Carnegie Mellon University
15-415 - Database Applications
Fall 2009, C. Faloutsos
Assignment 9: Database Application

Phase I Due: 11/10, 1:30pm

Phase II Due: 11/19, 1:30pm

Reminders

- Weight: **30%** of the homework grade, i.e. **9%** of the course grade.
- Out of **100** points.
- Please **type** your answers. Only drawings can be hand-drawn, as long as they are neat and legible.
- This assignment has two phases:
  Phase I is due 11/10 as hard copy in class.
  Phase II is due 11/19 as both hard copy in class and via email.
- Recitations: TAs will hold six recitations for this assignment.
  11/4 2:30pm-3:20pm DH 2122
  11/4 3:30pm-4:20pm DH 2105
  11/11 2:30pm-3:20pm DH 2122
  11/11 3:30pm-4:20pm DH 2105
  11/18 2:30pm-3:20pm DH 2122
  11/18 3:30pm-4:20pm DH 2105
- Estimated time: **15-30 hours**.
- For any questions contact Leman Akoglu (`lakoglu@cs.cmu.edu`).

Introduction

You are to create an online community web site, `CMUBook` similar to `Facebook` (`http://www.facebook.com`). In the process, you will learn the main challenges involved in providing such a service, and appreciate the importance of database technology in today's data-driven world.

This assignment is divided into two phases, as we describe in the Roussopoulos and Yeh methodology (see later for exact pointers).

In the first phase, you will plan your implementation strategy at a high level, and submit documentation that describes the major design decisions you have made. The first phase permits the TAs to evaluate your progress at an early stage and provide feedback.

The second phase is the implementation of the system. For both phases you are expected to give extensive documentation for the system. The target audience of the document is a fictitious person who will have to maintain the system afterward.

Requirements

Data Requirements

`CMUBook` is a system where users can upload photos, declare other users as 'friends', tag photos, etc. The system should include information about the users, the photos that users upload, and tag information about photos.

1. **User data:** For each user, we want to record the login name (e.g. "john123"), password, name, email address, and the role ("regular user" or a "system administrator"). Login names should be unique. Also, the password cannot be empty.

2. **Photo data:** For each photo, we want to record the URL and the owner. The user who uploads the photo is defined as the *owner* of the photo.

3. **Tag data:** In this assignment, we will assume that each photo contains only a *single* major person to be tagged (e.g. the one closest to the center). A photo can be tagged with the login name of an *existing* `CMUBook` member. The owner and the friends of the owner are eligible to tag a photo. An eligible person can tag a photo only once. We also want to record the timestamp of a tag. Tags can not be removed or edited, once added. To give an example, if on `Jan-1` the user `jsmith` tags a photo as containing `jsmith` and on `Jan-2` the user `tjohnson` tags the same photo as containing `alice`, the photo will have two tags (`jsmith`, `jsmith`, `Jan-1`) and (`tjohnson`, `alice`, `Jan-2`), where the first entry denotes the tagger (who generated the tag), the second denotes the "taggee" (the login name of the `CMUBook` user the tag refers to), and the third is the timestamp. A photo may have zero or more tags where the "taggee" can be different (controversial tags, as seen in the example).

4. **Friendship data:** For each user, we want to record a list of his/her friends; two users A and B become friends if A invites B and B accepts (See Task-8).

Functionality Requirements

Task-1 **Registration:** Before a user can start using your system, s/he needs to register by providing his/her name, email address, and by choosing a login name and a password. If a login name chosen by a user during registration already exists, the system should give an error message and prompt for a different login. You can just store the password as clear text, and you need not verify the validity of the email address. *Note:* a sample registration is provided for your reference. Please make any (or none) modification to meet the above requirement.

Task-2 **Login/Logout:** A user should be able to login with the login/password s/he provided during registration. When the user logs in, s/he should see his/her "profile information" including (1) all the information about this user (login, email address, etc.), (2) the list of all photos they own (their URLs), (3) the list of all their friends

(their login names) and (4) the list of users who sent friendship invitations (again, their login names).

For security, no one should be able to view any page of your web site without logging in (except the login page, of course!) Also, once logged in, the user should not need to log in to view other pages. A login session is valid until the user explicitly logs out, or the user remains idle for 30 minutes.

Hint: In this assignment we will use Apache Tomcat, which provides easy session and cookie management primitives using cookies and server-side persistence. The default timeout for the sessions for your tomcat JSP server is 30 minutes, so you need not modify that. Please refer to Chapter 6, Sections 6 and 7 of Thinking in Enterprise Java (see below) for detailed examples of using session and cookies in JSP.

Task-3 **Photo Upload:** A user could add a photo by specifying the URL. The user can attach zero or one tag at upload time.

