NYU Tandon School of Engineering
Computer Science and Engineering
CS-UY 3083, Introduction to Database Systems, Spring 2020
Prof Frankl

COURSE PROJECT

Project Overview

The course project this semester is **Finstagram**, a web application for sharing photos. Finstagram gives users more privacy than many photo sharing sites by giving them more detailed control over who can see which photos they post. The focus of the project will be on storing data about who posted which photos and who has permission to view them, tag others in them, see who's tagged in what, etc.

Users will be able to log in, post photos, view *some of* the photos posted by people they are following, detailed below; and take other actions such as tagging, liking, and/or commenting on photos they can view .. In part 1 of the project, you will design an ER diagram for the database. In part 2, you will convert *my E-R diagram*, which I will post after Part 1 is due, to a relational schema and write table definitions in SQL. Part 3 is a milestone to guarantee that you're making progress: *using my table definitions*, which I will post after Part 2 is due, you will write and test some of your SQL queries and about ¼ of your application source code. Part 4 will be the most work: you will revise your work from part 3, if necessary, and write and test the rest of the application code for the system. A detailed specification will be provided shortly after part 2 is due. Near the end of the semester you will schedule a demo/test session in which your Finstagram application will be evaluated, using data and tests we will provide. Of course, you should also test your code thoroughly as you're developing it.

Partners, etc

You may work alone or in a **team of up to 3 people** (i.e with zero, one, or two others.) For parts 3 and 4 of the project, there will be some extra requirements for multi-person teams, with the number of extra requirements increasing as team size increases. There will be a core of features that everyone must implement (such as posting a photo, seeing recent photos that are visible to the user who's logged in, etc.) and an additional two features per person for multi-person teams.. You may work on part 1 alone then team up for later parts. There will be some deadline set for team formation, around when part 2 is due. Note that each team member is expected to contribute roughly equally and each team member is responsible for understanding the entire system. For multi-person teams, part of your grade will be a team grade based on the core features and part will be an individual grade based on your extra features.

The total project grade will be 25% of your course grade. Part 1 counts for about 20% of the project grade. Part 2 counts for about 15% of the project grade. Part 3 counts for about 10% of the project grade, and Part 4 counts for the remainder. There may also be a quiz or exam question(s) based on the project.

PART 1: Due Wed Feb 26 via GradeScope

Previously posted; Solution is available in Resources → Project. Use that solution for Part 2.

PART 2 DUE 11:59pm Friday March 13

Use the ER diagram posted (my solution to Part 1):

A. Using the rules we studied for converting an ER diagram to a relational database schema, draw a relational schema diagram. Remember to underline primary keys and use arrows from referencing to referenced attributes to denote foreign keys. Note that you will need to rename a few attributes when the basic rules would result in two attributes with the same name in the same table. In particular, *username* will end up occurring in several different contexts. Choose meaningful names.

B. Write SQL CREATE TABLE statements for each table. Use INT for Photo IDs and for ratings; INT or Boolean for flags (such as followStatus, etc); DATETIME or something similar for dates, timestamps, etc (see https://dev.mysql.com/doc/refman/8.0/en/date-and-time-types.html); and VARCHAR for the other attributes. Remember to include PRIMARY KEY and FOREIGN KEY constraints.

C. Write a query to find the IDs of Photos that are SharedWith any FriendGroup to which the user with username "Ann" belongs. This should involve joining several tables with appropriate join conditions.

D. Read the project descriptions for parts 3 and 4 and start thinking about which additional features you will do. If you want to propose any features other than the suggested ones, write a brief description of those features and submit them. (The due date for this part will be later, but the sooner you hand them in the sooner you'll be able to get feedback on them.)

Part 2: What / How to hand in:

A. Via GradeScope Assignment Project Part 2ABC: A pdf file that includes
    a. The schema diagram
    b. The CREATE TABLE statements
    c. The query
B. Via GradeScope Assignment Project 2BC: A plain text file that includes all of your CREATE TABLE statements (B) and the query (C). These should compile successfully and you are advised to test the query.
C. Included with B
D. (optional) If you are proposing additional features that are not on my list of suggestions

DRAFT of PARTs 3 and 4

In parts 3 and 4 of the project, you will use the table definitions I will post (solution to part 2) to implement application code for Finstagram as a web-based application. You may use Python, PHP, Java, node.js, Go, Ruby, or C#. If you'd like to use some other language, check with me by 11/10/19 . *You must use prepared statements if your application language supports them.*

I will supply Python code for a sample application that includes most of the constructs you'll need in order to do this assignment, along with videos that explain the code in detail. If you do not have prior experience with web and/or database application programming, you should study these once I've covered the requisite background in class (week before Spring Break).

