NYU Tandon School of Engineering
Computer Science and Engineering
CS 6083, Spring 2018

April 8, 2018

Project #1 (due April 22)

In this project, your goal is to design a database that can serve as a relational backend for a new social networking site called *Jobster* that focuses on graduating students looking for jobs. In particular, the site should allow students to sign up and post their resumes and other information. Companies can also sign up and post information about themselves and job announcements. Students can decide to *follow* certain companies (which means they will get all job announcements by those companies), can apply for jobs, and can *friend* other students and forward tips and job announcements to their friends. Both companies and students can also search for matching applicants and job descriptions. All the data about students, companies, jobs, connections, and notifications must be stored in a relational database, and a web frontend will be designed in the second part of this project to access the database and present the content to users in a nice form.

Problem Domain: In your system, there are two classes of registered users, students and companies. Students that have signed up for the service can upload a profile consisting of basic information such as their name, the university they attend, their major and GPA, and maybe some other information (say, keywords about their interests and qualifications), and they can also upload a full resume. Students can invite other students to be their *friends*, and can accept or reject such requests from others. Companies that sign up can post a basic profile of their company such as their name, location of their headquarter, and industry (e.g., software, or transportation). Afterwards, companies can post announcements for open positions (i.e., job announcements). A job announcement should contain some basic information such as the company name, the job location, the job title, the salary, and the required academic background (e.g., an MS in Physics, or a BS in any field), plus a longer textual description of the position.

Students can find interesting positions by browsing, by searching by keyword (say, on job title, location, description, or all of these), or by *following* a company, which means that the student will be notified of any new job announcements of that company. Students can also forward any job announcements they find or are notified of to their friends, and a student should be able to apply for a particular job. (Here, applying just means that the company is asked to look at the resume of the student and decide whether to call or email them for an offline interview.) In addition, companies can push job announcements to suitable students by running a filter condition against all students' information – this means that all students whose university, major, GPA, or resume matches certain keywords and other conditions chosen by the company will be notified of the job, even if they did not sign up to follow the company. (In fact, this is probably how your website would make money, by charging companies for signing up and per recipient for any job announcement they push out – these are basically advertisements for jobs).

The system described here has of course many similarities with existing systems such as LinkedIn. Note that friendship is mutual, as in LinkedIn and Facebook, while following is unilateral as in Twitter. Any notifications (friendship requests and answers to such requests, messages from friends, notifications of new jobs, applications by students for a job) should also be stored inside the database. Then the next time a student or company logs into the web site or reloads a page, appropriate queries are run to display any new notifications. (Of course, in real systems there would also be the option of notifications via email or SMS.) Finally, students should be able to restrict access to their information (say, their GPA and resume) so that it is only visible to their friends and to those companies that they have applied to; other students and companies can only see their name, major, and university. However, companies may still push notifications to students using filter conditions on the restricted content, though they may not be able to see which students receive them.

Before starting your work, you should think for a little while about how you envision the final system to look like in the second part of the project, and what kind of operations and steps are there. 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 find content by browsing, or by searching using keywords and

other conditions. A user should be able to see a friendship request by another user, to answer a request, to follow a particular company, or to recommend a job to a friend. Most of the details are only important in the second part, but start thinking about them now.

Project Guidelines: 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 front end, which accesses this database. Both parts of the project may be done individually or in teams of two students, and you will receive email from the TAs about how to sign up as a group. The second part of the project will be due a few days 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 queries shown below. 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.

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 job sites, so you may want to look at some of them to understand the typical functionalities that they offer. Before starting your work, you should think a little about how you envision the final system to look like in the second part of the project, and what kind of operations and steps are there. 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 find content by browsing, or by searching using keywords and other conditions. A user should be able to see a friendship request by another user, to answer a request, to follow a particular company, or to recommend a job to a friend. Most 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 student account, with a name, a login name, and a password.
- List the names of all friends of a particular user.
- Delete all friendship requests that are older than one month and that have not been answered.
- List all students from NYU that are following Microsoft.
- List all job announcements posted in the last week that are looking for someone with an MS in CS.
- For each student with GPA > 3.5 whose resume contains the keywords "database systems", create a notification telling the student about a particular new job announcement that a company has posted.

(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.