NYU Tandon School of Engineering
Computer Science and Engineering
CS 6083, Spring 2017

April 5, 2017

Project #1 (due April 19)

Your task in this semester's project is to design a database-backed website for crowdfunding, similar to services such as *Kickstarter* or *GoFundMe*. Such services allow people to raise money for various activities, such as starting up a company, building a new product, creating a piece of art, or charitable causes. More precisely, users should be able to post new projects for which money is requested, and other users can then pledge money towards these projects, and can also discuss and like projects. Moreover, once the project is funded and completed, users who gave money (sponsors) should be able to rate projects to express their satisfaction. In this first part of the project, you have to design a relational database that can serve as a relational backend for the service. In the second part, you will then design the actual web site, which accesses this database.

In particular, the service you have to build should focus on funding creative projects, such as making music, creating paintings or statues, filming a movie, or writing books or poetry. As an example, imagine a band that needs to raise funds in order to record their next album, or to do their next concert tour. They post a request for funds, together with a description of the project, and maybe some samples of their music and some pictures. They also provide a minimum and maximum amount of funds needed, an end time for their fundraising campaign, and a planned completion time for the project (say, the targeted release date of the record or end of the tour). The fundraising campaign is over when either the maximum amount of funds is raised, or the end date of the campaign arrives, whichever comes first. At this point, the pledged funds will be released to the band, provided that the minimum amount has been reached; otherwise, the sponsors keep the pledged money and the fundraising campaign has failed.

After successfully funding their project, the band will then work (off-line) on the project, say, recording their next album. During this process, they may periodically post updates and new materials (e.g., mp3s of new songs) to the project page, and users can of course keep discussing the project. At some point, the band will declare the project to be completed. At this point, the sponsors will be asked to rate the project, based on the posted materials or other information. Note that we are assuming that sponsors do not get anything in return for their money, other than the satisfaction of having helped in the creation of new artistic works; this is in contrast to some services where sponsors get shares in created companies, or get a product that is being make with their money. It might of course make sense for the band to promise a CD of the new album to anyone pledging more than a certain amount, but you do not have to model this case.

Thus, in your relational design, you need to model users that should be identified by a unique user name or an email. On the site, users will need to be able to sign up, log in, post and modify some basic public and private profile information (say, hometown, interests, credit card, etc.), post comments on projects, and pledge funds for a project. Users may also post new projects and ask for funds for these projects. Projects should probably have a name and description, and maybe some tags or category labels, and owners of projects should be able to add new entries with updates about the project, containing text and multimedia data (mp3s, pictures, movies) about the project; the details of this are up to you and for now you may just use clob and blob types as placeholders. When a project is successfully funded, a charge to the credit cards of the sponsors should be created and recorded. Your design should also support basic social networking features. Users should be able to like a project, and should be able to follow other users. That way, users can easily see all the comments, pledges, and likes of the users that they follow.

Both parts of the project may be done individually or in teams of two students, and you should have already received email from the TAs about signing up as a group. The second part of the project will be due before the final exam. In this first part of the project, you have to design the database backend of the system, that is, an appropriate relational schema with keys, constraints, and maybe views, plus a set of useful queries. You should use your own database system on your laptop or on an internet-accessible

server. Use a system that supports text operators such as “like” and “contains”, since you should allow users to search the content by keywords, and make sure that multimedia data can be stored as well.

Note that the second project builds on top of this one, so you cannot skip this project. The system described here has of course many similarities with existing crowdfunding sites, so you may want to look at some of them. Before starting your work, you should briefly think about how you envision the final system to look like at the end of the second part of the project, and what kind of operations and steps there are. For example, there should be a login page, a page where a user can sign up for the first time, and a page where users can create or update their profiles. There should be ways for users to post projects and updates about projects, to sponsor projects, to post comments, like, and follow, etc. Users should also be able to search for interesting projects by keywords, tags, or quality (e.g., how many likes a project has, or how good the ratings for past projects by this user were). Some of the details are only important in the second part, but start thinking about them now.

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 relational schema for the above situation. Make sure your schema is space efficient and suitably normalized. 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 for some of the queries or functionality. Provide a detailed explanation of your design decisions!

(b) Create the database schema, together with key, foreign key, and other constraints.

(c) Write SQL queries (or sequences of SQL queries) for the following tasks:

- Create a record for a new user account, with a name, a login name, and a password.

- List all projects that contain the keyword ‘‘jazz’’ and that are currently looking for funds, sorted in descending order by posting time.

- List all users who have given money for projects containing the tag or category ‘‘jazz’’ in the past, sorted by the total amount they have successfully pledged (meaning, money that was actually charged).

- List all users who have completed at least 3 projects, and where each of their projects received an average rating of 4 stars or higher from its sponsors.

- List all comments by users that are followed by user ‘‘BobInBrooklyn’’.

- Insert a new project for a particular user, with a name, description, and other needed info.

- Insert a pledge to sponsor a project, for a particular user, project, and amount.

- Write queries for the end of a funding campaign. E.g., you could use triggers to detect when a campaign is fully funded or time is up; if successfully funded, generate charges to sponsors’ credit cards.

(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 entries each, but make sure there is enough data to generate interesting test cases. It is suggested that you design your test data very carefully. Show your test data as tables, not as long lists of insert statements, and discuss the structure of the data. Print out and submit your testing.

(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, diagrams, etc., that you will then revise and expand in the second part.