NYU Tandon School of Engineering
Computer Science and Engineering
CS6083, Spring 2022

April 4,2022

Project #1 (due 4/18)

The goal of this project is to build a web-based system for a question answering service, similar to Stack Overflow, Yahoo Answers, Baidu zhidao, or even piazza. This is a service where users can sign up and ask questions, and other users can then supply answers to these questions. Also, the people asking a question can then grade the answers and maybe select one best answer, while people providing answers may earn points (karma points or influence points). Users should also be able to search the site using keyword queries, and questions and answers should be organized into topics according to a topic hierarchy.

Problem Outline: In the completed system, users should be able to sign up for the service under some username, and to provide a short profile. After signing up, users may post questions, and may also post answers to questions from other users. Each question consists of a title and the body of the question, both pieces of text, maybe of type *clob* or *text*. For each question, the users should also provide a topic, chosen from a predefined and stored multi-level hierarchy of topics provided by the service. (Your service may try to cover all topics, or focus on a limited set of topics, say programming, or databases, or American history.) Answers to a question consist of a piece of text (the answer). For each question and answer, you should of course also store who posted it, and when it was posted. Users should also be able to search the posted questions and answers in the system using keyword queries and to read a particular question and the answers that were provided. A user asking a question may also select a best answer among the posted answers and may mark a question as resolved (meaning that the given answer solved the problem, or that the user does not need the answer anymore). All users (even users just reading the question and answers) should be able to award *thumb-ups* (or *likes*, or *brownie* or *karma points*) to answers. Finally, users who have posted a lot of answers or a lot of best answers or who received a lot of thumb-ups may see their status increased, say from basic user to advanced user to expert – the precise criteriums for these different classes can be defined by you and could also include giving more weight to recently posted answers.

Project Guidelines: In this first part of the project, you should develop a suitable relational schema for the service that can store all the information about users and user profiles, questions, answers, topics, and thumb-ups. In the second part of the project, you will need to build a web-accessible front-end interface, with a back-end server interacting with your database, which should implement the functions mentioned and allow users to access the service through a web browser. You should use your own database system on your laptop or an internet-accessible server. Use a system that supports full text search operators similar to "contains". Most of you will probably use mySQL, SQL Server, Postgres, or Oracle - if you want to use another system, please ask the instructor for permission.

Both parts of the project may be done individually or in teams of two students. You will receive an announcement from the TAs asking you to sign up as a group or an individual. The second part of the project will be due around May 7. The second project builds on top of this one, so you cannot skip this project.

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, and a suitable web interface will be designed and implemented in the second part. However, you should already envision and plan the interface that you intend to implement.

(a) Design, justify, and create an appropriate relational schema for the above scenario. Make sure your schema avoids redundancies. Show an E-R diagram of your design, and translate it into a 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.

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

(c) Write SQL queries (or sequences of SQL queries or scripting language statements) for the following tasks. You may use suitable placeholder values in the statements.

(1) Create a new user account, together with username, email, password, city, state, country, and profile.
(2) Insert a new question into the system, by a particular user and assigned it to a particular topic in the hierarchy.
(3) Write a query that computes for each user their current status (basic, advanced, or expert status) based on their answers and your own chosen criteria for defining the status.
(4) For a given question (say identified by an ID), output all answers to the question in chronological order from first to last. Output the answer text and the time and date when it was posted, and whether an answer was selected as best answer.
(5) For each topic in the topic hierarchy, output the number of questions posted and total number of answers posted within that topic.
(6) Given a keyword query, output all questions that match the query and that fall into a particular topic, sorted from highest to lowest relevance. (Select and define a suitable form of relevance – you could match the keywords against the query title, the query text, or the query answers, and possibly give different weights to these different fields.)

(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. Limit yourself to a few users, questions, answers, and topics, but make sure there is enough data to generate interesting test cases for the above queries. It is suggested that you design your test data very carefully. Draw and submit a little picture of your tables that fits on one or two pages and that illustrates your test data! (Do not submit a long list of insert statements, but show the resulting tables.) 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 E-R diagrams, tables, constraints, queries, procedures, tests on sample data, and a few pages of description. This should be a paper of say 8-10 pages with introduction, explanations, E-R and other diagrams, etc., which you will then revise and expand in the second part.