*PhotoShare: An on-line photo social network system*

**Photoshare Report**

Yufan Lin, Jiayu Gu

Mar 22, 2022

**Purpose of the Project**

In this project, we design, implement and document a database system for a web-based photo sharing application. We also provide the web-based interface to the database. Our final system is functional and similar to Flickr!

**Data**

The system manages the following information:

Users

Each user is identified by a unique user id and has the following attributes: first name, last name, email, date of birth, hometown, gender, and password. A user can have a number of Albums.

Friends

Each user can have any number of friends.

Albums

Each album is identified by a unique album id and has the following attributes: name, owner (user) id, and date of creation. Each album can contain a number of photos.

Photos

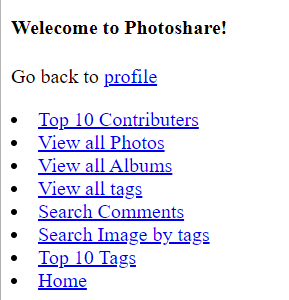
Each photo is identified by a unique photo id and must belong to an album. Each photo has the following attributes: caption and data. The 'data' field should contain the actual binary representation of the uploaded image file. Alternatively, the 'data' field can store the file location of the file that stores the image. Each photo can only be stored in one album and is associated with zero, one, or more tags.

Tags

Each tag is described by a single word. Many photos can be tagged with the same tag. For the purpose of this project we will assume that all tags are lower-cased and contain no spaces. For example, you can have many photos tagged with the "Boston" in different albums.

Comments

Each comment is identified by a unique comment id and has the following attributes: text (i.e., the actual comment), the comment's owner (a user) and the date the comment was left.



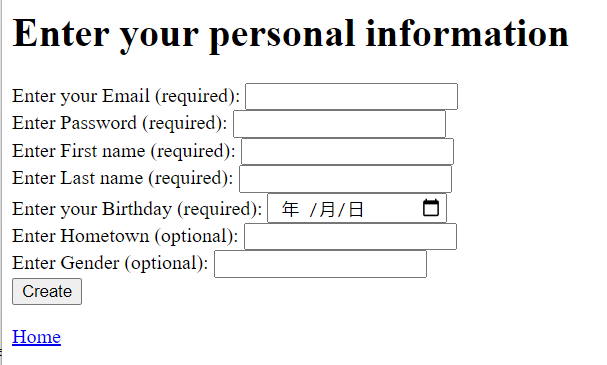
*(main page)*

**Use Cases**

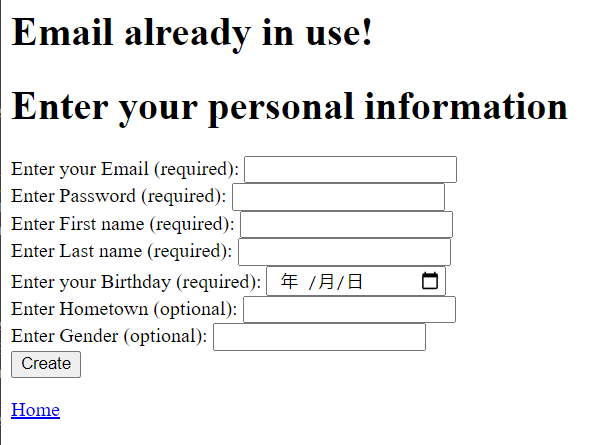
The following interaction with the system has been implemented.

**User Management**

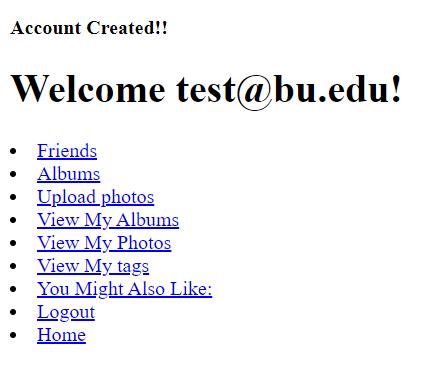
**Becoming a registered user.** Before being able to upload photos, a user should register by providing their first name, last name, email address, date of birth, and a password. If the user already exists in the database with the same email address an error message should be produced. The other additional information about each user is optional.



