

CS460 Project Assignment 1

PhotoShare: An on-line photo social network system

Database Design 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.

Welelcome to Photoshare!

Go back to [profile](#)

- [Top 10 Contributors](#)
- [View all Photos](#)
- [View all Albums](#)
- [View all tags](#)
- [Search Comments](#)
- [Search Image by tags](#)
- [Top 10 Tags](#)
- [Home](#)

[\(main page\)](#)

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.

Enter your personal information

Enter your Email (required):	<input type="text"/>
Enter Password (required):	<input type="password"/>
Enter First name (required):	<input type="text"/>
Enter Last name (required):	<input type="text"/>
Enter your Birthday (required):	<input type="text" value="年 / 月 / 日"/> 
Enter Hometown (optional):	<input type="text"/>
Enter Gender (optional):	<input type="text"/>
<input type="button" value="Create"/>	

[Home](#)

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

Email already in use!

Enter your personal information

Enter your Email (required):

Enter Password (required):

Enter First name (required):

Enter Last name (required):

Enter your Birthday (required):

Enter Hometown (optional):

Enter Gender (optional):

[Home](#)

(Otherwise, message 'Account created')

Account Created!!

Welcome test@bu.edu!

- [Friends](#)
- [Albums](#)
- [Upload photos](#)
- [View My Albums](#)
- [View My Photos](#)
- [View My tags](#)
- [You Might Also Like:](#)
- [Logout](#)
- [Home](#)

(User table)

	user_id	email	password	first_name	last_name	birthday	hometown	gender
▶	1	test@bu.edu	test	None	None	2022-01-04	Allston	female
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

(Guest mode page)

[login](#)

[Make an account](#)

- [Home](#)

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)

Add Friends

Enter Friend's Email:

Add

Friend list:

(After)

Add Friends

Enter Friend's Email:

Add

Friend list:

- ['test@bu.edu']

(User table)

	user_id	friend_id
▶	2	1
•	NULL	NULL

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.

Top users:

- test9@bu.edu
- test8@bu.edu
- test7@bu.edu
- test6@bu.edu
- test5@bu.edu
- test4@bu.edu
- test3@bu.edu
- test2@bu.edu
- test1@bu.edu
- test@bu.edu

[Home](#)

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)

Here are all photos

- caption1



1

- caption2



2

[Home](#)

[View my photos](#)

(View my photos)

Here are all your photos

- caption1



1

- caption2



2

[Home](#)

[View all photos](#)

(Browse albums)

Here are all albums

- [album1](#)

[Home](#)

[View my albums](#)

(Browse my albums)

Here are all your albums

- [album1](#)

[Home](#)

[View all albums](#)

[Create](#)

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)

[Home](#)

Create an Album!

Please enter the album name:

Create

Here are all your albums

Delete an Album

Please enter the album name :

Delete

[Home](#)

[Profile](#)

Album added!!

[Home](#)

Create an Album!

Please enter the album name:

Create

Here are all your albums

- album1

Delete an Album

Please enter the album name :

Delete

[Home](#)

[Profile](#)

(User table)

	albums_id	user_id	album_name	date
▶	1	1	album1	2022-03-22
*	NULL	NULL	NULL	NULL

(Delete album message - before & after)

Album added!!

Create an Album!

Please enter the album name:

Create

Here are all your albums

- album1
- album2

Delete an Album

Please enter the album name :

Delete

[Home](#)

[Profile](#)

Album deleted!!

Create an Album!

Please enter the album name:

Create

Here are all your albums

- album1

Delete an Album

Please enter the album name :

Delete

[Home](#)

[Profile](#)

(upload photo)

Upload a photo to Photoshare

Select photo (required): 未選擇任何檔案

Enter caption (required):

Enter tags (optional):

Here are your albums

Select Album (required):

[Home](#)

[Profile](#)

Photo uploaded!!

Hello test@bu.edu!

Here are all your photos

- caption1



- caption2



Go back to [profile](#)

- [Top 10 Contributors](#)
- [View all Photos](#)
- [View all Albums](#)
- [View all tags](#)
- [Search Comments](#)
- [Search Image by tags](#)
- [Top 10 Tags](#)
- [Home](#)

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)

Here are all your tags

- [1](#)
- [2](#)

[Home](#)

[View all tags](#)

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)

Here are all tags

- [tag1](#)
- [tag2](#)

[Home](#)

[View my tags](#)

Here are all photos for tag tag1

- caption1



[Home](#)

Here are all photos for tag tag2

- caption1



- caption2



[Home](#)

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.

Top 10 tags

- [tag2](#)
- [tag1](#)
-
-
-
-
-
-
-
-

[Home](#)

[View all tags](#)

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".

Search photo by tag name

Enter Tag:*

[Home](#)

[Back](#)

Here are all photos for tag ['tag1', 'tag2']

- caption1



[Home](#)

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)

It's your own photo!

[Home](#)

[All Photos](#)

[My Photos](#)

(else)

This is the Photo:

Caption: caption1



Like

Type Comment Here: 1

Create

Here are current comments

- Comment: comment1
From User: ['test2@bu.edu']

The Number of likes is: 0

These users liked the photo:

[Home](#)

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.

The Number of likes is: 1

These users liked the photo:

- User: ['test2@bu.edu']

[Home](#)

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.

Search Comments

Enter comment:

Comments:

- Comment: comment1
From User: 3

[Home](#)

Recommendations

Friend recommendation. We want to recommend possible new friends to a user.

Friend Recommendation :

- test@bu.edu
- test1@bu.edu
- test2@bu.edu

- [Home](#)
- [Profile](#)

'You-may-also-like' functionality.

- [You Might Also Like:](#)

You might like these photos

[Home](#)

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.