NYU Tandon School of Engineering
Computer Science and Engineering
CS6083, Fall 2018

November 11, 2018

Project #1 (due November 27)

You are hired by a startup company to help build the database backend for a new mobile app named *oingo* that allows users to share useful information via their mobile devices based on social, geographic, temporal, and keyword constraints. Of course, oingo borrows ideas from many existing applications such as Foursquare, Twitter, Facebook, etc. The main idea in oingo is that users can publish information in the form of short notes, and then link these notes to certain locations and certain times. Other users can then receive these notes based on their own location, the current time, and based on what type of messages they want to receive. The startup believes that this may become a popular application as it can be useful in many scenarios. For example:

- Suppose you discover a nice Thai restaurant (by finding it on the web or by walking by it) that you want to try out some time for lunch. To make sure that you do not forget the restaurant, you want the app to send you a note if you are ever within 300 yards of the restaurant during lunch time.
- Or maybe you see a store that you think one of your friends might really like. You could then add a note about this store, such that your friend (or maybe everybody) sees the note if they come within a certain distance of this place.
- Suppose the local historical society wants to leave a note that tells tourists about an interesting place (a building or a monument), so that tourists within 100 yards who use the app can see the information.
- A store or restaurant might want to show information about special deals to potential customers nearby during times when such deals are available (e.g., during Happy Hour for a bar).
- Or you might sit somewhere and be interested in meeting other people, either your friends who are nearby or people that you do not know yet.

So, the idea is that you can write a short note consisting of a few words (there might be a length limit, similar to a tweet). You can then associate this note with a location, a radius of interest (say, 100 yards) around the location, a schedule when the note can be seen (say, for the next half and hour, or on a particular date, or every Friday between 5pm and 7pm), and one or more tags that specify what kind of note it is (e.g, #tourism, #shopping, #food, or #transportation – or #me to indicate the location where a person is if they want to publish this fact while they are there).

Now the second part of the problem is how people can control what kind of notes they want to receive/see. For example, Carol might want to receive any notes about #food during lunch time, from friends or anyone else. Maybe if she is on the Lower East Side after 5pm, she wants to also receive any #me notes by friends so she can meet them. And if she is in SoHo during the weekend, she wants to receive any notes with tag #shopping. Thus, a user can have different filters (sets of rules) on what they want to see based on their current location, the current time, and maybe also their current *state* (such as “at work”, “lunch break”, “just chilling”, or whatever they choose as the description).

So this describes the basic idea behind the system. Before starting your work, you should think about what you envision the final system to look like, what data needs to be stored, and what kind of operations need to be performed. For example, there should be a login page, a page where a user can sign up for the first time (by supplying an email address and choosing a user name), and a page where users can create or update their profiles. Users should be able to ask other users to become friends, and should be able to answer friend requests. They should be able to post notes, with a short text (probably of type clob), a few tags, a location, a radius, and a schedule specifying when the note is visible. Other users should also be able to attach comments to the note, if the user allows it, such as “Thanks, great restaurant!”. Users need to be able to upload filters to specify when and in what situation they want to receive what types of notes.

Project Guidelines: In this first part of the course project, you will have to design the relational database that stores all the information about users, friendships between users, notes published by users, and filters that users have for what kind of notes they want to receive at different times and in different situations (i.e., states and locations). In the second part of the project, you have to design a web-accessible interface for this system. Of course, a mobile app for iPhones and Android devices might be a better choice for this type of application – but this is not a mobile app design course so this is not expected. Instead, for the second part you should design a browser interface that would also work on small screens, with not too much information on screen and with big buttons etc. You should use your own database system on your laptop or an internet-accessible server. Use a system that supports text operators such as like and contains. Both parts of the project may be done individually or in teams of two students. You will receive an email from the TAs asking you to sign up as a group. The second part of the project will be due around the final exam. Note that the second project builds on top of this one, so you cannot skip this project.

Note that both posting and receiving notes requires you to model a schedule; think about what kind of schedules (time rules) you can support without things being too complicated. It should be possible to specify things such as"Fridays between 5pm and 7pm" or "every day between 2pm and 3pm" or "on 2/15/2013 from 4pm to 5pm", but should people be able to say "the second Thursday of each month" or "on Easter Sunday"? When people specify areas (e.g., "'in SoHo"), think about how to best do that – maybe there is an extra table with definitions of the approximate boundaries of commonly used places and neighborhood names, or should for simplicity everything be modeled as a circle (point plus radius around it)?

Two more remarks: First, it is recommended to always store time stamps and locations when a note is posted by a user, when the user performs any action, and also at regular intervals in between (say every 5 minutes). (For privacy reasons, these records may be deleted after some time.) Second, you should of course not use database permissions or views to implement content access restrictions, such as a note only being visible to friends or being completely private (some users may only take notes for themselves). There will not be a separate DBMS 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 logged-in user is identified through the use of cookies in the second part of the project.

Project Steps: Following is a list of steps for this part of the project. Note that in this first part, you will only deal with the database side of this project - a suitable web interface will be designed in the second project, where you will also have a little more freedom in what to exactly implement.

(a) Design, justify, and create an appropriate database 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. Discuss in particular how you model schedules, locations, and areas.

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

(c) Write SQL queries (or sequences of SQL queries) for the following tasks. You may use suitable placeholder values in the statements.

(1) Create a new user account, with name, login, and password.
(2) Add a new note to the system, together with tags, and spatial and temporal constraints.
(3) For a given user, list all her friends.
(4) Given a user and her current location, current time, and current state, output all notes that she should currently be able to see given the filters she has set up.
(5) Given a note (that maybe was just added to the system) and the current time, output all users that should currently be able to see this note based on their filter and their last recorded location.
(6) In some scenarios, in very dense areas or when the user has defined very general filters, there may be a lot of notes that match the current filters for a user. Write a query showing how the user can further filter these notes by inputting one or more keywords that are matched against the text in the notes using the contains operator.

(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 notes and filters 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 diagram that shows your test data (not a long list of insert statements) and discuss the structure of the data.

(e) 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. This should be a paper of say 10-15 pages with introduction, explanations, ER and other diagrams, etc., that you will then revise and expand in the second part.