CS460 Project Assignment

PhotoShare: A simple on-line photo social network system

Deadlines

Database design report: Saturday, June 11, at 11.59PM.

Final report and implementation: Tuesday, June 28, 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 social sharing application. You need also to provide the web-based interface to the database. The final system should be functional and will be similar to Flickr!

To be done in groups of two. Please find your partner asap. Don’t forget to add your teamwork when submit via Gradescope.

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 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 tag "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 interaction with the system 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. The other additional information about each user is optional.

Adding and Listing Friends. You should allow a user to add a new friend on the friend list. For simplicity, you do not have to verify the friendship relationship. Also, you must 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 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 authors.

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 an 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.

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 them 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 virtual album created if we select all the photos owned by the user that contain 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 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. Here you have to consider *all* photos in the database, not only yours.Again, tags should be clickable so that when a user clicks one of them all photos tagged with this tag 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. Both registered and anonymous users can leave comments. Users cannot leave comments for their own photos. If registered user leaves a comment then this counts towards their contribution score as described above.

Like functionality. We want to add a **Like** functionality. If a user likes a photo, should be able to add a like to the photo. Also, we must be able to see how many likes a photo has and the users that liked this photo.

Recommendations

'You-may-also-like' functionality. Given the type of photos uploaded by a user we'd like to make some recommendations to them about other photos they may like. To achieve that do the following: Take the five most commonly used tags among the user's photos. Perform a disjunctive search through all the photos for these five tags. That is, find all the photos that contain at least one of the five tags. A photo that contains all five tags should be ranked higher than one that contained four of the tags and so on. Between two photos that contain the same number of matched tags prefer the one that is more concise, i.e., the one that has fewer tags over all.

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 individually. You are allowed to talk with each other about the project but you are not allowed to share code or queries. If you have questions about the project you should ask the Instructor first.

Overview of Project Phases

You will be provided with the infrastructure 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. However, you need to submit everything together after completing all the phases.

Phase I – Database Design

Initially, you must design and create a database for the application. This includes creating tables to store this data, and designing rules for how these tables relate to each other so that the data they store can be combined in meaningful ways. As part of the report for this phase, you will turn in an E-R diagram of your database design, and the translation to the relational schema (Create Table SQL command). Also, you have to provide all the constraints and the SQL code that you used to enforce them (CHECK or ASSERTION clauses).

Phase II - API Implementation and Web Site

In this phase, you should write the SQL queries and some Python code that implements an interface to your database and executes the queries. This part will be used in the next phase for the web interface. Finally, you will create a web site for your database and add the photo sharing functionality.

Deliverable

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

Here are more details:

- Saturday, June 11, 2022 at 11.59PM: You need to provide a report with the E/R diagram, the relational schema, the assumptions that you make, and the integrity constraints. Submit the report by Gradescope.
- Tuesday, June 28, 2022 at 11:59PM. The final report with the code and the additional assumptions that you made and the limitations of your system. In particular, you should submit the following by Gradescope:
    1. The final report that should contain the final schema and any additional assumptions and constraints you made.
    2. A zip file with the photoshare directory (all your code). Alternatively, you can submit the photoshare directory.