15-415: Database Applications
Project 3

CMUQFlix - CMUQ's Movie Recommendation System

School of Computer Science
Carnegie Mellon University, Qatar
Spring 2014

Assigned date: March 19, 2014
Due date: April 05, 2014 by 11:59PM

I. Project Objectives:

You are to create a movie recommendation web site, CMUQFlix, which allows users to (1) login, (2) "like" movies, and (3) get personalized recommendations for movies. The goal is to learn how to set up a web site with a database back-end.

II. Data Storage:

CMUQFlix uses PostgreSQL to store different types of data as described below:

1. **User data:** For every user, we want to record his/her login name (e.g. "superman2014"), password, and email address. Login names should be unique and passwords cannot be empty.

2. **Movie data:** For the movies, you will use the movies database (from P1) with only one field, *title*, the movie title.

3. **Movie "likes":** For each user, the system should store the movies he/she has "liked." A user cannot "unlike" a movie. If a user "likes" a movie multiple times, only one count is recorded.

III. System Functionalities:

CMUQFlix's website consists of different web pages that serve different purposes as described below:

1. Registration Page: Before a user can start using your system, he/she needs to register by providing a login name, email address, and password. None of the fields can be empty. 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.

2. Login/Logout Page: A user should be able to login with the login/password he/she provided during registration. For security reasons, no one should be able to view any page of your website without logging in. 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 use Tomcat which provides easy session management. 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* for a detailed example of using session and cookies in JSP.

3. Profile Page: When a user logs in, he/she should see his/her "profile information". This includes (1) all the information about this user (login name, email address), (2) the list of movies he/she has "liked."

4. Movies Likes Page: A user should be able to navigate to a web page which lists all movies in the database with a checkbox "Like" next to each entry he/she has not yet liked.

5. Movies Recommendation Page: A user should be able to navigate to a web page which generates up to 5 personalized movie recommendations based on his/her existing "likes."

    - If the user has not "liked" anything before, display the top 5 movies with largest in-degree (= most "likes"). Break ties by sorting the titles alphabetically.

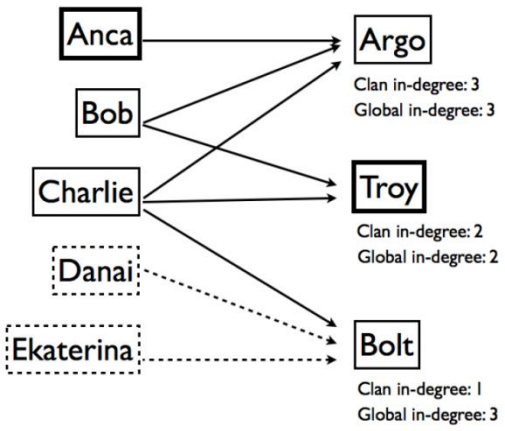

Figure 1: Recommend a movie example (T5), and "clan": the logged in user is Anca; her "movie clan" is Bob and Charlie (since they have at least one movie in common with Anca); the system should recommend 'Troy' and 'Bolt', with 'Troy' first, because it is more favorite within Anca's "clan"

    - If a user has "liked" at least one movie, then recommend movies that similar-minded people "liked", sorted by popularity (among those similar-minded people). See **Figure 1** for an example: to make a recommendation for user "Anca" (in

double box), find her "movie clan", i.e., persons who liked at least one movie in common with "Anca" (i.e. Bob and Charlie, her "movie clan" in this example); then, report the most popular movies within Anca's "clan". That is, display up to 5 movies with the largest in-degree from the "movie clan," in decreasing order, breaking ties by sorting in alphabetical order on the title. In the example of the figure, the movies to recommend to Anca would be 'Troy' (clan-in-degree = 2), and 'Bolt' (clan-in-degree = 1). (Of course, we don't want to return 'Argo', since Anca has liked it already.)

6. Reporting Page: The system should print a summary of user activity in a report page. The page should contain:
   - The total number of registered users in the system.
   - The totals number of "likes" in the system all users.
   - The list of movies that no one has liked, sorted in alphabetical order on the title.
   - The list of the "avid users" - the usernames of users with the most number of "likes", limited to top 10, sorted by the number of likes. Break ties by sorting the usernames alphabetically.

