OnlinePhotoGallery Project Report

V.1

Bo Zhou

Baihong Qi

Yueran Sun

Group 11

2016-Mar-31

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/ 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', '".$now_date."', 'system added') to insert admin as a friend in the group list of this group.
2. Group Edit:
   When user select an exist group, it will call select-show.php via jquery and ajax.
   In select-show.php, it returns all information about the group you select by SELECT user_name FROM users WHERE user_name<>'admin' AND user_name<>'".$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.

2. 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:

Related page:

search.php
connDB.php
imageView.php
show_image.php

Search for the photo result:

After we insert a list of keywords and a time period and notice the system how to rank the reult, It execute the following query in cracle database.

SELECT photo_id, thumbnail, ((SCORE(1) * 6) + (SCORE(2) * 3) + SCORE(3)) score This statement select the proper attributes of the image need to display and order. Score represent how frequency keyword occurs in different column,

using the function: Rank(photo_id) = 6*frequency(subject) + 3*frequency(place) + frequency(description)

FROM images WHERE (CONTAINS (subject, '%cat%', 1) > 0 OR CONTAINS (place, '%cat%', 2) > 0 OR CONTAINS (description, '%cat%', 3) > 0)

This indicate what kind of photos should be selected, using the function CONTAINS tto find if the keywords occurs in the columns,and (owner_name = 'Aa' or 'Aa' = 'admin' or permitted = 1 or permitted in (SELECT group_id FROM group_lists WHERE friend_id = 'Aa') or 'Aa' in (SELECT user_name FROM groups WHERE group_id = permitted)) This rep[resent which photo has the permission to be diaplayed. It should be the photo of the user, or user is the admin, or user is in the group of this image, or user is the owner of the group.

and timing >= TO_DATE('2016/04/01', 'yyyy/mm/dd') and timing <= TO_DATE('2016/04/15', 'yyyy/mm/dd') ORDER BY timing DESC this statement is added to the end of query if a time peroid is selected. ORDER BY score DESC

if no time peroid is given, then the table is ordered by the score.

Data Analysis Module:

Related page:

dataAnalysis.php

Analysis the data:

We use the Check box to send message to Form the SELECT clause, WHERE clause and GROUPY CLAUSE. There are four Input Text which allow admin to specify user, subject, start date and end date. Admin can enter keywords of user name and image's name to get the corresponding user and image. Admin also can enter a start Date and End Date to limit a time range.

The corresponding SQL statement is

```
SELECT owner_name, subjects, EXTRACT(YEAR FROM timing) year,
EXTRACT(MONTFROM timing) month,
TO_CHAR(timing+1,\'IW\',\'NLS_DATE_LANGUAGE = American\') week
' WHERE owner_name in ('.$nameList.')',AND CONTAINS (subject, \".$contains.'\', 1) > 0
AND timing >= TO_DATE(\".$startDate.'\', \'yyyy/mm/dd\')
AND timing <= TO_DATE(\".$endDate.'\', \'yyyy/mm/dd\') +1

GROUP BY owner_name,subject,EXTRACT(YEAR FROM timing),
EXTRACT(MONTH FROM timing),TO_CHAR(timing,\'IW\',\'NLS_DATE_LANGUAGE = American\')
```

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.