CSC309H5 Winter 2016

Assignment 2: Twitter App

Due date: February 26, 2016, 10:00 PM

Work in groups of 2 students

Before we start, outside the allowed libraries, all code in this assignment must be your own! That is, you must write from scratch you HTML, CSS, Javascript, PHP, schema, etc.

Your job is to create a web service which interacts with data retrieved from the Twitter API. Your application will be browser based, and it will run on large and small screens (be responsive).

Below are some descriptions of the service and the requirements that need to be satisfied. The descriptions are not completely concrete so that it allows room for your creativity; however, the specified requirements must be satisfied no matter what your application will look like.

In general, the service would allow registered users to login to the service and interact with Twitter content in a personalized manner. A user creates “word clouds” generated from tweets and share the word clouds with all other users in the service. The service should consist of the following views.

1. Landing page, where a user can choose whether to login or sign up.
2. Sign-up page, where user register to the service by creating their username and password.
3. Main page, which is shown after a user logs in or signs up; most of the web service’s functionalities are performed in this page..
4. Error pages, in case of service outage, you need to display proper error pages. Google Image “error pages” to see examples.

General Requirements:

1. The tools that you are allowed to use in this assignment: HTML, CSS, JavaScript, PHP, PostgreSQL.
2. The library that you are allowed to use: **JQuery** and **twitter-api-php**.
3. Get the latest version of JQuery: http://jquery.com/download/
4. Get the latest version of twitter-api-php: https://github.com/J7mbo/twitter-api-php
5. All pages of your website must be responsive, i.e., they must have reasonable layouts viewed from both desktop browsers and smartphone browsers. Use CSS Media Query properly.
6. You website does not need to support Internet Explorers.

Landing page requirements:

1. The landing page has your service's name, logo, as well as some tagline that tells users what your service is about.
2. The landing page also has the login boxes and button, where users can login using their username and password. The landing page must validate the username and password entered and return error messages when something is wrong with the entry (wrong password, user does not exist, etc).
3. The landing page also has a button that leads users to the sign-up page.

Sign-up page requirements

1. New users to the service will create their profiles in the sign-up page. The profile includes the following information.
    a. username
    b. password (which should be hashed at the client side, which is not perfectly secure but better than sending plaintext password to the server) **Update: hashing at the server side is okay, too.**
    c. profile picture, the user can upload their own profile picture, or choose from some predefined profile pictures provided by the service. A user must always have a profile picture.
    d. We don't require email addresses from users.

2. The sign-up form must be validated for the following criteria
    a. the username is not already taken
    b. the username must start with an alphabet and contains only alphabets, numbers or underscore ("_").
    c. the username is case-insensitive
    d. the password must be at least 6 characters long.
    e. the size of the uploaded profile image must be smaller than 50KB in size.

3. After successfully signed up, the user will be led to the **main page** of the service.

Main page requirements

1. The main page must be **completely AJAX-based**, i.e., all page updates must be done through AJAX requests and responses. The URL of the main page should never change. And the user never need to refresh the page while using the service.
2. The default view of the main page is the public timeline, where we display a list of the "word clouds" shared by all users in the service. You can click "Like" on the word clouds you like, and each post displays the total number of "likes" it gets. The user need to be able to sort the list according to "Latest" or "Most popular".

3. The object to shared in this service is the word cloud generated with tweets from Twitter e.g., you retrieve a bunch of recent tweets about “Star Wars”, process these tweets by counting word frequencies and generate a word cloud that could look like the following. Your word cloud does not need to look exactly like the following picture, but you should try to make it as visually appealing as you can (this can be done by properly manipulating the CSS properties of the displayed words).

2005
group
see
Vader
Empire
power
19
next
set
1999
draft
Back
films
IV
two
Wars
Luke
May
15
series
Nazi
II
25
side
Jedi
One
21
years
first
Main
film
v
later
life
Star
trilogy
new
Ram
Lucas
16
original
John
time
Han
III
write
six
Story
George
also
VI
J
12
Anakin
good
made
turn
Episode
duel
VII
writing
Darth
2015
space
Yoda
date
video
August
Networks
rebels
1980

4. The purpose of this assignment is that you retrieve text data from Twitter, process the data in some in a sensible way, and present the processing result in an interesting manner. So as long as this purpose is met, you could also do something that is different from the word cloud. If you want to do something different, please make sure to consult with Larry first and get his approval of your idea.

5. There is a “CREATE” button somewhere in the page, by clicking on which the user can create a new word cloud. An example flow would be:
    a. enter some keyword like “Star Wars”, click on “generate word cloud by keyword”, then the server will retrieve from Twitter Search API recent tweets with keyword “Star Wars”, process those tweets and generate a word cloud.
    b. or, put in some twitter username like “@UofT” then click on “generate word cloud for twitter user”, then server will retrieve from Twitter API the recent tweets posted by @UofT, process it and generate a word cloud.
    c. After the word cloud is generated, you may click on a “share” button to share it with other users in the service.

