Carnegie Mellon University
15-415/615 Database Applications
Fall 2015,
C. Faloutsos & A. Pavlo

HW7: Database Application

TAs: Dana Van Aken, Jinliang Wei,
Anna Etzel, Yujing Zhang, Jiaxi Xiong
Overview

• Design & implement a simple web application called ‘CMUYak’

• Today:
  • Application specs
  • Homework deliverables
  • Very brief intro to PHP
Data requirements

- **Users**
  - Username (2-50 characters)
  - Password (hashed 32 characters)

- **Posts**
  - Has title and body text
  - Have to record **when** they were posted
  - Have to record **where** (represented in integer x, y coordinates) they were posted
  - Posts can be “voted” by users
  - Posts may contain one or more hashtags

- **Hashtags**
  - tagname (2-50 characters)
Data requirements

Example:
The user “Smith” creates a new post with
- the title “Savage DB Research” and
- the content as “Winter is coming! #GOT”
- at location (0, 0).

This should be stored in the table that contains information about posts.
The application will also parse and extract the hashtag “#GOT” contained in this post and stored separately in the database without the number sign prefix (i.e., “#”).

The database should also contain a reference that identifies that the particular hashtag was used in the post.
Functionality requirements

1. Reset database
2. Create user account
3. Login
4. Add Posts with possible hashtags
5. Timeline
6. List all posts for a given user
7. Search for posts
8. Search for posts within range
9. Search for a tag
10. Delete a posts
11. Vote for a post
12. Unvote for a post
13. List most popular post
Functionality requirements

14. Recommend posts based on votes

• For user U recommend posts that are “voted” by like-minded users.
• Like-minded users of U are users who “vote for” what U would vote for.
• Rank them according to how many common votes a post has
• Don’t include posts that U already voted for
User recommendation example

Recommend posts to user 1

user 2, user 3, user 5 also vote for what user 1 votes for

user 2: post 3
user 5: post 3, post 5

*Note that it doesn’t include posts user 1 already voted for

sort by # votes for each post => recommend post 3, post 5
Functionality requirements

15. User statistics
   - # posts by the user
   - # votes by the user
   - The number of unique hashtags used by that user’s posts
     
     (For example, if a user has two posts, each of them contains “#CMU” in their content. We’d say that the number of unique hashtags used by this user is 1.)

16. Global statistics
   - List of $K$ posts with most likes
   - List of $K$ most active users
   - List of $K$ hashtags that appear the most often
   - List of $K$ hashtag pairs that appear together the most often
Example web application

http://www.contrib.andrew.cmu.edu/~jiaxix/cmuyak/
Homework Specifics

• Follow the design methodology from Lecture 18
• Organized in 2 Phases
  • Phase 1 – Design: due 11/11
  • Phase 2 – Implementation: due 11/30
Phase 1

You are free to come up with your own design choices as long as

- they follow the methodology
- they are reasonable
- you are able to justify unconventional choices
Phase-I

- description
  - req. anal.
    - top level I.F.D.
      - sys. anal.
        - task + doc forms.

Phase-II

- impl.+test.
  - code.
  - tests
  - user’s man.

- conc. mod.
  - schema.
  - task emul.
  - pseudo-code
Top level information flow diagram

registration form

T1-reg.

user record

external document (web forms)

tasks

internal document (database tables)

System boundary
Document + Task forms

Task forms and task list
- not required for this homework

Document forms and document list

- D1: registration form
- D2: login form
- D3: timeline form
- ...
- Dx: user record
- ...

external

internal
Document forms

D1: registration form
- username
- Password

Dx: user record
- username
- Password
Phase-I

1. Description
   - req. anal.
   - top level I.F.D.
   - sys. anal.
   - task + doc forms.

Phase-II

1. Impl. + test.
   - code.
   - tests
   - user’s man.

2. Conc. mod.
   - schema.
   - task emul.
   - pseudo-code
E-R diagram

- Specify cardinalities
- Think about weak/strong entities
- Justify unconventional choices
Relational schema

- Give the definition of the schema
- Give SQL DDL statements including constraints.
Phase-I

- description
  - req. anal.