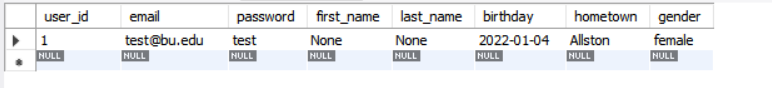
*(If the user already exists in the database with the same email address, an error message should be produced)*



*(Otherwise, message ‘Account created’)*



*(User table)*



*(Guest mode page)*

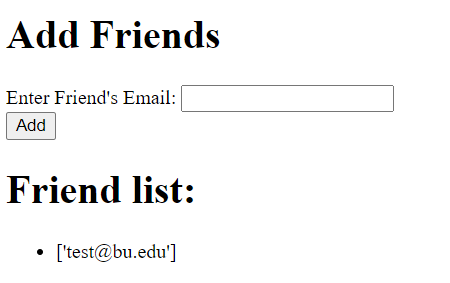


**Adding and listing Friends.** A user can add a new friend to the friend list. No need to verify the friendship relationship. Also, a user can to search for other users in the system (in order to find friends to add). Finally, a user can list his/her friends.

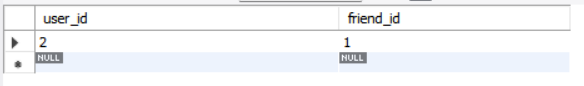
*(Before)*



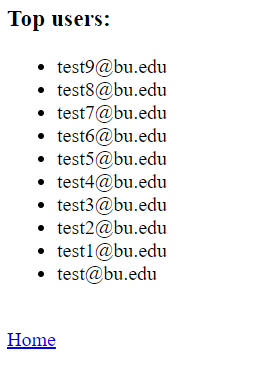
*(After)*



*(User table)*



**User activity.** To motivate users in using the site we'd like to identify the ones who make the largest contribution and list them on the site. The top 10 users should be reported.



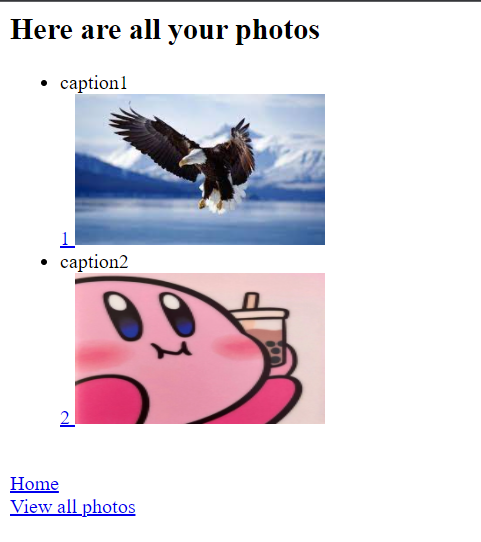
**Album and Photo Management**

**Photo and album browsing.** Every visitor to the site, registered or not, should be allowed to browse photos. In this project we will assume that all photos and albums are made public by their authors.

