NYU Tandon School of Engineering
Computer Science and Engineering
CS 6083 INET, Spring 2019

March 25, 2019

Project #1 (due April 14)

Project Description: For the class project you will have to design and implement a backend of a web based application that will use a database system to manage all the application data. Your application should include elements of social networks (users have friends, relationships, posts to other users, etc) , multimedia content (photos, videos, etc) and location services (maps). More specifically, your system should support the following three aspects. First, users who have signed up for a service should be able to post content, such as a profile, post entries (also called notes or postings), and some multi-media items such as photos, audio or video, which will be made accessible via a page on the site that is created for that user. Second, the users should be able to define relationships with other users that they know or that they consider their friends. Users should then be able to restrict content so that only they, or their direct friends, or friends of these friends, or everybody, can access their content. Third, other users or visitors should be able to browse and search for content, subject to the access restrictions selected by the users who posted the content. Users who visit another user's page on the site should also be able to leave greetings, comments, and tips about other places for similar items, and to ask recommendations for services and tools (e.g. best security conference), etc. Thus, the most important concepts in this project are users, posts, comments, activities, locations, and their various relationships.

In this first part of the project, you are asked to design a database backend for such a system, that is, a suitable relational schema with appropriate keys, constraints (and maybe views), plus a set of useful queries that should be created as stored procedures. You may use either your own database installation on your laptop (based on Windows, Linux or Mac), or on the internet, but no MS Access. Make sure to use a database system that supports text operators, such as "like" and "contains", since you should allow users to search the site and textual content by keywords.

Note that in the second part, to be handed out in several weeks and due at the end of the semester, you will have to extend this part to provide a basic suitable web-based interface. Thus, you cannot skip this project. Following are more details about the problem domain you are dealing with. Obviously, we will have to make some simplifying assumptions to the scenario to keep the project reasonable.

You are free to choose your own application scenario, below you will find a list of possible applications for users of specific social network application that you may decide to choose, or you can create your own application scenario. To learn about social networking sites, you may want to look at a few existing systems such as *Facebook*, *LinkedIn*, *FourSquare*, or *Pinterest*. There are many differences between these sites, so in the following we describe the minimum requirements for your system.

Problem Domain: We assume that in order to post or view content, a user has to sign up with the site, select a unique user name, and provide some information such as name, age, and city of residence. Once she has signed up, a user can post a profile, can like certain activities, and can post entries to a diary. A user should also be able to post multimedia content such as photos (or maybe videos) as part of a diary entry. You may want to use data types such as `blob` or `clob` to store multimedia or text in the database. Each diary entry should have a title (not necessarily unique), a time stamp when it was posted, and a body (the actual content of text and maybe multimedia). Users should also be able to comment on other users' diary entry, subject to suitable constraints (see below).

Another important concept is that of a location for an activity. Users can specify locations, for example a city they visited or a city for a conference they attended. A location might have a location ID, a name (e.g., "CSS'14 Venue"), and longitude and latitude coordinates. Of course, there are many popular locations that are jointly used by many different users (e.g., a popular city) but users can also add new locations of their own. Users should be able to like certain locations *for a particular activity*, and they should be able to connect a diary post to a particular location. (You may want to think about how you can use map interfaces in the second project to allow people to search and specify locations, but this is not needed for the first part.)

A user should also be able to identify other users as friends. Two users are friends only if each one has selected the other as a friend (thus, friendship is mutual). In practice, once one user selects another one as a friend, an email might be sent to the other user to notify him so that the user can decide whether to reciprocate. Alternatively, the next time the other user logs in, a message may list new people who have selected him as friend. You do not have to deal with the details of this at the moment. The important thing is that a friendship only becomes visible to others in the system once both users have agreed to be friends. This defines a friendship graph on all users in the system that allows a user to restrict access to their content. In particular, for any profile, diary entry, or multi-media object that a user posts, she can decide whether it should be visible only to her direct friends, or to friends of friends (FOF), or to every other user. You may also allow users to label

a friendship itself as visible to everyone or only to FOFs or only to direct friends. Finally, users should only be able to add comments to those diary entries they can actually see.

Finally, users may want to surf or browse the content of the system. They may decide to visit another user's page, which consists of his profile, the list of diary entries in reverse chronological order, the list of comments by other users about these entries, and the list of friends. A user's page, or any other page, should probably be accessible via some URL derived from the user name or location name, e.g., `http://mysocialnetworkapp.com/jimbo/` for a user with user name `jimbo`. From such a page, it should be possible to go to friends of this user by following links. It should also be possible to search the system by keywords in various ways, e.g., to get a list of all diary entries by friends that contain the word "malware", or all profiles that contain "security expert", subject to the access constraints specified by the content owner. Additionally, when a user logs in, besides pending friendship requests, she might also see a *news feed* which will be constructed from friends and their diary entries in reverse chronological order, newest event first. Each news item might have a like button and a counter with all the likes for that item from all users that clicked it. (Note that many of these items only have to be implemented in the second project, but you should design a database in this first project that can support these types of operations.)

