CPSC 304 Project Instructions: Project Proposal

Due Date: Saturday, September 30, 2017 at 23:59

Last Update: September 23, 2017

Goals and Motivation

The goal of the project is to allow you to have the freedom to design your own application. We want to let you have the freedom to concentrate on some aspects and not on others. A consequence of this is that we won't be telling you exactly what to do. Since we don't expect you to read our minds, here is one purpose of the checkpoints: We will give you feedback on the project and tell you if you're on track to do well—and if not, then how to get there, without penalizing you for not being able to read our minds. However, for those who want a bit more of an idea of an acceptable project is, we offer the following (rough) guidelines. We expect that each application should eventually have:

- At least 7 entities and 5 relationships
- The ability to handle multiple classes of users (e.g., customers and bank employees)
- At least 10 queries that users will be able to ask via the application interface
- Note that you will create an ER diagram and formal specifications *later*. The first part is the just the Project Proposal. If you want to see a *preview* of the formal specifications and ER diagram, take a look at the following Bookstore (UBStore) example. (You may need to copy-and-paste the links.)

  https://www.ugrad.cs.ubc.ca/~cs304/2017W1/project/p1/p1-desc.html

  https://www.ugrad.cs.ubc.ca/~cs304/2017W1/project/p1/p1-solution.html

  Note that this example is based on a previous offering of this course—so, be sure that you follow the *current* instructions. You may *not* model any previous UBC example data (e.g., discussed in lectures—although the music/radio case is fine), employee supervision (discussed in a textbook), a bookstore (project topic in this example), MP3 storage (may be discussed in class), or a project given to you in the tutorials. Additionally, this must be a *new* project—you may *not* reuse a pre-existing project like something you got from someone else, something out of a book or from the Internet, something from a co-op term, etc.

What to Turn In

- A cover page: https://www.ugrad.cs.ubc.ca/~cs304/2017W1/project/CoverPage.html.

- Project Description: What is the domain? In particular, what aspects of the domain will be modeled by the database?

- What are the application specifications (i.e., what functionality will the system provide)? Take a look at the Sample Project Description below for an example.

- What platforms do you plan to use for your project (e.g., Oracle, Java, MySQL, PHP, other)? Note that you are allowed to use any programming language or relational database that you please, as long as you create an application for a relational database, that you meet all of the other requirements, and that you do all programming and query writing yourself (e.g., you can't use a platform that's going to write your SQL for you). At this time, we plan on offering support for Oracle from the CS machines, along with either JDBC or PHP/Web. If, however, you decide to go with anything other than the provided infrastructure (including using PHP on your own server), you may be required to submit additional information or additional checkpoints. However *no* support will be available for anything other than the recommended software.

    - Here are some tutorial web pages:

      https://www.ugrad.cs.ubc.ca/~cs304/2017W1/tutorials/tutorials.html

- Other, more specific comments if appropriate.

- Reminder: We just need the project proposal. You don't need to submit the formal specifications and ER diagram yet!

Sample Project Description

As an example, if we were going to turn in a banking proposal, we might submit something like the following for the project description:

The domain that we are going to model is Banking: data will be stored about the accounts in one bank.

The aspects of the bank that we will be modeling will be the information that is relevant to the customers of the bank. This includes things like the accounts that a given customer has, information specific about the customer (e.g., the customer's name and address), and the customer's credit cards. There will be a number of different kinds of accounts, and some of these accounts will have additional information to let customers know if that account is right for them (e.g., a mutual fund account will contain information on what stocks are included in the mutual

fund and how the fund performed over the past year). There will be two different classes of users of the system: the customers, and the bank's employees. The customers will be able to access their own accounts, and transfer money from one account to another. They will also be able to update some of their personal information, like their bank account password. The bank employees will be able to access all of the customers' data—both the customers' accounts and the customers' personal information. Only the bank employees will be able to change things like a customer's social insurance number (SIN). Bank employees may reset a customer's account password, but may not see what it actually is.

This project will be done using the Department of Computer Science's Oracle DBMS, and we will use Java and JDBC. We do not anticipate using any special software or hardware beyond this.

How to Use Handin

One group member should be the person doing all the electronic `handin` submissions. This will simplify things when the TAs have to check off the deliverables and associate them with the correct group.

To submit your Cover Page and your Project Proposal file (PDF preferred), perform the following steps:

- On an undergraduate machine (e.g., using `ssh` on `remote.ugrad.cs.ubc.ca`), copy the file(s) that you want to hand in, to the directory `~/cs304/project_proposal` (note that it will wind up in your home directory). You can create this directory using:
    - `mkdir ~/cs304/project_proposal`
- Copy your file(s) into this directory.
- Then, from your home directory, run:
    - `handin cs304 project_proposal`
- Take a screen shot of your successful submission, in case any problems exist.

Only one group member should submit the assignment. Group members: verify with your group that the submission has actually taken place!