Error page requirements

When there is anything wrong with the service so that the user cannot get the expected response for their request, we should give users feedback about what is going on. The error message feedback could be displayed in the two following ways.

1. A separate page saying "Oops, something is wrong", this kind of page should be used for serious long term errors.
2. A pop-up or drop-down within the main page saying something like "Could not connect to Twitter, please try again later". This is used for temporary errors.

Database requirements

1. You will use PostgreSQL as you database, and access it through PHP.
2. Based on what will be implemented in the service, you need to figure out what tables need to be created and what the relations are between the tables. Some clearly needed tables include:
    a. table storing user profiles: each user's username, password (hashed) and profile picture, plus any additional information.
    b. table storing word cloud posts, each post should have an ID, timestamp, content, author, number of likes, plus any additional information.
    c. table storing the user-post relations, i.e., which user posted which word cloud.
3. When submitting your assignment, you need to provide a schema script which will populate the the database with some dummy data, so that the TA can see the service in fully functional mode.

Twitter API

Twitter API is very well documented and developer friendly, you will need to follow the the Twitter API documentation at https://dev.twitter.com/rest/public to figure out how to retrieve data from Twitter API. Roughly you need to do the following steps:
1. Login to dev.twitter.com as a developer and create an application.
2. Get the key, secret key, and authorization token that will be used to authenticate your twitter access.
3. Use your credentials together with the twitter-api-php library to access Twitter data.

Here are some tutorials about creating Twitter App and using the PHP library, each of which takes 8 easy steps.
- http://iag.me/socialmedia/how-to-create-a-twitter-app-in-8-easy-steps/
- http://iag.me/socialmedia/build-your-first-twitter-app-using-php-in-8-easy-steps/

Be aware of **rate limits**: Twitter has certain limitations on how frequent an app can access the API. Be careful not to exceed this limit, or it could seriously delay your progress. Read the following page for details.

- https://dev.twitter.com/rest/public/rate-limiting

Some API calls that are more relevant for this assignment:

- The search API (search tweets by keywrods): https://dev.twitter.com/rest/public/search
- statuses/user_timeline (get tweets by a user): https://dev.twitter.com/rest/public/search

Background processing of the tweets

You have the freedom to decide how exactly you want to process the data downloaded from Twitter, you should try to process it in such a way that the processing result you get is interesting and informative. Some of things that you may consider doing:

- chop up the messages into words
- filter out some "stop words" such as "the", "a", "are", "is", "am"...
- remove duplicate words in one tweet
- remove hashtags
- compare the frequency of a word with it normal frequency in general text documents.

Advanced/Creative features:

Part of the marks you get for this assignment will be for any advance or creative feature you add to your service. Below are some ideas:

- Use geo-location info on the phone to get nearby tweets
- More secure authentication process (e.g., client hash the password using a seed provided by the server, etc)
- Long polling, i.e., new posts automatically show up without needing to click on a button, without needing to send a request with fixed interval.
- Some fancy animation effects on the front-end.
- Allow user to connect the app with their Twitter account, so they can repost their word cloud to their Twitter (or even Facebook) timeline
- Some extra interaction with the word cloud, e.g., click on a word, show something.
- Be creative and surprise me...

Submission checklist

You will submit a single file named **a2.zip** to MarkUs, which should include the following files

1. readme.txt, which briefly explains what you have accomplished in this assignment, what are the requirements that you have / have not met; what are the advance / creative feature that you have implemented. Or any other info that is necessary for the TA to know while marking.

2. At the beginning of readme.txt, include the URL to the live version of your web service, which is run under one group member CS.UTM account.
3. All HTML, CSS, JavaScript, PHP or images files that are used to implements the service.
4. Also include the JQuery and twitter-api-php library files that you use.
5. A schema.sql script that populates the database with some dummy data so that all your service's features can be tested.
6. Before submission you should replace in all files all the UTORIDs and passwords with placeholders "UTORID" and "PASSWORD", so that the TA will be able to easily grep them and change them into his own credentials and run your service under his account.
7. The total size of your zip file must be smaller than 5MB.

Evaluation

- 80% of the marks will be for the basic requirements listed above, the mark will be given according to whether a requirement is met, as well as the quality of the implementation. i.e., how responsive and interactive the interface is, how well it handles error situations. Code organization and coding style will also be taken into consideration.
- 20% of the marks will be for the advanced / creative features, the marks will be given according to the technical complexity of the feature, and implementation quality and the creativity of the feature.
- We are supposed to be web developers, NOT designers, so we won't be too picky about the artistic aspects of your work. You don't need to spend much time on the artistic design of your service. However, you DO need to keep your interface clean and clear, e.g., keep things properly aligned and organized, and easy to recognize and use.

That's it. Have fun!