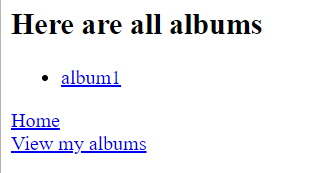
*(Browse all photos)*



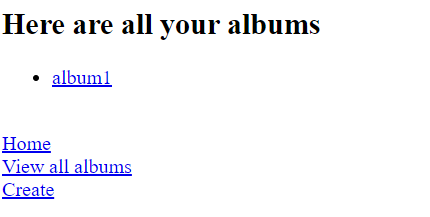
*(View my photos)*



*(Browse albums)*

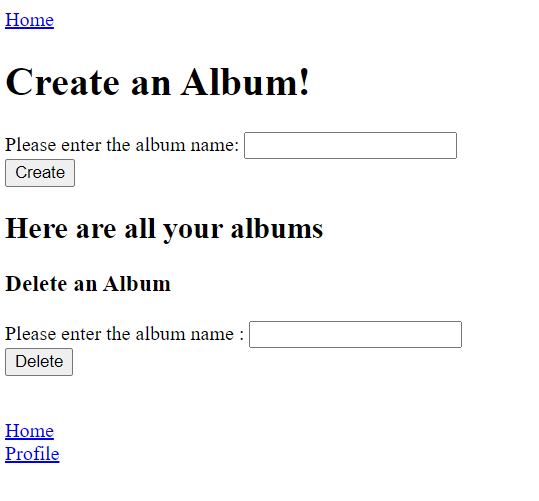


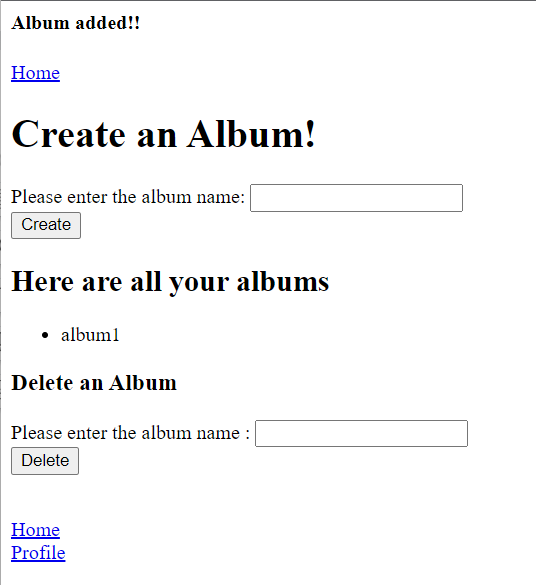
*(Browse my albums)*



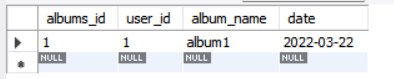
**Photo and album creating.** After registration, users can start creating albums and uploading photos. The relevant fields are described above. Users should also be able to delete both albums and photos. If a non-empty album is deleted, its photos should also be purged. Users should only be allowed to modify and delete albums and photos which they own.

*(Create albums)*

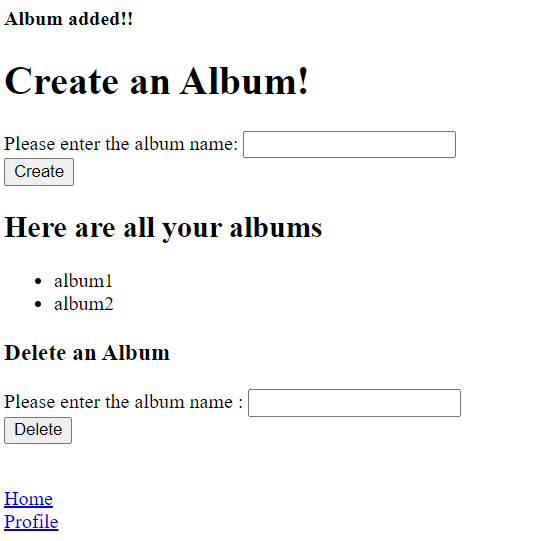


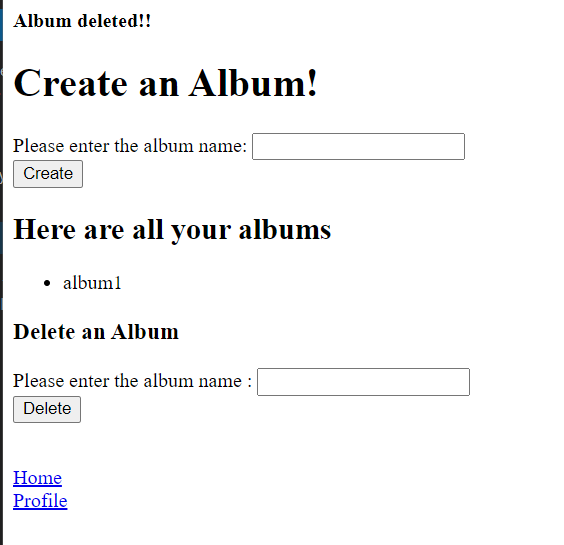


*(User table)*



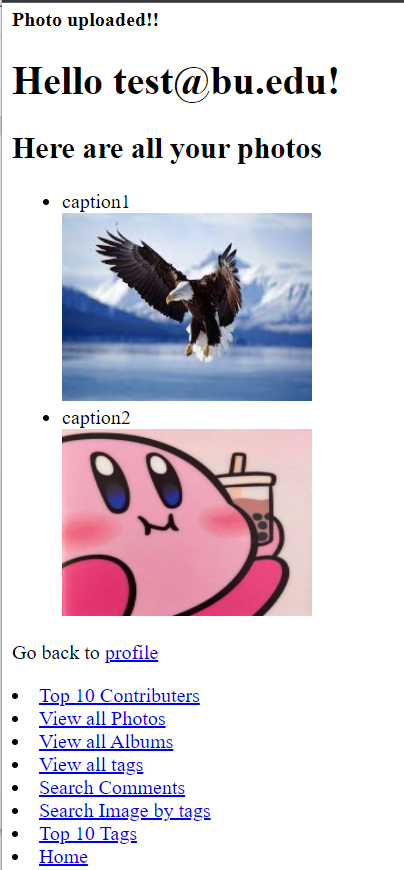
*(Delete album message - before & after)*





