391 Project Report

­­

**­­By:**

**Hugh Craig**

**Daniel Shin**

**Ismail Mare**

# Introduction

Our photo sharing web application allows users to upload photos, share them with friends and family and also view pictures uploaded by friends and family. Users can create groups of users and share photos only to that group and also share to all users. Our system is hosted on UofA servers at consort.cs.ualberta.ca, and the databases are hosted on UofA Oracle servers. The system was built using a combination of HTML, PHP, CSS, Bootstrap, and JavaScript.

## General Files

* **index.php:** Opens login/signup.php webpage if their designated links are clicked. Contains the web-app logo and signup and login buttons. This is the first file to be called when starting the client at consort.cs.ualberta.ca/~<ccid>. This page has two buttons that redirect the user to login.php, or signup.php. Checks if there is an active session meaning there is a user still logged in. If this is the case the user is redirected immediately to profile.php. This file displays the user.
* **header.php:** The header on the pages at the top that allow the user to redirect to other modules. There are links to searching, groups, displaying and uploading pictures. This file is required in many other PHP files.
* **profile.php:**

Requires setup.php to access the database and homepage.php to display the header files, wallpaper, and logo. Called after a user logs in or signs up. Displays some information about the user. Also is called when a user clicks the home button at the top of the page displayed by the header.php.

SELECT user\_name, first\_name, last\_name, address, email, phone FROM persons WHERE user\_name =\''.$username.

This query is used to get information about the user. This information is displayed to the user.

* **function.php:** Functions necessary for string sanitizing, and creating and destroying sessions.

## Database Connection Files:

* db.php:

File that holds classes and methods for connecting to the database. From this file we can make a connection, and execute commands. The executeStatement function will take an SQL query after a connection is made and queries the database, and returns the array of results.

This file also contains the following methods

* connect () Used to establish a connection to the database.
* getConnection () returns the connection
* disconnect () to disconnect from the database.
* executeStatement($stmt) for executing generic statements.
* executeStatementAlt($stmt) for executing statements with array return values.

* setup.php:

This file will generate a database object. Calls db.php to create that object and create a connection. This is where we specifiy the username and password to access the database

* setup.sql:

SQL commands to generate tables required for this database.

* config.php:

Used for database connection. Where we specifiy the host, port and SID for the database.

## Style sheets (/include/css directory):

* signin.css: css file for formatting. Used in multiple files
* theme.css: css file for formatting . Used in multiple files

## Images (/include/images directory):

* bgimage.jpg: Background picture for website
* logo.png: Photoshare logo in blue
* logored.png: Photoshare logo in red

## Usermanagement Module (/usermanagement directory):

This module allows your clients to register an account by providing required information, including a unique user name, password, first name, last name, address, email, and phone number, and to log in as a registered user to perform various tasks.

* **usermanagement:**

The directory that holds all the files that are related to signing up, logging in and logging out to the website.

* **login.php:**

Requires setup.php to access the database and homepage.php to display the header files, wallpaper, and logo. This file is called from index., displays the login form. Does some error checking. Making sure the fields have values, and are a certain length. This file contains mostly html and bootstrap for displaying forms and logos, and JavaScript for checking the form. Making sure the form contains values are correct.

* **loginAuth.php:**

This file is called from login.php after the user clicks the login button in the login.php file. Checks if the username and password are in the database. If the user and password are valid, profile.php is called to display the users profile. If the user does not exist, the login.php page is called again and an alert is displayed with an error message. A session is created after a successful login.

Select user\_name from users where username=’$username’ AND password=’$password’

* **signup.php:**

Requires setup.php to access the database and homepage.php to display the header files, wallpaper, and logo.

This file is called from index.php. Allows the user to register to the application. Does some error checking. Makes sure all the values are entered, passwords match, valid phone number and a valid email. Uses html, bootstrap and JavaScript.

* **signupAuth.php:**

This file is called from signup.php. Checks if the username is taken. If the username is not taken and the values entered are valid, a row is created in the users table and in the persons table in the database. A session is created, the user is logged in.

SELECT user\_name FROM users WHERE user\_name =\''.$username.

