NYU Tandon School of Engineering Nov. 11, 2016
Computer Science and Engineering Department
CS-UY 3083
Prof. Phyllis Frankl

Project Overview

The course project this semester is FindFolks, a system to organize groups of people with common interests, announce events these groups are hosting, and gather feedback on the events, similar to Meetup. The main purpose of a group is to organize events for people with common interests, such as chess tournaments on the weekends, or hikes in the mountains..

Users of the database application program you will write (in part 3 of the project) will be able to search for groups and events of interest, create events, sign up for events, rate events, etc. 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 later) to a relational schema, write table definitions in SQL, write some queries, and prepare a project plan. Part 3 will be the most work: using **my** schema from part 2, you will revise your queries, if necessary, and write application code for the system. A detailed specification of several required features will be provided soon. In addition, you will be asked to propose and implement some additional features.

PART 1

Each registered FindFolks user (member) is identified by a unique username and has other attributes including password, name (first and last), email, and zip code.

Members can designate other members as friends. There is no limit on how many friends a member can have.

A group has a unique ID (g_ID), a name, and a short description and is usually focused on a particular interest or activity, for example "Brooklyn Ping Pong Maniacs'', "MySQL User Group'', or "NYC Chinese Cooking''. Any member can create a group; each group is created by exactly one member. Any member can join any group.

The main purpose of a group is to organize events. Each event has a unique ID (e_ID). and a title (e.g., "Friday Night Ping Pong''), a short description, a start and end time, (including the date). Only certain authorized group members can organize group events.

Each event takes place at a single location. A location has a location name (e.g. "Joe's pub") and zip code. Location names are unique within zip codes, but there may be different locations with the same name in different zip codes. Each location also has a street address, GPS co-ordinates, and a small description.

Members who want to attend an upcoming event can sign up, indicating that they plan to attend. Members can rate an event from one to five stars.

To help users find appropriate groups (and to help groups target potentially interested members), groups and members can specify any number of interests. An interest consists of a category and a keyword, for example (sports, ping pong) or (games, Brooklyn).

What You Should Do

1. Design an ER diagram for FindFolks. It should include entity representing Member, Group, Event, Location, and Interest (and maybe others, if needed). There should be several relationship sets, some of which have attributes.

   When you do this, think about: which information should be represented as attributes, which as entity sets or relationship sets? Are any of the entity sets weak entity sets? If so, what is the identifying strong entity set? What is the primary keys (or discriminant) of each entity set? What are the cardinality constraints on the relationship sets?
2. Additional requirement for teams with three members; (optional for others:) Think of at least one useful additional use for the system that requires additional data, describe it (at the level that users would understand), and indicate how it would change your ER model. It should be complex/interesting enough that it involves adding at least one more relationship set to the model.

Draw the ER diagrams **neatly**. You may draw it by hand or use a drawing tool.

You may work alone or with one or two partners. If you work with a team, you will be required to add a few extra features to the application, but the total amount of work per person will be less than if you work alone. Teams are encouraged to try pair programming, rather than simply dividing up the work.

Teams will be required to submit a work plan, indicating who will do what, and will be required to submit an evaluation at the end. Note that each team member is expected to contribute roughly equally to each aspect of the project and each team member is responsible for understanding the entire system. Normally all team members will receive the same grade, but I may deduct points from individuals who are not pulling their weight on a team.

If you are doing this project with a team, you must notify me by sending e-mail to the TAs by Oct 6.

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

PART 2

You must use the posted *Solution to Part 1* for part 2. First, compare your solution to the posted solution and think about what you did correctly, what you did incorrectly, and what aspects could have been modelled correctly either with your ER or with the one we've posted.

1. Following the techniques we studied, derive a relational schema diagram from the the posted Solution to Part 1 ER diagram. Remember to underline primary keys and use arrows from the referencing schema to the referenced schema to indicate foreign key constraints.
2. Write and execute SQL CREATE TABLE statements to create the tables. Choose reasonable types for the attributes. Most of them should be VARCHAR.
3. Write and execute INSERT statements to insert data representing:
   a. Two Groups, named "Brooklyn Bowling" and "Brooklyn Badminton"
   b. At least three members, one of whom belongs to and created both groups and is authorized to create events for those groups and two of whom each belong to one group; one of whom is authorized to create events for a group they didn't organize. (You can use names of people you know, famous people, or make something up).
   c. Two events created by each of the groups
   d. One mutual friendship and one non-mutual friendship
4. For one of the people you've inserted, Write and execute queries to
   a. list the group names of all groups the person belongs to
   b. list the event name, date, time, sponsoring group (name) and location of all events sponsored by groups to which the person belongs
   c. For one of the groups the person belongs to, check whether the person is authorized to create events for the group
5. Write and briefly explain the purpose of *n* additional queries that are relevant to FindFolks, where *n* is the number of members of your team.
6. <u>Additional requirement for teams with **three** members; (optional for others:)</u>
   a. add your additional entity sets, relationship sets, and/or attributes (from the additional requirements in your part 1 ER diagram) to the Solution to Part 1;
   b. create additional tables, accordingly;
   c. write and execute INSERT statements to insert several rows into each of those tables;
   d. and write at least one query relevant to the additional feature you proposed in Part 1 that uses the additional data. Include a brief explanation.