*(upload photo)*

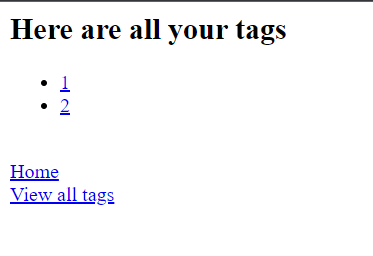




**Tag Management**

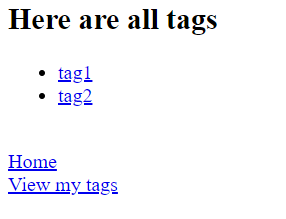
**Viewing your photos by tag name.** Tags provide a way to categorize photos and each photo can have any number of tags. You may think of the tags as virtual albums. For example, suppose that a user has two distinct albums each of which contains a photo with the tag 'friends'. The means should be provided to view the photos owned by the user in the virtual album (tag) 'friends'. One possible user interface design for this functionality is to present tags as hyperlinks. When a tag is clicked the photos tagged with it are listed.

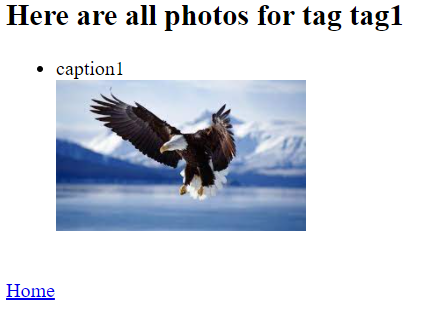
*(view my photos by tag name)*

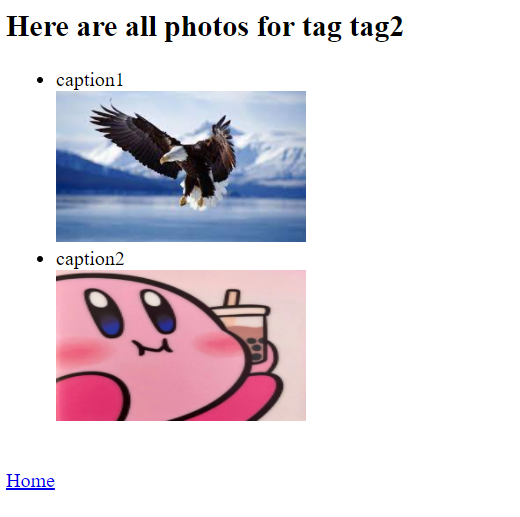


**Viewing all photos by tag name.** Furthermore, the system should allow users to view all photos that contain a certain tag, i.e., not only the ones they have uploaded but also photos that belong to other users.

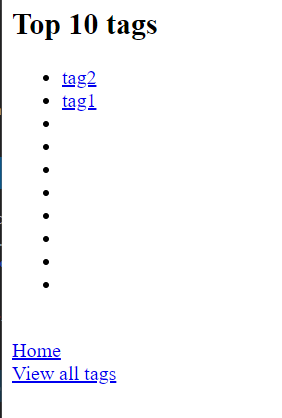
*(view all photos by tags name)*







**Viewing the most popular tags.** A function should be provided that lists the most popular tags, i.e., the tags that are associated with the most photos. Again, tags should be clickable so that when a user clicks one of them all photos tagged with it are listed.



**Photo search.** The functionality should be provided so that both visitors and registered users can search through the photos by specifying conjunctive tag queries. For example, a visitor could enter the words "friends boston" in a text field, click the search button and be presented with all photos that contain both the tag "friends" and the tag "boston".





**Comments**

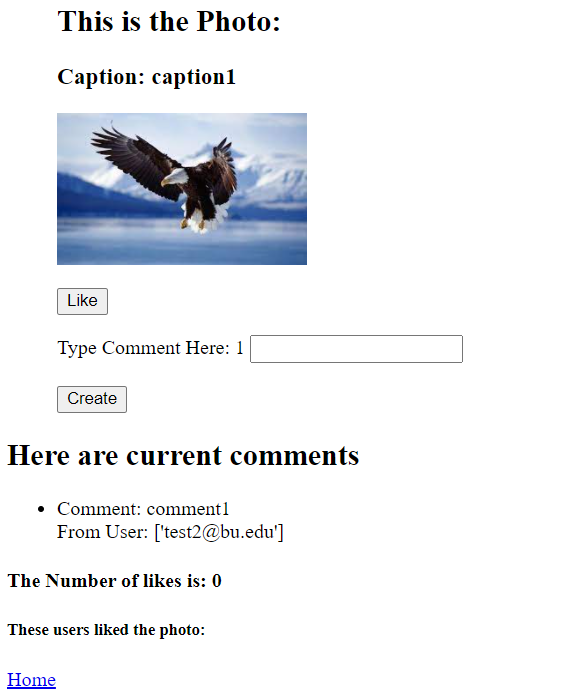
**Leaving comments.** Registered users can leave comments on photos. Users cannot leave comments on their own photos.