-uses to check if the username is in the database and not unique

Insert into users values ($username, $password, NULL)

Insert into persons values ($username, $firstname., $lastname, $address., $email, $phone)

-Adding users to the users table and persons table in the database.

* **logout.php:**

This file is called if the user clicks logout. Destroys the session, and logs the user out; index.php is then called to allow a new user to be logged or register.

## Data Analysis Module (/admin directory):

* **admin.php:**
* **adminDB.php:**
* **adminSQL.sql**

## Searching Module (/search directory):

* **search.php**
  + This file contains the frontend forms for searching by keywords and/or dates, and also to sort by most-recent, oldest, or default. There is also an option for retrieving the top five most popular images visible to the user. The form data is sent to searchDB.php to evaluate the search. Before creating queries to search the data based on their expressions entered, we must rebuild our indexes incase photos are uploaded/updated.
* **searchdb.php**
  + This file handles the search request by formulating a query based on the expressions entered from the form on search.php. It solves this by incrementally building up a query based on whether or not a user entered certain fields, or pressed certain search condition buttons.
  + SQL Statements used (since the SQL statement varies based on whether the user enters certain criteria, I will present partial conditions used for the query)

If keywords are not empty, then we have a query condition for checking the indexes:



If the user has entered a date time frame:



The conditions to check if the user has permission to view the image:



If the user has not specified an ordering to the search:

If the user has specified most recent images first:

If the user has specified oldest images first:

All of the previous conditions are appended to the following simple query depending on if they were entered or not:



Query for selecting top five images



The photo id’s that are returned from these queries are then

placed into a one-dimensional array and sent to the searchResult.php page to be displayed to the user, otherwise they are sent back to search.php with a corresponding message of no results found.

* **moreinfo.php**
  + Part of the display module that allows users to click on a photo returned from the search to enlarge it and give more information on it. It also das
  + The queries associated with this can be found in the display module of this page.
* **searchResult.php**
  + This page is used to display the photos retrieved from the search executed in searchdb.php. It is passed all the photo ids that were returned by the search in a PHP session variable and then displays all the images in a HTML table. When you click on a photo, it also enlarges it with the details.
  + SQL Statements used:

****

## Uploading Module (/upload directory):

## uploadForm.php

* Requires setup.php to access the database and homepage.php to display the header files, wallpaper, and logo.

Users can upload pictures to the website, and they must fill out some basic information about the picture. If all the required fields are completed, they will be sent to uploadDB.php.

* **uploadDB.php**

Requires setup.php to access the database.

This file will insert the image uploaded by the user into the database.

The regular sized image, and a resized smaller image (thumbnail) will be stored in the database as a blobs.

INSERT INTO images (photo\_id, owner\_name, permitted, subject, place, timing, description, thumbnail, photo) VALUES (:photoid, :ownername, :permitted, :subject, :place, TO\_DATE(:datetime, \'MM/DD/YYYY\'), :description, empty\_blob(),

empty\_blob ()) returning thumbnail, photo into :thumbnail, :photo';

* Inserting images

'SELECT DISTINCT(photo\_id) FROM images'

* **updateImage.php:**

Requires setup.php to access the database.

Users can select the images they wish to upload from the display module, and will be sent to this file. Users can then update any fields.

* **updateImageDB.php**

UPDATE images SET permitted = :permitted, subject = :subject,place =:place,timing = TO\_DATE(:datetime,\'MM/DD/YYYY\'),description =:description,WHERE photo\_id = :image\_id';

- This the query where we update the pictures info.

## Security Module:

* One cannot login into the system without providing a valid user name and password.

This is enforced in the user management module, and usermanagement directory.

* Each user may create any number of groups, each of which has a group name, and consists of a list of user names.

In the group directory, users can login create any number of group, and any number of users to a group. The only constraint is the group name must be unique to the user. Meaning two users can have the same group name, but a user may not have the same group name for two groups.

* Each image stored in the system has a unique owner, and the image, as well as its descriptive information, can only be updated and/or removed by its owner.

A user can upload images, and their username is stored as the owner. Only the user can update the images information, or delete the image entirely.

* Each image can be viewed by public, i.e., by any user

a designated group, i.e., all users in the group, or

private, i.e., by the owner only.