PART 3

In Part 3, you'll implement FindFolks as a web based application. **You must use the table definitions that are posted (derived from the E-R diagram that I posted)** unless you need to make some small additions/modifications to support your additional features. If you do modify the table definitions, you will be responsible for translating the test data we will provide so that it matches your table definitions.

REQUIRED Application Use Cases (aka features):

1. **View Public Info:** All users, whether logged in or not, can
   a. See the events that are occurring during the coming 3 days.
   b. select an interest and see list of groups that share that interest

2. **Login:** User enters their username, x, and password, y, via forms on login page. This data is sent as POST parameters to the login-authentication component, which checks whether there is a tuple in the Person table with username=x and the password = md5(y).
   a. If so, login is successful. A session is initiated with the member's username stored as a session variable. Optionally, you can store other session variables. Control is redirected to a component that displays the user's home page.
   b. If not, login is unsuccessful. A message is displayed indicating this to the user.

Note: In other words, members' passwords are stored as md5 hashes, not as plain text. This keeps the passwords more secure, in case someone is able to break into the system and see the passwords. You can perform the hash using MySQL's md5 function or a library provided with your host language.)

Once a user has logged in, FindFolks should **display** her **home page**. Also, after other actions or sequences of related actions, are executed, control will return to component that displays the home page. The home page should display

   c. Error message if the previous action was not successful,
   d. Some mechanism for the user to choose the use case she wants to execute. You may choose to provide links to other URLS that will present the interfaces for other use cases, or you may include those interfaces directly on the home page.
   e. Any other information you'd like to include. For example, you might want to include groups the user belongs to, events she's signed up for, etc. on the home page, or you may prefer to just show them when she does some of the following use cases.

After logging in successfully a user may do any of the following use cases:

3. **View My Upcoming Events:** Provide various ways for the user to see upcoming events for which she has signed up. The default should be showing events for the current day and the next three days. **Optionally** you may include a way for the

user to specify a range of dates, limit the display to events sponsored by particular groups, etc. In each case, the display should include the event_id, title, start_time, location_name, zipcode, and the name of sponsoring group.

4. **Sign up** for an event: User chooses an event and signs up for the event. You may find it easier to implement this along with a use case to search for events. If the user has previously signed up for the event no additional action is needed (but FindFolks should not crash).
5. **Search for events of interest:** Shows info (event_id, title, etc.) about upcoming events sponsored by groups that share an interest with this user. Optionally, you may limit the date range and/or provide other ways for the user to search for events that they might wish to attend. You might want to combine this with other related use cases.
6. **Create an event**: If the user is authorized to do so, he or she creates a new event for a group, providing all the needed data, via forms. The application should prevent unauthorized users from doing this action.
7. **Rate an event** that the user has signed up for and that has already take place.
    a. Ratings are between 0 and 5 stars
    b. Ratings cannot be done before the event starts
    c. If a user tries to rate an event that they didn't sign up for or that didn't already start FindFolks should display an error message.
8. **See average ratings** of all events (during last three days or, optionally, some specified date range) that were sponsored by groups to which this member belongs.
9. **See friends' events:** Displays a list of upcoming events for which any of the member's friends are signed up. (Here, the current user X's friends are people Y such that there is a tuple (friend_of, friend_to) = (X, Y) are in the friends relationship, i.e. people who have "friended" X.
10. **Logout:** The session is destroyed and a "goodbye" page or the login page is displayed

ADDITIONAL FEATURES

You must propose at least *2n* additional use cases, where n is the number of people on your team. In other words: if you're working alone, two more features; team of two, four more features; team of three, 6 more features. (Suggestions ... various ways to search for events of interest, search for groups of interest, join group, creating groups, granting ability to create events, make friends, etc.)

At least *n* of these must involve modifying inserting and/or updating rows in some table(s). If you have a team of three some of your additional features must involve the additional attributes or tables that you added in parts 1 and 2.

Additional Requirements:

You should implement FindFolks as a web-based application. If you want to use a DBMS other than MySQL, SQLserver, or Oracle or to use a programming language other than Python/Flask, Java/JDBC/Servlets, PHP, or C#, please check with me by November 17. You will need to bring the host computer to the demo/test session at the end of the semester or make the application available remotely over the web.