IV. System Architecture

CMUQFlix adopts three-tier architecture with a *front-end*, *middle tier*, and *back-end* as shown in **Figure 2**.

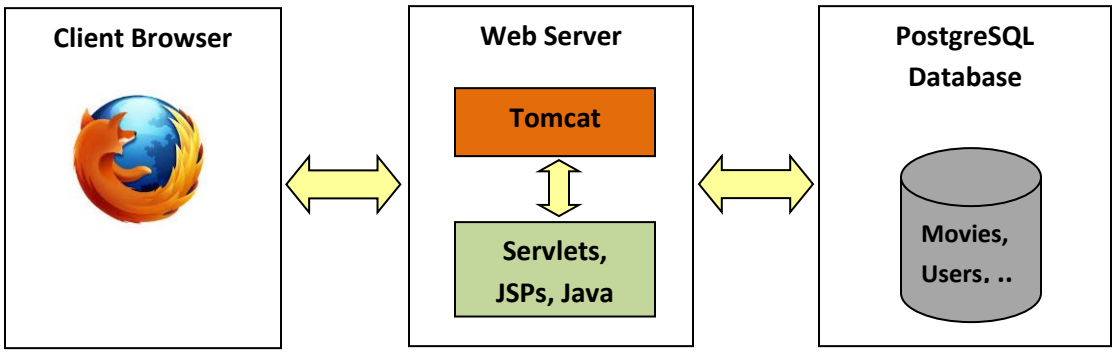

Figure 2: Three-Tier Architecture of CMUQFlix.

The *front-end* is a client browser displaying the webpage of CMUQFlix. The *back-end* is a postgreSQL database housing the movies and user relations (among others). The middle tier is the middle-man that connects the front-end with the back-end to make a whole system. Particularly, the middle tier consists of a web server (such as Tomcat) that, as the name implies, serves clients' HTTP requests for HTML webpages.

An HTML webpage served by Tomcat is either a *static* page whose HTML content has been predefined at the time of the website creation or a *dynamic* page whose HTML content is produced by invoking and running a **servlet** or JavaServer Page (**JSP**). Servlets and JSPs are technologies that help encode our application's logic (like communicating to a back-end and querying a database) and dynamically generate web

pages. Therefore, the web server houses servlets and/or JSPs that collectively make the content of CMUQFlix.

V. Project Milestones:

To develop CMUQFlix, you are required to do the following:

- Configure and run Tomcat on your virtual machine
- Create a PostgreSQL database with all the relations required to store CMUQFlix's movies and users information
- Copy the directory *cmuqflix* (posted on the course webpage) to Tomcat's webapps directory. This directory shall be the working directory for this project.
- Write HTML, servlet, JSP, and/or Java code that collectively achieve the functionalities listed in section IV.

VI. Code Organization:

The *cmuqflix* directory (to be placed in Tomcats webapps directory) contains all the code accessible online (including HTML, servlet, JSP, and/or Java). You must organize cmuqflix as follows:

```
~ /webapps        # webapps directory
    /src          # source code dir, including Java, JSP and HTML
    /cmuqflix     # where you deploy your web application
        /WEB-INF
            web.xml   # your web configuration file, see Tomcat doc
            classes   # dir for compiled Java classes (.class files)
            lib       # dir for external library (e.g. postgresql-8.3.jdbc4.jar)
```

VII. Final Deliverable:

In an archive (named P3_<your_andrew_id>.zip) add your modified *cmuqflix* directory. Also add a script containing all CREATE statements required to create the relations you define and use for this project.

VIII. Getting Help:

You can get help by visiting the professor and the TA during their office hours or by appointment. You can also post your questions on Piazza. The link for the course home page on Piazza is: http://piazza.com/qatar.cmu/spring2014/15415/home

IX. Late Policy:

- If you hand in on time, there is no penalty (duh!).

- 0 -24 hours late = 25% penalty.
- 2 4 -48 hours late = 50% penalty.
- More than 48 hours late = you lose all the points for this project.

NOTE: You can use your grace-days quota. For details about the quota, please refer to the course syllabus.