Two more remarks. First, it is recommended to add various time stamps indicating when content was posted by a user, when she selected someone as a friend, or when she last accessed the system. This will allow you, e.g., to offer welcome messages to users that log in indicating any new friends or any new entries or photos posted on friends' pages since the last time they logged in. Second, you should not use database permissions or views to implement content access restrictions - there will not be a separate database account for each user, but the web interface and application itself will log into the database. So, the system you implement can see all the content, but has to make sure at the application level that each user only sees what she is allowed to see.

Project Steps: Note that the following list of suggested steps is intended to help you address the problem. You do not need to follow them in this order, as long as you come up with a good overall design that achieves the requested functionality. Note that in this first problem, you will only deal with the database side of this project - a suitable basic web interface will be designed in the second project. However, you should already envision, plan, and describe the interface that you plan to implement.

(a) Design, justify, and create an appropriate relational schema for the above situation. Make sure your schema is space efficient. Show an ER diagram of your design, and a translation into relational format. Identify keys and foreign key constraints. Note that you may have to revisit your design if it turns out later that the design is not suitable.

(b) Use a database system to create the database schema, together with primary keys, foreign keys, and other constraints.

(c) Write SQL queries (or sequences of SQL queries or PL/SQL statements) for the following tasks.

(1) **Content Posting:** Write a few (4 to 5) queries that users need to sign up, to create or edit their profiles, to post an image, or to add a new entry to their diaries.

(2) **Friendship:** Write queries that users can use to add or accept someone as their friend, to list all their current friends, or all their FOFs.

(3) **Browse/Search Queries:** Write a few (say 4 to 5) different queries that a user could use when accessing content in your system. For example, a user might want to see all profiles or diary entries by his friends, or by his FOFs, or by anyone, that contain certain keywords such as "ice hockey". A user may want a list of all diary entries by his friends during the last week. A user may want to see all locations liked by friends, or all locations for a conference.

(d) Populate your database with some sample data, and test the queries you have written in part (c). Make sure to input interesting and meaningful data and to test a number of cases. Limit yourself to a few users and a few diary entries and friendships each, but make sure there is enough data to generate interesting test cases. It is suggested that you design your test data very carefully. Draw and submit a little "map" that illustrates your test data! Print out and submit your testing.

(e) Consider defining appropriate stored procedures for various common tasks. Each stored procedure should have a few specified input parameters (such as the user name, or a set of keywords on which a search is performed, or the title, body, and of a new log entry that should be created, etc.), and should return a defined result. Note that in the second project, you have to call these procedures from a web-based interface outside your database, so find out how to define stored procedures, how they can be called, what restrictions there are, and what sort of technologies (like Java+JDBC, PHP, CGI) there are for interfacing from a web server.

(f) Document and log your design and testing appropriately. Submit a properly documented description and justification of your entire design, including ER diagrams, tables, constraints, queries, procedures, and tests on sample data, and a few pages of description.

Suggested Application Scenarios:

- Olympic games fans social network: users will be fans of summer or winter olympic games that will have the chance to socialize with other fans, make friends, suggest events, places, comment on athletes performances, place on the map venues for competitions, share scores, etc.

- Scientists social network: users will be professors and researchers that aim to socialize with people with similar research interests or to keep in touch with their research peers, they will post about their discoveries, conferences that they attend, lab picture and videos with their latest inventions.

- Sports fans social network: users will be sports fans that look to find users with preference for the same sports and teams, they will post comments and news about their team, games that they have been to, photos, videos, etc.

- Music fans social network: users will be music fans of all kinds, they will post about their favorite bands, music halls, news about concerts, photos and videos with their idols, etc.

- Employees of a large corporation social network: users will be employees that will maintain their collaboration projects online, they will post about updates, interesting developments with the project, interested related information, resources, etc.

-Outdoor enthusiasts social network: users will be people that enjoy outdoor activities such as hiking, birdwatching, fishing, snowshoeing, canoeing, kayaking etc, and want to socialize with other users with similar interest, to share information about places they visited such as tips about not so popular fishing spots, locations of stunning views, information about new trails, new parks, etc.

-University students social network: application where users can connect and follow other students from the university, upload different kinds of posts and events. Also, they can collaborate with other students for projects or other activities. In addition, users can create groups for different organizations and courses.

-Online bidding system: application similar to ebay for products that users want to sell. A user can simply post the product he wants to sell and for a specific time frame users can bid for that specific posted product and at the end, whoever bids the highest wins. Users can ask questions about products, can like or comment on different posts and follow different users.

-Real estate listing portal: application similar to zillow/trulia/craigslist housing system, where users could post the housing details/multimedia they want to sell or rent out. In addition the system will have the functionalities related to social media like comments, likes or follow different users or posts.

- Your own choice of application scenario approved by the Instructor.

NOTE: Once you pick an application scenario please contact the Instructor for validation of your proposal.

Project part 1 grade breakdown:

Database design	30
Create database schema (DDL)	15
Sample queries (DML)	20
Query result samples	15
Stored procedures, functions, views	10
Logs	10
Total	**100**