- top level I.F.D.
  - sys. anal.

- task + doc forms.

Phase-II

- impl.+test.
  - code.
  - tests

- user’s man.

- conc. mod.
  - schema.

- task emul.
  - pseudo-code
Task emulation/pseudo-code

- No need to write pseudocode
- Simply give all SQL DML statements for all tasks
Phase 1: What to hand-in

- **Due 11/11**
- **Hard copy** (in class)
- **Electronic copy** (Blackboard)
Phase 2

• We provide an API in **PHP**
• Implements the web site functionality
• Has empty calls to the database
• write PHP code that
  1. wraps the SQL statements
  2. returns the output to the rest of the given code (PHP arrays)
• No need to provide user manual
Phase 2

• Unzip hw7.zip
• You need to edit 2 files
• config.php
  add your login & url info
• functions.php
  • Contains empty definitions of the functions that you have to implement
• We don’t you suggest other part of code base, but you may look into them and see how they function.
```php
<?php

// Connecting, selecting database
$dbconn = pg_connect("host=localhost dbname=publishing user=www password=foo")
    or die('Could not connect: ' . pg_last_error());

// Performing SQL query
$query = 'SELECT * FROM authors';
$result = pg_query($query) or die('Query failed: ' . pg_last_error());

// Printing results in HTML
echo "<table><tr>
while ($line = pg_fetch_array($result, null, PSQL_ASSOC)) {
    echo "|\n    foreach ($line as $col_value) {
        echo "|\n    }
    echo "|\n}
echo "</table>\n"

// Free resultset
pg_free_result($result);

// Closing connection
pg_close($dbconn);
?>
```


---

**Start connection**

**Issue query & read results**
PHP arrays

Array creation:

```php
$array = array(
    "foo" => "bar",
    "bar" => "foo",
);
```

Bulk insertion (like stack):

```php
<?php
$stack = array("orange", "banana");
array_push($stack, "apple", "raspberry");
print_r($stack);
?>
```

Securing your application

- **SQL injection**

  ```
  statement = "SELECT * FROM users WHERE name = '"" + userName + '"";
  ```

- Set name equal to `" or 'l'='l`

- The SQL statement that gets executed is

  ```
  SELECT * FROM users WHERE name = '"" OR 'l'='l';
  ```

- Results in un-authorized log-in!!!!

- Your code has to account for that

  - Hint: `pg_escape_string()`
Phase 2: What to hand-in

• **Due 11/30**

• **Website (IMPORTANT):** See hw7.pdf for details

• **Hard copy (in class):** ONLY new/changed code (save the trees 😊 )

• **Electronic copy:** A .zip with all the code
Homework 7: Architecture

Any host

Client

Browser

http

Web Server

Apache

cmuYak Web app

PHP

CMU Contributed Web Server

PostgreSQL Database Server

hw7 database

CMU AFS

andrew_id/www

Any host

Client

Browser

http

Web Server

Apache

cmuYak Web app

PHP

CMU Contributed Web Server

PostgreSQL Database Server

hw7 database

CMU AFS

andrew_id/www
Access to web server

• You will use the Computer Club Contributed Web Server
• Apache server + Postgres DB server
• Publishes *.php code in your AFS ‘www’ directory
• More details
  • http://www.club.cc.cmu.edu/doc/contribweb.php
  • HW7 description (read carefully)
Publishing your web app

• Please do the following ASAP and let us know if it doesn’t work!
  1. Sign up for the web server here
     http://my.contrib.andrew.cmu.edu
  2. Create DB user account here
     http://www.club.cc.cmu.edu/doc/contribweb/sql.php
  3. Unzip hw7.zip and copy contents on folder ‘cmupostly’ under your AFS www directory
  4. Edit config.php with your own db+server parameters
  5. Edit folder content permissions: chmod +rx
  6. Go to
     http://www.contrib.andrew.cmu.edu/~andrew_id/cmuyak
Questions?

- Come to **office hours** (5 TAs + 2 instructors)
- Post your questions on **blackboard**.