Task-4 **Photo Browsing:** A user must be able to search photos by providing a "taggee" as the search key (a valid `CMUBook` member login). A user can search for only his/her own photos and the photos of his/her friends. The result should return all the (eligible) photos: the owner, the URL, and all the tag information (tagger, "taggee", timestamp) for each photo.

Photos in the result should be *sorted* by their number of tags with the provided "taggee" in descending order. For example if a user searches for photos with "taggee" `jsmith`, a photo containing 5 tags total, 4 of which have "taggee" `jsmith` should be listed *before* a photo containing 10 tags total, 3 of which have "taggee" `jsmith`. You should show only the top-10 photos if more photos qualify.

Task-5 **Photo Tagging:** A user must be able to view his/her photos as well his/her friends' photos and tag the ones s/he wants to tag. Recall that a user can tag a photo only once.

Task-6 **Profile Review:**

A user should be able to see his/her friends' "profile information": their user information and list of photos.

Task-7 **Search by User Name:** A user must be able to search users by name. Partial matches are allowed: for example asking for "Smith" should match the user name "John Smith, Jr.". Your system should return the login name of all qualifying users. Friends of the user should be listed at the top and the rest should be at the bottom. Both lists should be *sorted* in ascending alphabetical order on `name`.

Task-8 **Friendship Management:** A user should be able to send a friendship invitation to another user if they are not friends already. Conversely, s/he should be able to accept his/her friendship invitations.

In this assignment we will allow *only* bi-directional friendships. That is, user `A` and user `B` are considered friends, only after one of them has sent an invitation to the

other, *and* the latter has accepted. For simplicity, you are not required to implement denial of invitations, nor retraction of invitations.

Task-9 **Reporting:** The system should contain a built in administrator "`admin`". Make sure you submit your password in your hard-copy for the second phase. The administrator should be able to check summary statistics of user activity in a report page. The page should contain:

(a) the total number of registered users in the system;
(b) the total number of photos in the system;
(c) *Heaviest uploaders:* the top-10 users who own the largest number of photos, and the numbers of photos they own. Break tie using the login name (in ascending alphabetical order); exclude users that do not own any photo (note that in some cases less than 10 users might qualify).
(d) *Heaviest taggers:* the top-10 users who tagged the highest number of photos, and the numbers of photos they tagged. Break ties using the login name; exclude users which have not tagged any photo.
(e) *Most popular photos:* the top-10 photos who contain the highest number of tags, and the numbers of tags they contain. Break ties using the URL; exclude photos which have no tags.
(f) *Most popular users:* the top-10 users who were tagged in the highest number of photos, and the numbers of photos they were tagged in. Break ties using the login name; exclude users which have never been tagged.
(g) *Busy days:* the top-10 days in which the highest number of photos were tagged, and the numbers of photos that were tagged in those days. Break ties using the timestamp (in ascending time order); exclude days which have no tagging activity.

Setup

A web server (`Tomcat 6`) with `JSP` support has already been set up for you. To access it, you should log into your account on `newcastle.db.cs.cmu.edu` (the one you used for HW 3 and HW 7). Please clean up your directory and type the following commands:

- create the web directory `www`:

```
$ ~lakoglu415/setup_db.sh
$ ~lakoglu415/setup_web.sh
```

  If you are prompted about replacing any files, choose the "[A]ll" option.
- create the database hw9, and add user 'www' after starting the pgsql server:

```
$ createdb hw9
$ createuser www -P
Enter password for new role:
Enter it again:
Shall the new role be a superuser? (y/n) y
```

  Note: this user is for setting up the website only, it is not a user of the `CMUBook` *web site.*
- add table 'users' to the database:

```
$ psql hw9
hw9=# create table users(login varchar(32), fullname varchar(64),
passwd varchar(64), email varchar(128));
```

  Note: this table is required to run the demo code only, for your submission, you should change the table as per the data and functionality requirements.
- exit the sql environment (`\q`). Run `echo $PGPORT`. You should get a port number.
- change the port number for localhost in line 37 of `www/register.jsp` to the one you just got.
- change the password in line 37 of `www/register.jsp` to the one you just entered.
- access `http://newcastle.db.cs.cmu.edu:8080/<andrew-id>415/register.jsp`. It should be similar to the sample registration page at `http://newcastle.db.cs.cmu.edu:8080/lakoglu415/register.jsp`.