This is enforced when a user searches for images.

Depending on an images security settings, only images that are permitted for them to see will be shown. Eg. Groups they are in, or if images are public.

## Groups:

* groups.php:

Requires setup.php to access the database and homepage.php to display the header files, wallpaper, and logo.

Displays the groups the user has created. From here the user has the options to create a new group, add friends to a group or delete friends from a group.

* groupAdd.php

Requires setup.php to access the database and homepage.php to display the header files, wallpaper, and logo. The user can create a new group here. Provided the group name is unique to the other groups they already own. Calls groupAddDB.php to add to the database

* groupAddDB.php

Requires setup.php to access the database. Adds a group to the database.

'SELECT group\_name FROM groups WHERE group\_name =\''.$group\_name.'\' AND user\_name =\''.$user.'\'')

This is a check to determine if the user already owns a group with the same name.

'Insert into groups values (\''.$group\_id.'\',\''.$user.'\',\''.$group\_name.'\',SYSDATE)';

Group is then added to the database.

* friendsAdd.php

Requires setup.php to access the database and homepage.php to display the header files, wallpaper, and logo. User can specify a group, a username of their friends, and a notice message. Calls friendsAddDB.php to add the friend.

* friendsAddDB.php

Requires setup.php to access the database. Adds a friend to a group that the user owns.

('SELECT group\_id FROM groups WHERE group\_name =\''.$group\_name.'\' AND user\_name =\''.$user.'\'');

Check to determine if the user is the owner of the group.

('SELECT user\_name FROM users WHERE user\_name =\''.$friend.'\'');

Check to see if the username exists.

'Insert into group\_lists values (\''.$group\_id.'\',\''.$friend.'\',SYSDATE,\''.$notice.'\')';

Friend is then added to the database

* friendsRemove.php

Requires setup.php to access the database and homepage.php to display the header files, wallpaper, and logo. User can specify a group, a username of their friends, and a notice message. Calls friendsRemoveDB.php to remove the friend.

* friendsRemoveBD.php

Requires setup.php to access the database. Removes a friend to a group that the user owns.

('SELECT group\_id FROM groups WHERE group\_name =\''.$group\_name.'\' AND user\_name =\''.$user.'\'');

- Check to determine if the user is the owner of the group.

('SELECT user\_name FROM users WHERE user\_name =\''.$friend.'\'');

Check to see if the username exists.

-' Insert into group\_lists values (\''.$group\_id.'\',\''.$friend.'\',SYSDATE,\''.$notice.'\')';

- Friend is then removed from the database

SELECT friend\_id FROM group\_lists WHERE group\_id = $group\_id and friend\_id = $friend

- Checks to see if indeed the user has a friend for a particular group ID.

## Display Module (/display directory):

* **display.php:**

Display’s the images owned by the user. Requires setup.php to access the database and homepage.php to display the header files, wallpaper, and logo. Displays all the images in a table. User has a drop down action list. Options include delete images, and update images and its security settings. If delete image link clicked, user will be sent to delete.php where they will select the images they would like to delete. If they click update, they will be redirected to update.php where they will select images to update.

Users also can click an image and they will be redirected to displaySingle.php where the single image will be displayed, along with more information including subject, place, description, and more.

* **delete.php:**

Requires setup.php to access the database and homepage.php to display the header files, wallpaper, and logo.

Here, users select images they wish to delete. They check off the pictures they wish to delete. This file then hands off an array of photo\_ids to deleteDB.php

* **update.php**

Requires setup.php to access the database and homepage.php to display the header files, wallpaper, and logo.

Users can check off all the images they wish to change. They can select one or more. This file then hands off an array of photo\_id’s to updateImage.php in the upload directory. The users can then enter new information and all the photos will be updated.

* **deleteDB.php**

Requires setup.php to access the database.

'DELETE FROM images WHERE photo\_id=\''.$check.'\''

$check is one photo\_id from an array of photo id’s

* **moreinfo.php:**

Requires setup.php to access the database and homepage.php to display the header files, wallpaper, and logo.

Pictures displayed in display.php can be clicked, and then they will be sent here where the user can get more details about a picture.