Part 3 (Due April 3, 2020) is a milestone in which you must show that you've developed key queries and written and tested some of your code. See details, below.

Part 4 is the completed project, due on April 26, 2020 **(small penalties for a few days after that; increasing penalties thereafter)**. You will hand in your code and a short write up (details coming soon). You will also schedule a demo session in which one of the TAs will run through a series of tests with you.

Your Finstagram implementation should allow a user to log in; view photos that she has access to and data about those photos; post photos and designate who can view them; make, accept, or reject "follows" requests; and create friendGroups. In addition, you will propose and add some other features, as described below.
Assume each user has already registered and created a password, which is stored as an SHA-2 hash. (We will provide Python/Flask code to manage user registration and login).

A photo ***p is visible to Person U*** if and only if

- AllFollowers == True for p and U has been accepted as a follower by p's photoPoster, OR
- p is shared with a FriendGroup to which U belongs.

Remember that a FriendGroup is identified by its groupName *along with* the groupOwner (the username of the owner of the group).

(We will assume that in the initial database, the owner of each FriendGroup is a member of that FriendGroup. When modifying the database to create new friendGroups, Finstagram should maintain that, by adding the user as a member of each FriendGroup she creates. Thus, a user will be able to view photos she has shared with any of the FriendGroups she's created.)

Finstagram should support the following use cases:

Login: The user enters her username and password. Finstagram will add “salt” to the password, hash it, and check whether the hash of the password matches the stored password for that username. If so, it initiates a session, storing the username and any other relevant data in session variables, then goes to the home page (or provides some mechanism for the user to select her next action.) If the password does not match the stored password for that username (or no such user exists), Finstagram informs the user that the login failed and does not initiate the session. **We will supply Python/Flask code for this. If you’re using a different implementation language, you’ll need to write this yourself.**

The remaining use cases require the user to be logged in.

Features 1 — 5 (view visible photos, view further photo info, post a photo, manage followers, and create friend group) are Mandatory.

1) **View visible photos:** Finstagram shows the user the photoID of each photo that is visible to her, arranged in reverse chronological order.

2) **View further photo info:**
   Display the following for visible photos (You may include this with the results of “view visible photos” or you may supply a different way for users to see this additional info, such as clicking on additional links).
   a) Display the photo or include a link to display it. (If you have trouble actually displaying the photo, just display the photo’s pID for close to total credit for this part).
   b) The firstName and lastName of the photoPoster
   c) The postingDate,
   d) the usernames, first names and last names of people who have been tagged in the photo (taggees), provided that they have accepted the tags (Tag.acceptedTag == true)
   e) The usernames of people who have ReactedTo the photo and the emoji and/or comment they gave it

3) **Post a photo:**
   Finstagram prompts the User to enter
   a) the location of a photo on the client computer,
   b) a designation of whether the photo is visible to all followers (allFollowers == true) or only to members of designated FriendGroups (allFollowers == false).

   In response to the data submitted by the user, Finstagram
   c) inserts data about the photo (including current time*, and the current user as photoPoster) and the value of allFollowers into the Photo table. *Finstagram can find the current time with an SQL function or a function in the host language.
   d) either copies the photo to a dedicated folder with file name that includes the pID and stores this location as the filePath or it stores the photo as a BLOB datatype. (More details will be provided on how to do this).

    e) gives the user a way to designate FriendGroups that the user belongs to with which the Photo is shared.

4) **Manage Follows:**
    a) User enters the username of someone they want to follow. Finstagram adds an appropriate tuple to Follow, with acceptedFollow == False.
    b) Finstagram displays list of requests others have made to follow the current user and the user has the opportunity to accept, by setting acceptFollow to True or to decline by deleting the request from the Follow table.
5) **Add friendGroup:**
    a) User provides a name for the friendGroup.
    b) Finstagram creates the group with current user as the groupOwner, provided that they don't already own a group with this name. Finstagram gives a meaningful error message if the current user already has a group with this name.
    c) If the group is created successfully, Finstagram adds the current user as a member of the group.

