Final Project

As you know the final project is a group project. The exact nature of your project will be negotiated between your group and me, but here are some general guidelines:

- Your project should have a scope that is appropriate to a multi-week, two- or three-person project.
- Your project should be robust enough to withstand use by users who are not the developers. It should be free of bugs. You may assume legal, good user input so that you can focus more on the more important aspects of the project.
- Any data that you store on the server side must be stored in a database.
- Your project should be well engineered. Your code should be well-designed, well-commented, and as simple as possible.

Each will need to meet with me on some weeks for 10 to 20 minutes to discuss progress. I will make a signup sheet available for group meetings later. Here are the due dates scheduled for the rest of the semester.

- **Thursday, May 10**: A team member should send me a message with your team formation info. (done already)
- **Monday, May 14**: You should provide your group’s written proposal describing what domain your app will be in, what the general idea of your project will be, what key features your app will support, key technologies and tools to be used (languages, programming environments, dbms server, etc.), expected results, tentative schedule, task assignment for each team member, and whatever else you want to include. Submit a Word file (proposal.doc) on Blackboard (one copy per group).
- **Tuesday, May 15**: Your group will have a 10-minute meeting with me during the class time to discuss your group’s ideas for your group’s project. During this meeting we want to address whatever questions and concerns your group may have.
- **Wednesday, May 23**: Submit a written progress report on your implementation to Blackboard.
- **Tuesday, May 29**: Submit an updated written progress report on your implementation to Blackboard.
- **Thursday, June 7**: In class project demonstrations.
- **Monday, June 18**: Completed project due via electronic submission to Blackboard by 5:00pm. To submit your group’s project, name your group’s zip or tar file using the initials of one of the members, e.g., jed.zip or jed.tar if John E. Doe is a member of the group.

As part of your **final submission**, you should include a report which should include at least the following:
- General description of your project, including goals, requirements.
- Use cases (brief).
- E-R diagram. (I am assuming you will be using E-R diagrams as opposed to ODL or UML.)
- What normalization techniques you applied and what the results were. Show the steps that you used during the normalization. You should show in what form your final set of relations are, e.g., in BCNF or 3NF (or 4NF if you went that far). Or, you may say that certain relations are in one form and certain other relations are in another form, and still certain other relations are in what form, etc. Your design should be sophisticated enough that there are aspects of your design that gets affected by the normalization techniques. If you say that we decided not to use any decomposition because decomposition would be counterproductive for our project, that would not be acceptable to me. Whether decomposition is helpful or not for your performance, lossless-join, dependency preservation, etc., I want to see that they are applied and had some effects on your design. I say this because for this project it is more important that you know how to apply these techniques than whether your end result is more efficient or not. If you don’t quite understand what I am saying here, be sure to ask me for clarification.
- User documentation (user’s manual), including for the database administrator. I as a user should be able to follow your documentation to see the configuration and be able to deploy and run your application. In your document you should provide the URL that we can use to access the front page of your web application.
- Which of your original goals were met and which were not. If not, explain why.
- How you might improve if you were to spend more time, say another month, on it.
- If you were to add additional features, what would they be?
- Any other that you would like to add.
- Code.

Some **general requirements** for the project include:

- The user-interface should be web-based.
- The application and trial data should be accessible to me. If you can’t meet this requirement, let me know in advance.
- The database should ideally support a fairly good number of tables, say 10 to 20, rather than just several tables, when normalized. Your application domain should be rich and complex enough to result in a good set of tables with a good set of functional dependencies. If it is not rich enough to provide enough complexity to the extent that your normalization becomes trivial, you have selected the wrong domain.
- The bulk of the work should be in the area of database design, rather than in supporting application code. Although the user interface can be non-trivial, it should not dominate other aspects of the project. Remember that this is a database systems course, not a web programming course.
- I am assuming that you will be hosting your database on an open-source non-proprietary database management system, such as MySQL, PostgreSQL, SQLite, probably on your own machine. Let me know in advance if this is not the case.

**Your Contribution:** If at any point during your work on the project you feel that the other member(s) of your group isn’t (aren’t) contributing a sufficient amount of time/effort, please let me know as soon as possible so that I can intervene. I reserve the right to give different grades on the final project to different members of a group.

**Suggestions:** These are the steps I would suggest that you take:

1. Find a domain to work on.
2. Gather enough knowledge on the domain. Perhaps you should find a domain where you already have sufficient knowledge. You don’t have much time to spend to research/learn a domain.
3. Come up with a high-level design using a high-level language such as E-R diagrams, ODL, or UML. Although you are free to choose any of these, I would recommend that you use E-R.

4. Translate your high-level design into relations.

5. By now, you would have a set of relations. Your design should be complex enough to have at least 10 tables (relations) at this point. Now, apply normalization techniques depending on which normal form(s) you want your relations to be in. Some relations may end up being in 3NF, others in BCNF, and still others in 4NF if you decided to go that far. I am guessing that you will end up with 15 to 20 relations by the time you are done with normalization.

6. By now, you have the final set of relations. Implement them as tables into a DBMS, e.g., MySQL, with enough data to populate the tables. I am sure you will be debugging these tables as you build them.

7. Write an application that accesses these tables. Your application as I said should be web-based, say PHP or something else. I do not require that you use PHP, but if you are using something other than PHP, please let me know in advance. Your web interface should be decent although I am not concerned about the commercial value of it.

8. If you have any questions, please ask.