Polytechnic Institute of NYU
Computer Science and Engineering
CS 6083, Spring 2013

April 4, 2013

Project #1 (due April 19 at 4:00pm)

You are hired by a startup company to help build the database backend for a new mobile app named *jingo* that allows users to share useful information via their mobile devices based on social, geographic, temporal, and keyword constraints. Of course, jingo borrows ideas from many existing applications such as Foursquare, Twitter, Facebook, etc. The main idea in jingo is that users can publish information (small messages or 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 restaurant (by finding it on the web or by walking by it) and you think your friends might like this place. Or you find a nice pair of shoes in a shop window. It would be useful if you could post a note about this place, such that your friends (or maybe everybody) receive the note if they come within 300 yards of this place. Or maybe only if they come within 300 yards during lunch time.
- Suppose the local historical society wants to leave a note that tells tourists about an interesting place (a building, plaza, or tree), so that tourists within 100 yards who use the app can see the information. Or a bus company puts information about the schedule so that people within 100 yards of the bus stop can see when the next bus comes.
- 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 an SMS or tweet), and a hyperlink. You can 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, on a particular date, or every Friday between 5pm and 7pm), and one or more keywords (tags or channels) 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 *state* (such as "at work", "lunch break", "just chilling", or whatever they choose as the description), the current time, and their current location.

So this describes the basic idea behind the system. 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, it should be obvious that 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 try to design a browser interface that would also work on small screens, with not too much information on the screen and 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. However, you have to decide on a partner and email the TAs (Shi or Aditya) with your names by Tuesday, April 9. If we do not hear from you by that date, we will assume that you are doing an individual project. 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.

Before starting your work, you should think about what kind of operations need to be performed, and what kind of data needs to be stored. 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 hyperlink, a few tags, a location, a radius, and a schedule specifying whey when the item should be 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 specify when and in what situation they want to receive what types of notes.

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, or should everything be modeled as a circle (point plus radius around it)?

Extra credit: There are many opportunities for extra credit in this assignment, but they will only be graded with the second part of the project. Still, you may want to plan ahead. You could come up with smart ways to model areas ("in SoHo"), or you could try to use location data from places such as Yelp to identify if a user using the #me tag to announce her present location (this is basically a check-in) is currently close to and maybe inside a business known to Yelp. You could assume that certain common tags, such as #me, #food, #tourism, or #shopping, are predefined and maybe even suggested automatically when the user posts a note, but others can be invented as the user wants to. Concerning the state of a user such as "at work", ideally the system could even learn how to predict the state using machine learning based on location and time, so that the user does not have to manually change the state – but this is maybe too much work. Anyway, think about the possibilities and what you can do to make the final system nice and easy to use – develop a vision of what such as app could look like and what it should do. Finally, you could consider adding a "like" button (or star rating) so that other users could rate the usefulness of a note.

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: Note that the following list of suggested steps is intended to help you attack 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 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) Describe some basic assumptions that you will make in your design. Describe any extra features you plan to add to the description, and any things you are planning to not support because they seem too complicated or

useless (or you ran out of time). Discuss why you made these decisions.

(b) 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.

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

(d) Write SQL queries (or sequences of SQL queries) for the following tasks.

(1) **Signing Up and Posting:** Write a few (4 to 5) queries that users need to sign up, to login, to create or edit their profile, or to post or delete a note together with its hyperlink, tags, and various other constraints.

(2) **Filters:** Write two queries showing how different types of filters can be stored in the system, using locations, schedules, and tags.

(3) **Finding Notes:** Given a user and her current location, current time, and current state, write a query that outputs all notes that she should currently be able to see given the filters she has set up. Also write a query that adds a comment about a particular note, such as "Thanks for the tip!".

(4) **Reverse Finding Notes:** 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.

(5) **Searching Notes:** In some scenarios, in very dense areas or when the user has defined very general filters, it may be impractical to display all notes that should be displayed according to the filters. 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. For extra credit, suppose the user does not want to supply extra keywords, can you think of ways in which the notes can be ranked from most to least important in some way? (Describe your idea and try to show how to implement it with a query.) could be ranked from

(e) Populate your database with some sample data, and test the queries you have written in part (e). 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 (similar to figure X in the book) that shows your test data (not a long list of insert statements) and discuss the structure of the data.

(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. 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.