Extra Features:

You must Add **two extra features per person:**

- Working alone: 2 extra features required
- Two person team: 4 extra features required
- Three person team: 6 extra features required

These individual features should be designed and implemented by an individual team member who will get an individual grade for this work. You may help your teammates understand what they should do and you should review and test their code, but they should have primary responsibility for planning and implementing their features. Each additional feature must involve interactions with the database and must be relevant to this particular project. *Generic features like a registration page are not acceptable*. For each feature you must write a description including:

d) The name of the team member who is primarily in charge of implementing this feature
e) The queries (and any other SQL statements) used in your implementation of the feature
f) A clear indication of where to find the application source code for the feature. Include plenty of comments and/or a few sentences explaining the code.
g) One or more screenshots demonstrating the feature, showing how it appears in the browser; Also show the relevant data that's in the database when you execute this demonstration (before and after if the feature changes the data), either as screenshots or as text.
h) For features other than those I suggested, also include:
    i) What the feature is (in the style of the requirements above)
    ii) A sentence or two on why this is a good feature to add to Finstagram

iii) A clear explanation of any changes to the database schema that are needed (including additional tables, additional attributes, or additional constraints)

Note: *If your extra features require extensive modification to the database schema, please develop and test them as a separate branch of your project, so that you'll still be able to demonstrate the basic features with data we will supply.*

Here are a few ideas for extra use features:

6) **Optional (this can be TWO of your extra features:) Manage tags:**
   a) Current user, whom we'll call x, selects a photo that is visible to her and proposes to tag it with username y
      (1) If the user is self-tagging (y == x), Finstagram adds a row to the Tag table:
          (x, photoID, true)
      (2) else if the photo is visible to y, Finstagram adds a row to the Tag table:
          (y, photoID, false)
      (3) else if photo is not visible to y, Finstagram doesn't change the tag table and prints some message saying that she cannot propose this tag.
   b) Finstagram shows the user relevant data about photos that have proposed tags of this user (i.e. user's username is Tag.username and acceptedTag is false.) User can choose to accept a tag (change acceptedTag to true), decline a tag (remove the tag from Tag table), or not make a decision (leave the proposed tag in the table with acceptedTag == false.)

7) **Optional (this can be one of your extra features:) React to a photo:** Users can add comments and/or emojis to photos that are visible to them.
8) **Optional (this can be one of your extra features:) Unfollow:** Think about what should be done when someone is unfollowed, including some reasonable approach to tags that they posted by virtue of being a follower. Write a short summary of how you're handling this situation. Implement it and test it.
9) **Optional (this can be one of your extra features:) Search by tag:**Search for photos that are visible to the user and that tag some particular person (the user or someone else)
10) **Optional (this can be one of your extra features:) Search by poster:**Search for photos that are visible to the user and that were posted by some particular person (the user or someone else)
11) **Optional (this can be one of your extra features:) Add friend:** User selects an existing FriendGroup that they own and provides username of someone she'd like to add to the group. Finstagram checks whether there is exactly one person with that name and updates the Belong table to indicate that the selected person is now in the FriendGroup.

Unusual situations, such as the person already being in the group, should be handled gracefully.

12) **Optional (these can be some of your extra features:)** Analytics … find photos with highest average “like” ratings, or liked by many people, or people who’ve had most photos liked, or some such.
13) Be creative. Propose something in part 2 D, check that it’s approved, then implement it.

DELIVERABLES:

Part 3, due April 3:
You’ll hand in your code for Parts 3 and 4 by putting it in a GitHub repository and sharing it with CS3083-TA username that we’ll supply later. Aside from sharing the repo with your team and the course staff, it should be private. For Part 3, hand in the following:

1) In your GitHub repo:
    a) Code for at least two of the required features
2) Via GradeScope:
    a) Link to your GitHub repository.
    b) A few screenshots showing the browser as you execute the code from part the required features you’ve implemented, executing on some test data
    c) Plan for which optional features you will implement and (if you are working with a team) who will be responsible for writing them and who will help test them. If any of your optional features require you to modify the table definitions, please indicate modifications.

Part 4:
You will hand in the rest of your source code and a brief summary (details to be posted later) and will schedule a session in which you’ll give a demonstration and we’ll run some tests. More details later.