**Media Evaluation Project**

This project is an ecommerce project to obtain information about various objects you are interested in purchasing. The system must be written in Java and has to be able to be compiled by the Oracle/Sun javac command and run from the Oracle/Sun java command, both from the command line of a Windows prompt. If you are an Apple user, you need to test this on a PC since the school lab only contains Windows machines.

**REQUIREMENT#1)** **Significant Code.** In order to demonstrate a reasonable sized project, your project should have at least 850 lines of code (not including blank or comment lines, where the GUI code cannot count more than 200 lines of code.)

**REQUIREMENT#2)** **Input Files.** The owners maintains a list of items they would like to acquire for their inventory. The system should load this data into a database for subsequent updating and query. This has to occur in two places in your program. The first is a file name from the command line with command line flags as parameters. Second, while the program is running, the user can also have a group of items listed in a file for the inventory. You need to keep track (in your storage) the item title, price, date of auction, seller/place and a picture of item and time of query.

**OPTIONS:** Optionally, you can also keep track of item description, the extra pictures provided inside each of the online listings, and results for both ongoing and completed listings.

**REQUIREMENT#3)** **Data Storage Management**. During the operation of the program, a user may want to add an individual record or modify/update or delete a record from the data storage (based on your phase 1 code). You should have a GUI for this purpose. Hash based storage is minimally required.

**OPTIONS:** It would be impressive to additionally provide the option of storing the data in mysql as a Database. If chosen, this option should be another parameter on the command line to give the user the option when reading in the initial set of data whether it will be stored by hash based storage or databases. In addition, this would be acceptable as one of the two innovations mentioned at the end of this document.

**REQUIREMENT#4)** **Persistence**. The privileged user/admin from a GUI should be able to reconstruct the storage from backup data (**Transactional Log)**. There has to be enough information in the log to carry out this task. To accomplish this, ever transaction that modifies the database should be written to a text file (the log) that contains the name of the transaction (e.g. INSERT, DELETE, MODIFY) with its parameters, along with the date/timestamp for security purposes. If the transaction doesn't make changes to the database, it need not be logged in for persistence, but you may want to keep it for security logging purposes.

**OPTIONS:** In terms of security only an admin should be able to do this. Also, for security purposes history of all transactions would be helpful with a time stamp. This would only be useful if you set up a registration system for users and their passwords and only the admin account can have access to the transaction log and use the reconstruction GUI. This would be considered an innovation (see end of requirements).

**REQUIREMNT#5) Offline queries.** The user should be able to conduct offline queries of your database via a GUI. The extent of these queries will depend on what you have stored in your database; i.e., the queries will target the specific information you have stored about each item and/or transactional history in your storage.

**OPTIONS:** Temporal considerations can enhance the offline query. Did you maintain a history of when items entered and left the system? Are you able to provide evaluation data over time and/or for specific time ranges (assuming the user or your system conducts such evaluations at regular intervals)? For example, the user may be monitoring the site for the same item over a number of months. Your data storage will be keeping the data obtained from each such query. Thus, the user can ask for example, how many of a given item(s) were up for sale over a period of time etc.

**REQUIREMENT#6) Online searching.** This will be based on Phase 2 URL and Network code, adapted to the shopgoodwill.com website. The system should provide a GUI similar to the “**Advanced search**” of shopgoodwill.com. Minimally, its purpose is to the information mentioned in Requirement#2 with a timestamp of when you to give more information to the user. For this, you need to construct the proper parameters in the url by concatenating and forming the appropriate URL for submission (again, based on the Phase 2 code). Once you received the html from the webserver, you should extract the above necessary information (typically based on regex) and store the data in the storage for subsequent searching.

**OPTIONS:** The Advanced Search has lots of possible parameters. The requirement mentions the minimum. The options would be to implement as many of the other parameters as well.

**REQUIREMENT#7) Reporting/Output File.** As stated at the beginning, the goal is to enable a person to enhance the current holdings of a personal inventory (or collection). The system should have the ability to output a report on what information has been obtained. Similar to Requirement#2, this should occur in *two* places. First, if the user provided an input file at startup, the user should have a further option to name an output file as the second parameter in the command line. In this case, the user wants to bypass the general system and automatically obtain information about each item and directly write a report to the output file. This file should not be in prose, but should list all the information your database/storage would contain for each item after an evaluation has been performed. The format should be data1 | data2 | … | dataN for the N pieces of data you are accumulating in the database for each item. The character | is the delimiter separating the data, so it can subsequently be read into Excel or other programs. At the top of this file should be the header: parameter1 | parameter2 | … | parameterN for the names of the information in that column of data (you can choose the header names). This functionality should also be provided to the user at any time using the program via a GUI, which would allow the user to choose the name of the file and select whether or not to perform a new search now to report on, or to report on whatever information is currently in your database. The latter is particularly important if the system is offline.

**OPTIONS:** If you have kept temporal information then, many more parameters will be included, as well as the possibility to write to a series of files (with same basic name and date of search as appended as suffix), where the results for each date of search will be written to a different file.

**REQUIREMENT#8) Documentation/Comments.** Every project must contain proper **documentation** and a reasonable amount of code comments. The protocol followed should be as discussed in class: meaningful variable names, class and method descriptions, purpose of variable by declaration and by parameter list of methods, before each block of code, and when appropriate line by line comments. A user manual (.doc/x) should be provided describing the functionality of your project and showing a picture of each GUI provided with a description below it as how to use it. If necessary, an installation guide should describe how to compile/install your system, listing system requirements.

**OPTIONS:** Scientific writing is a field to itself. Many embellished types of documents can be provided as well.

It is expected that you add at least two embellishments/innovations to the requirements above, from at least two different requirements.

**REQUIREMENT#9) Innovation#1.** Since it is your choice, you fill in the details.

**REQUIREMENT#10) Innovation#2.** Since it is your choice, you fill in the details.

**OPTIONS:** You can add more additional features if you think that is useful for the project.