*(For registered users…)*

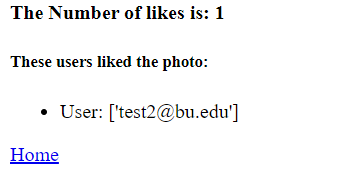
*(cannot leave comments on own photos)*



*(else)*



**Like functionality.** We want to add a **Like** functionality. If a user likes a photo, she should be able to add a like to the photo. Also, any user should be able to see how many likes a photo has and the names of the users who liked this photo.

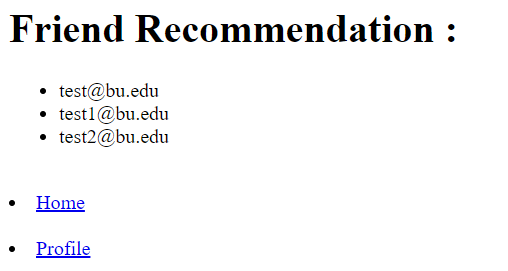


**Search on comments.** In this feature, we implement a search function based on the comments. The user can specify a text query (one or more words) and the system should find the users that have created comments that **exactly** match the query text.



**Recommendations**

**Friend recommendation.** We want to recommend possible newfriends to a user.



**'You-may-also-like' functionality.**





**----------------------------------------------------------------------------------------------------**

**SQL schema**

CREATE DATABASE IF NOT EXISTS photoshare;

USE photoshare;

CREATE TABLE Users(

user\_id INTEGER AUTO\_INCREMENT,

email VARCHAR(255) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL,

first\_name VARCHAR(255) NOT NULL,

last\_name VARCHAR(255) NOT NULL,

birthday DATE,

hometown VARCHAR(255),

gender VARCHAR(255),

PRIMARY KEY (user\_id));

CREATE TABLE Friends(

user\_id INTEGER,

friend\_id INTEGER,

PRIMARY KEY (user\_id, friend\_id),

FOREIGN KEY (user\_id) REFERENCES Users(user\_id),

FOREIGN KEY (friend\_id) REFERENCES Users(user\_id));

CREATE TABLE Albums\_have(

albums\_id INTEGER AUTO\_INCREMENT,

user\_id INTEGER NOT NULL,

album\_name VARCHAR(255) UNIQUE,

date DATE,

PRIMARY KEY (albums\_id),

FOREIGN KEY (user\_id) REFERENCES Users(user\_id) ON DELETE NO ACTION);

CREATE TABLE Tags(

tag\_id INTEGER AUTO\_INCREMENT,

word VARCHAR(255) UNIQUE,

PRIMARY KEY (tag\_id)

);

CREATE TABLE Photos(

photo\_id INTEGER AUTO\_INCREMENT,

albums\_id INTEGER NOT NULL,

album\_name VARCHAR(255),

user\_id INTEGER NOT NULL,

caption VARCHAR(255),

data LONGBLOB,

PRIMARY KEY (photo\_id),

FOREIGN KEY (user\_id) REFERENCES Users (user\_id),

FOREIGN KEY (albums\_id) REFERENCES Albums\_have (albums\_id) ON DELETE CASCADE);

CREATE TABLE Tagged(

photo\_id INTEGER,

tag\_id INTEGER,

PRIMARY KEY (photo\_id, tag\_id),

FOREIGN KEY(photo\_id) REFERENCES Photos (photo\_id) ON DELETE CASCADE,

FOREIGN KEY(tag\_id) REFERENCES Tags (tag\_id));

CREATE TABLE Comments(

comment\_id INTEGER AUTO\_INCREMENT,

text VARCHAR (255),

date DATE,

user\_id INTEGER NOT NULL,

photo\_id INTEGER NOT NULL,

PRIMARY KEY (comment\_id),

FOREIGN KEY (user\_id) REFERENCES Users (user\_id),

FOREIGN KEY (photo\_id) REFERENCES Photos (photo\_id) ON DELETE CASCADE );

CREATE TABLE Likes(

user\_id INTEGER,

photo\_id INTEGER,

PRIMARY KEY (photo\_id, user\_id),

FOREIGN KEY (user\_id) REFERENCES Users (user\_id),

FOREIGN KEY (photo\_id) REFERENCES Photos (photo\_id) ON DELETE CASCADE);

**----------------------------------------------------------------------------------------------------  
Assumptions**

‘Friend recommendation’ gives all recommend users, including yourself.

Unregistered user cannot like or comment a photo. He/she can only review photos and albums.

A user cannot like or comment his/her own photo.

Album’s name is unique.