CS 336 -- Principles of Information and Data Management

Spring 2019
Requirements Specification for the Database Programming Project

Introduction

In this project, you will design and implement a relational database system to support the operations of an **online auction system**. You will use HTML for the user interface, MySQL for the database server, and Java, Javascript, and JDBC for connectivity between the user interface and database server.

You will have to install your own virtual machine with a web server that will host your web application as well as a MySQL server. Everything will be under the AWS (Amazon Web Services). Many resources and recitations will be provided about how to do everything so don't worry. ☺☺

You are to work in **teams of four**.

Project Specification

As you probably know, there are a multitude of online auction systems on the web. The most popular one is eBay. I suggest that you visit this web site (although I suppose all of you have visited it many times) to get an understanding of the look-and-feel of an auction web site and how such a system is supposed to function.

The basic idea behind your online auction system is that it is intended to support a company that lets users buy and sell an item in an on-line auction. (Think *eBay*. So we will call the company that hired you BuyMe. If you are not familiar with how *eBay* works, please go to eBay to look for something to buy, and look at eBay's FAQ/Help pages.)

In general, a seller posts an item for sale, starting an auction, which will close at a specified date and time. The seller also posts an initial price, an increment for bids [which indicates a lower bound on how much must be added to current bid for the next valid bid] and a (secret) minimum price [the seller is not willing to give up the item for less]. Potential buyers post bids as part of the auction, and the user with the highest bid at closing time gets to buy the item. (Actually, they *must* buy the item.) The process of bidding is described in the above Help pages. The one fancy thing you need to implement is "automatic bidding", which involves the buyer setting a (secret) upper limit on how much they are willing to pay; every time someone bids higher than your current bid, the computer automatically puts in a higher bid for you, till your upper limit is reached. So if there is not much bidding, you may get the item for less than your upper limit. If someone bids past your upper limit, BuyMe will alert you.

NOTE: Each teams' BuyMe site will be restricted to a specific category of items (e.g., vehicles OR clothing OR computers ...) with at least 3 hierarchical subcategories (eg., truck, motorbike, car, foreignCar, for Vehicles), which **require** certain specific fields to be filled for those kinds of items. (*motivation*: I have been annoyed on eBay when people do not mention shoe size, or belt width and length. The choice of category is up to each group in the course. [**Bonus points** for any team that can *change the kinds of items available* without recompiling the code (i.e., be driven by data in the database).] Additional features for your BuyMe (some of which are not available on eBay) depend on the three classes of users supported:

I.*End-users* (buyers and sellers)

1. They must of course be able to create and delete accounts, and login and logout.
2. An end-user can search the list of items on auction according to various criteria based on the fields describing an item. (The more complex searches your team supports the higher the credit.)
3. Potential buyers should be able to set **alerts** for items they are interested in buying. (*motivation*: I am often looking for certain items (e.g., sandals, winter coats) that are no longer manufactured. I am interested when such items show up, esp. if I could also specify sizes & colors.)
4. A user should be able to view the *history of bids* for any specific auction, the list of all auctions a specific buyer or seller has participated in, the list of "similar" items on auction in the preceding month (and auction information about them). [You get to decide "similarity measure".]
5. If you wish, you can anonymize end-users so their actual login-names and e-mail addresses do not show up in auctions.

In addition to end-users, the BuyMe system must also support its staff.

II. Customer representatives are available to end-users for answering questions, and modifying any information, as long as the customer rep decides this is reasonable. (This includes resetting passwords and removing bids. So your system need not support any specific rules for removing bids.) They must therefore be able to perform such actions, as well as removing illegal auctions.

III. **One** *administrative staff member* , whose account will have been created ahead of time, will be able to create accounts for customer representatives. This person must also be able to generate summary sales reports, including: total earnings; earnings per { item, item type, end-user}; best-selling {items, end-users}. Communications/notifications between BuyMe, end-users and/or customer representatives, which would normally be done via email, are to be *simulated* by inserting a row into a special table named `EMAIL`, with columns labelled `from, to, subject, date_time, content`.

[Additional features are welcome.]

Implementation requirements

The system is to be accessed through a web interface, programmed in jsp, accessing a mySQL database. Details of the technical set up, which uses the Amazon Web Services (AWS), will be covered in recitation. At the end, you will need to provide us with a link, where we can invoke and test your program. This means that if you choose to develop the system on your own machines, you are responsible for making it available at some URL! In order to be fair to everyone in the class, please do not use frameworks or other tools that make programming easier. (If we had time to teach everyone such a framework, we would.)

Good luck! ☺

PROJECT PARTICIPATION RULES:
If someone is **not participating properly** in the project, by not carrying their weight, not attending meetings, not submitting things, it is suggested that the others send emails cc-ing the project TA, so we can keep an eye out when apportioning credit for the project.

IMPORTANT DATES

Feb 23 – *ER diagram* due date
Mar 2 – *Database schema* due date
Mar 24 – *Login web application* due date
Apr 21 – ***Final Project*** due date