selected friends from table group\_lists.

If the update notice button is clicked, it will use UPDATE group\_lists SET notice='".$notice."' WHERE group\_id='" . $group\_id . "' AND friend\_id='" . $name . "' to update new notice for selected friends.

Part 3 is used to delete the whole group.

It will be submitted to delete-group.php.

In delete-group.php, it will use UPDATE images SET permitted=2 WHERE permitted='".$q."' to change all images which is opened to this group to private. Then it will use DELETE FROM group\_lists WHERE group\_id = '".$q."' to delete all friends in the group. At the end, it will use DELETE FROM groups WHERE group\_id = '".$q."' to delete the group from table groups.

**Uploading Module:**

Related page:

upload\_file.php

upload\_folder.php

upload-one.php

upload-multi.php

This module is implemented in two part, one part is to upload one file, and the other part is to upload all images in one folder.

They all use the same method and sql query to upload files, in upload\_file.php and upload\_folder.php, they allow users to choose images and other image information for submission. Then these information will be submitted to upload-one.php and upload-multi.php separately.

In upload-one.php and upload-multi.php, it will move pre-loaded image to a temp folder in order to resize it into a thumbnail. Then use INSERT INTO images (photo\_id, owner\_name, permitted, subject, place, timing, description, thumbnail, photo) VALUES('" . $id\_guess . "', '" . $owner\_name . "', '" . $permitted\_id . "', '" . $subject ."', '". $place ."', '". $now\_date."', '". $description."', empty\_blob(), empty\_blob() ) RETURNING thumbnail, photo INTO :thumb\_img, :object” to insert the thumbnail, original photo and other information into table images. Meanwhile, bind thumbnail and photo with variables thumb\_img and object which are two empty blob type variables. Then save the binary data into the table.

**Display Module:**

Related pages:

all\_photos.php

own\_images.php

imageView.php

show\_image.php

edit\_image.php

update-image.php

delete-image.php

This module is implemented into two parts.

1. Explore Photo:

Use SELECT i.photo\_id FROM images i WHERE owner\_name <> '".$user\_name."' AND '".$user\_name."' IN (SELECT gl.friend\_id FROM group\_lists gl WHERE gl.group\_id = i.permitted) ORDER BY i.timing DESC, i.owner\_name ASC to get all photos that the user can view, and show those thumbnail in Other users’ photos fieldpart.

Use SELECT iv.photo\_id, count(\*) AS numberOfviewer FROM images\_viewed iv WHERE iv.photo\_id NOT IN (SELECT i.photo\_id FROM images i WHERE i.permitted = '2') GROUP BY iv.photo\_id ORDER BY numberOfviewer DESC to choose the photos which are not in private group and sorted by the count of viewed.

If all images haven’t been viewed yet or the number of popular photos are less than 5, it will use SELECT i.photo\_id FROM images i WHERE i.permitted <> '2' AND i.photo\_id NOT IN (SELECT iv.photo\_id FROM images\_viewed iv) ORDER BY i.timing DESC to find all fresh photos that are newly uploaded and haven’t been viewed as well.

All photo\_id will be passed to imageView.php. In that page, it uses SELECT thumbnail, photo FROM images WHERE photo\_id=". $\_GET['image\_id'] to find the binary data of the image then return to img tag for showing.

Each thumbnail can be clicked to see it original photo, the photo id will be passed to show\_image.php.

In show\_image.php, it will check if the user has permission to see the original photo. Use SELECT g.friend\_id FROM images i, group\_lists g WHERE i.photo\_id = '".$id."' AND i.permitted = g.group\_id to check if the user is in the group list of the group this image opened to, and use SELECT owner\_name FROM images WHERE photo\_id = '".$id." to check if the user is the owner of the image. If one of the condition matched, pass the photo id to imageView.php to get data of the photo, otherwise, it will show a reject image for showing forbidden.

1. Manage own photos:

It implemented in own\_images.php. In own\_images.php, it use SELECT i.\*, g.group\_name FROM images i, groups g WHERE i.owner\_name = '".$user\_name."' AND i.permitted = g.group\_id to show all the images the owner is the user or use SELECT i.\*, g.group\_name FROM images i, groups g WHERE i.permitted = g.group\_id to show all images if the user is admin.

The thumbnail is allowed to click to view the original photo.

If edit is clicked, photo id will be passed to edit\_image.php.

In edit\_image.php, it uses SELECT i.\*, g.group\_name FROM images i, groups g WHERE i.photo\_id = '".$id."' AND i.permitted = g.group\_id, SELECT g.group\_id, g.group\_name FROM groups g, images i WHERE g.user\_name=i.owner\_name AND i.photo\_id='".$id." and SELECT group\_id, group\_name FROM groups WHERE user\_name IS NULL to find all information about this image. It also provides input fields for information update.

If delete is clicked, photo id will be passed to delete-image.php.

In delete-image.php, it uses DELETE FROM images\_viewed WHERE photo\_id='".$id." to delete all viewed information of this photo in table images\_viewed, then use DELETE FROM images WHERE photo\_id='".$id."' to delete the image from table images.

**Search Module:**

**Data Analysis Module:**

**Other table or file:**

New table: images\_viewed.

Created by

CREATE TABLE images\_viewed (

photo\_id int,

viewer varchar(24),

PRIMARY KEY(photo\_id, viewer),

FOREIGN KEY(viewer) REFERENCES users,

FOREIGN KEY(photo\_id) REFERENCES images

);

It records the history of count of viewed of one image.

connDB.php:

This file is used to run connect function for all queries in the whole system.