CS460 Project Assignment 1

PhotoShare: An online photo social network system

(To be done in groups of two.)

Deadlines

Database design report due: Friday, February 10, at 11:59PM.

Final report and implementation due: Friday, March 3, at 11:59PM.

Skeleton code is provided, you can find it on Piazza resources. Follow the README (in the skeleton folder) to get the code running.

Purpose of the Project

In this project, you will design, implement, and document a database system for a web-based photo sharing application. You also need to provide the web-based interface to the database. The final system should be functional and will be similar to Facebook/Flickr etc.

To be done in groups of two. Please find your partner asap.

Overview of Project Phases

You will be provided with the code base for this project, and only asked to fill in a few pieces. We have divided the project into 2 phases. You will be given detailed instructions and examples of all the utilities required for each phase.

Phase I – Database Design (due Friday, February 10, at 11:59PM)

For Phase I, you need to design and create a database for the application, based on the client's requirements for the app. This includes creating tables to store the data and designing rules for how these tables relate to each other, so that the data is organized in a meaningful way. In your Phase I report, you will turn in an E-R diagram of your database design, and the translation to the relational schema (Create Table SQL statements). Also, you need to provide all the constraints that you used in the database design (e.g., ASSERTION clauses). Please read the client's requirements carefully (the Data and Use Cases sections).

Phase II - App Implementation (due Friday, March 3, at 11:59PM)

In this phase, you need to implement a web interface to your database. The web interface should support all the use cases required by the client. Eventually, you will create a functionable photo sharing web app.

Data

The system should manage 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 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 share 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 "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.

Use Cases

The following interactions with the app should be 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. Other additional information about a user is optional.

Adding and listing Friends. You should allow a user to add a new friend to the friend list. For simplicity, you do not need to verify the friendship relationship, i.e., the user does not need the new friend’s confirmation to add her to the friend list. Also, you should allow the user to search for other users in the system (to find new friends to add). Finally, you should allow a user to list his/her friends.

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. We'll measure the contribution of a user (contribution score) as the number of photos they have uploaded plus the number of comments they have left for photos belonging to other users. 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 owners.

Photo and album creating. After registration, users can start creating albums and uploading photos. 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/photos owned by themselves.

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'. There should be a way for the user to view all the photos having the 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.

Viewing all photos by tag name. Furthermore, the app should allow users to view all photos that share the same tag, i.e., not only the ones they have uploaded but also photos that belong to other users. This functionality is similar to the one above and hence they could be presented together. One possible representation is a switch between "View all photos / View my photos". You can toggle the switch to jump from one view to the other.

Viewing the most popular tags. A function should be provided to list the most popular three tags, i.e., the three tags that are associated with the largest number of photos. The tree tags should be presented in descending order.

Photo search. A function should be provided for both visitors and registered users so that photos can be searched by specifying conjunctive tag queries. For example, a visitor could enter the words "friends boston" in an input box, click the search button, and be presented with all photos that contain both the tag "friends" and the tag "boston".

Comments

Leaving comments. Both registered and unregistered users (visitors) can leave comments on photos. Users cannot leave comments on their own photos. If a registered user leaves a comment, then this counts towards his/her contribution score as described above.

Like. We want to add a Like function. A user should be able to add a like to a 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. You want to implement a search function for the comments. A user can specify a text query (consisting of one or more words) and the app should find the users that have created comments that **exactly** match the query text. The system should return the names of these users ordered by the number of comments that match the query in descending order.

Recommendations

Friend recommendation. We want to recommend possible **new** friends to a user. One simple approach to recommend new friends to a user *A* is to find all friends of *A* and find their common friends. That approach is called "friends-of-friends" and is used in some systems as the first step to make friends recommendation. Sort the recommendations based on the number of times each recommended friend appears in the list of friends of friends.

'You-may-also-like'. Given the types of photos uploaded by a user, we'd like to make some recommendations to the user about other photos he/she may like. To achieve that you can do the following: take the three most frequently used tags among the user's photos. Perform a disjunctive search through all the photos for these three tags. A photo associated with all three tags should be ranked higher than another photo that contains only two tags, and so on. Between two photos that contain the same number of matched tags, the one that is more concise, i.e., the one that has fewer associated tags, has a higher rank.

Rules of the Project

Implementation Tools

We will provide you with all the tools you need to complete this project. All queries to the database will be made via SQL statements. For this project, you are expected to be familiar with Python and SQL. Everything else will be explained or automated for you. The project is to be done in groups of two. You are allowed to talk with each other about the project, but you are not allowed to share code or queries.

Deliverables

You should provide two deliverables for this project. The first one is the database design and the second one is the final implementation of the app.

Here are more details:

- Friday, February 10, at 11:59PM: You need to provide a report with the E/R diagram, the relational schema (Create Table SQL statements), the assumptions that you make, and the integrity constraints. You can submit the report via Gradescope.
- Friday, March 3, at 11:59PM: The final report should include the final schema, additional assumptions that you make during the implementation, and the limitations of your system (what functions are implemented and what are not). You need to submit to Gradescope the following:

    1. The final report
    2. A zip file with all your code.