Enforcing complex constraints: Your FindFolks implementation should prevent users from doing actions they are not allowed to do. For example, FindFolks should prevent users who are not authorized to do so from organizing events for the group. This should be done by querying the database to check that the user belongs to the group, and is authorized by creator before allowing him to create/alter the event. You may also use the interface to help the user to avoid violating the constraints. For example, you could provide a list of groups and events for which the user is eligible and allow to make changes. However, you should not rely solely on client-side interactions to enforce the constraint, since a user could bypass the client-side interface and send malicious http requests.

When a user logs in, a session should be initiated; relevant session variables should be stored. When the member logs out, the session should be terminated. Each component executed after the login component should authenticate the session and retrieve the user's pid from a stored session variable. (If you're using Python/Flask, you can follow the model in the Flask examples presented to do this.)

You must use *prepared statements* if your programming language supports them. (This is the style used in Flask; if you're using PHP, use the MySQLi interface; if you're using Java/JDBC, use the PreparedStmt class.) If your programming language does not support prepared statements, Free form inputs (i.e., text entered through text boxes) that is incorporated into SQL statements should be validated or cleaned to prevent SQL injection.

You should take measures to prevent cross-site scripting vulnerabilities, such as passing any text that comes from users through htmlspecialchars or some such function, before incorporating it into the html that FindFolks produces.

The user interface should be usable, but it does not need to be fancy.

For testing/debugging you will probably find it useful to execute your SQL queries directly through PHPmyAdmin, before incorporating them in your application code.

Extra features: There are lots of opportunities to add functionality that makes FindFolks better. You may add more features (beyond the required *2n* extra features) for fun. However, your grade will be based almost entirely on our evaluation of the basic features (including those that you propose). This will be done largely by executing a series of tests. Please don't add any bells and whistles that interfere with testing the basic features.

What you should do

1. Study my solution to Part 2. Drop the tables you defined in Part 2 then install the ones given in my solution to Part 2. (If you defined extra tables or attributes, for additional features, merge them into my solutions to Part 2.)

2. Before you start coding, think about what each component will do. If there are commonalities among many of the use cases, think about how you will modularize your code.
3. For each component
    a. Using some sample data, write the queries executed by the component, and test them.
    b. Write the application code for the component.

4. Test the component with additional values, including values that are not valid input
5. Suggestion: Implement and test the components one at a time. When a component is ready, add links to it to your home page (or enhance the home page with the interface of the new component.) You will get partial credit if some of your features work, even if others have not been implemented.

PROGRESS REPORT A progress report will be due in about 2 weeks. In this report, you must propose the additional features that you plan to add and show code for use-cases. For team projects, you must demonstrate that each team member has written some of the code. You will hand this in electronically and I may also schedule brief meetings to discuss. Details TBD. **THIS IS A MANDATORY PART OF THE PROJECT AND WILL AFFECT YOUR GRADE.**

Complete Final Project Hand in instructions: You will hand in:

- Your source code. (Details about whether to zip it, etc, will be provided.)
- A list of the files in your application and what's in each file. (E.g. "homepage.phpscript to generate home page".)
- A separate file that lists all of the use cases and the queries executed by them (with brief explanation). This should be well organized and readable. It should be detailed enough to give readers a good idea of how your application works, without making them dig through all the code.
- Brief description of additional features you added.
- For team projects: A summary of who did what.
- **For team projects you will also hand in (individually, via another link to be provided) a brief self-assessment and an assessment of your teammates' work.**
- Shortly before the project is due, I'll ask you to sign up for a time slot to demonstrate your project to me or one of the TA's. This demo will mainly consist of running application on a bunch of test cases. We may also ask you to explain some of your code. No formal presentation will be expected.
- More detailed instructions on what to hand in, how to hand it in, and test data to load into your tables will be provided later.

The total project grade will be 25% of your course grade. Part 1 counts for about 15% of the project grade. Part 2 counts for about 10 to 15% of the project grade. Part 3 counts for about 70 to 75% of the project grade. There may also be a quiz or exam question(s) based on the project.

Instructions for working with a Team

You may work alone or with one or two teammates. You may work on part 1 alone then choose a teammate for parts 2 and 3, but you may not switch teammates. If you work with a teammate, you will be required to add some extra features to the application, but the total lines of code per person will be substantially less than if you work alone. Teams are encouraged to try pair programming, rather than simply dividing up the work.

Deadline for adding teammates for Part 3: Monday Nov 20. THIS IS A HARD DEADLINE. NO EXCEPTIONS.

Teams will be required to submit a work plan, indicating who will do what, and will be required to submit an evaluation at the end. Note that each teammate is expected to *contribute roughly equally to each aspect of the project and each teammate is responsible for understanding the entire system. Normally all team members will receive the same grade, but I may deduct points from individuals who are not pulling their weight on a team.*