After setup, every time you login, you can start the postgresql server by running

```
pg_ctl -w start
```

Every time when you logout, you must stop the postgresql server by running

```
pg_ctl stop
```

Code Organization

In this project, you will write `HTML`, `JSP`, and `Java` code. The `www` directory contains all the code accessible online. Here is the recommendation for your code organization (# for comments). Feel free to deviate from it, but following it will make grading easier.

```
-~                      # home directory
  - src                 # source code directory, including Java, JSP and HTML
  - www                 # where you deploy your web application

    - WEB-INF
      - web.xml         # your web configuration file, see Tomcat doc
      - classes         # you should put all compiled Java classes here
      - lib             # any external library, e.g. postgresql-8.3-603.jdbc4.jar
    - META-INF          # Tomcat specific context configuration directory
                        # no need to modify
```

Note: In the handout sample implementation of registration, all processings are directly included in `registration.jsp`. While this provides a simple solution, it may create complex `JSP` pages in a large project. A more elegant way is to use the `MVC` pattern - separating functionality of model, view and control. There are many such tools available, e.g. `struts`. You are welcome to use `struts` if you would like, but any working solution will get full points (with, or without `struts`).

Phases

The two phases of this assignment follow the work-processes methodology from *Adaptable Methodology for Database Design* by Roussopoulos and Yeh [IEEE Computer, May 1984] (`http://www.cs.cmu.edu/~christos/courses/dbms-F09/slides/CMU_ONLY/Roussopoulos-Yeh.pdf`, the lecture slides are also available at `http://www.cs.cmu.edu/~christos/courses/dbms-F09/slides/20methodology.pdf`).

- [**Phase I**] Environment and Requirement Analysis, System Analysis and Specification, Conceptual Modeling, Task Emulation.
- [**Phase II**] Implementation and Testing

Point Distribution and Deliverables

Phase I: report in hard copy, Due: 11/10 1:30pm [35 points]

The Phase I report should contain the following:

1. [**1 pt**] The top-level information flow diagram, (very important - also, don't forget the system boundary).
2. [**1 pt**] The list of documents.
3. [**3 pts**] The document forms, including the assumptions and design decisions you made.
4. [**5 pts**] The E-R model (omit attributes, to avoid cluttering the diagram). Make sure you specify cardinalities of the relationships following the format in hw1 solution.
5. [**2 pts**] List of the attributes for each entity and relationship.
6. [**3 pts**] Explanations of the any non-obvious entities and relationships.
7. [**5 pts**] The schema in the relational model. Make sure the schema is in a good form (BCNF or 3NF).
8. [**2 pts**] Explanations (e.g., primary keys, additional functional dependencies, explanations why a table is not in BCNF etc.)
9. [**2 pts**] The SQL DDL statements to create the above relational schema.
10. [**10 pts**] The SQL DML statements for all the tasks.
11. [**1 pt**] Run the registration task given in the demo code. Report the output generated by the webpage when you register your andrew-id.

Phase II: report in hard copy, Due: 11/19 1:30pm [15 points]

The Phase II report should contain the documentation produced in this phase:

1. [**0 pts**] system administrator password.
2. [**5 pts**] a source program listing.
3. [**5 pts**] a user's manual for the system.
4. [**5 pts**] your testing efforts: erroneous cases that your system can detect and handle reasonably (e.g. *non*-member trying to access the system, user trying to view a *non*-friend's profile information, user trying to tag a photo *twice*, etc.).

Phase II: source code in soft copy, Due: 11/19 1:30pm [50 points]

You should also zip all your source code into a file `hw9-your-andrew-id.zip` and send to `lakoglu@cs.cmu.edu` with the subject "`15415-hw9-`*`your-andrew-id`*". The weight of the tasks are as shown in Table 1. Notice that about two thirds of the points are for basic implementation that supports the desired functionality, and the rest are for error checking (as described above).

Note: GUI is not the emphasis in this assignment.

Task	basic implementation	testing/error checking
1.Registration	0	0
2.Log in/Log out	5	2
3.Photo Upload	4	2
4.Photo Browsing	4	2
5.Photo Tagging	4	2
6.Profile Review	4	2
7.Search by User Name	4	2
8.Friendship Management	4	2
9.Reporting	5	2
Total	34	16

Table 1: The weight distribution for Phase II tasks

Optional Documentation and Resources

Thinking in Enterprise Java (`http://www.mindviewinc.com/downloads/TIEJv1.1.zip`). Chapter 4∼6 covers how to connect to databases, servlets and JSP. Chapter 3 on RMI is optional.
Thinking in Java (`http://www.mindviewinc.com/downloads/TIJ-3rd-edition4.0.zip`): a good introductory book on Java programming;
Thinking in Patterns (`http://www.mindviewinc.com/downloads/TIPatterns-0.9.zip`): Java design principles and methods.
Tomcat 5.5 manual is available at `http://tomcat.apache.org/tomcat-5.5-doc/index.html`.
struts is a framework for building enterprise web application. It uses MVC pattern (Model-View-Controller). The API is available at `http://struts.apache.org` and you could find some nice tutorial at `http://struts.apache.org/2.x/docs/tutorials.html`.