Assignment 2

75 pts

In this programming assignment you will create a simple micro-blogging Web application called Chattr. You will be provided a skeleton for implementing this application in PHP with a PostgreSQL backend database. This document uses the word *must* to mean that the described behavior is required and *must not* to mean that the described forbidden. The word *should* means that the described behavior is recommended but not required, and the words *should not* mean that the described behavior is discouraged. The word *may* means that the described behavior is allowed. *You will only be graded on required behavior.*

1 Required Functionality

1.1 Basic Functionality

When a user goes to the Chattr site, he must see a login screen:

Figure 1: Main page, shown when user requests `index.php`.

If a user already has an account and enters his user name and password correctly, he must be redirected to a page of their Chattr posts (`view.php`):

Figure 2: Message page generated by `view.php` when a user is logged in.

Entering a message into the text field and clicking "Submit" must add the message to the user's page.

The "Logout" button must take the user to the `logout.php` page, which will log him out and return him to the front page (`index.php` shown in Figure 1). If the user name and password are not correct, the user must be shown the login failure page:

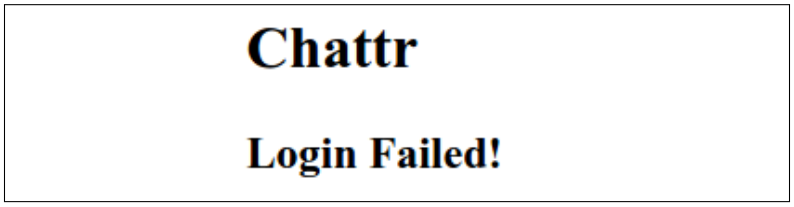

Figure 3: Login failure page generated by `login.php` when a user does not enter a valid user name and password.

If the user does not have an account, they can enter their desired user name and password on the front page, check the "New user" checkbox, and click submit. If the chosen user name does not exist, it must be created with the requested password and proceed as if the user had logged in (see Figure 2). If the requested user name is already taken, the user must be shown:

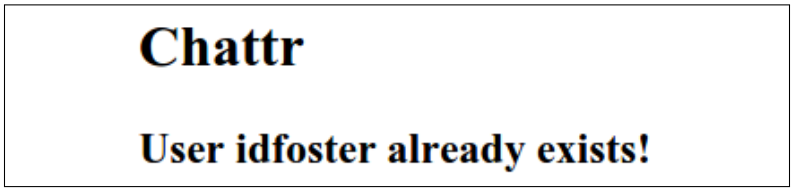

Figure 4: New user failure page generated by `login.php` when a user name already exists.

Finally, whether logged in or not, any user must be able view any other user's posts by navigating to `/view.php?user=XYZ` where `XYZ` is a user name:

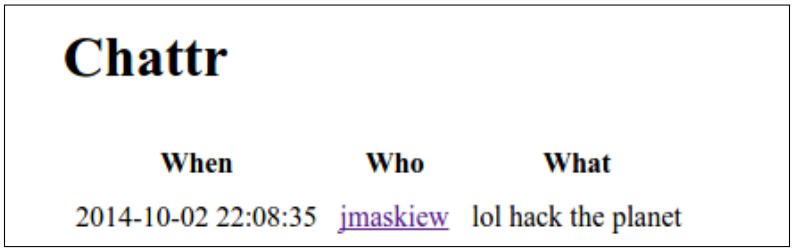

Figure 5: Message page generated by `view.php?user=jmaskiew` when viewing another user's page.

Messages on the `view.php` page may be displayed in any order. If a user navigates to `/view.php?user=XYZ` and there is no user with user name `XYZ`, the following error page must be displayed:

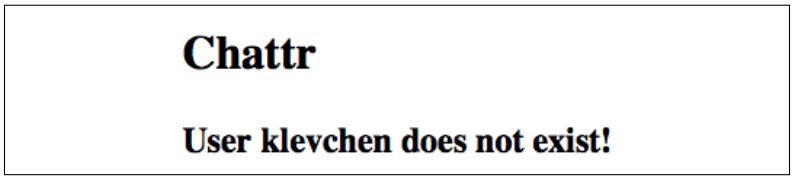

Figure 6: Message page generated by `view.php` when viewing the page of a non-existent user.

The option to post a message and the logout button must only be shown to a user when viewing their own posts and logged in. When viewing another user's posts or when not logged in, the posting form and logout button must not be shown (Figure 5). A user must be able to view another user's posts without logging in.

1.2 Additional Requirements

Your application must also satisfy the following additional requirements.

- **R1:** Both user names and passwords must be case-sensitive.
- **R2:** You must support user names containing letters, numbers, and underscores ("_").
- **R3:** You must support passwords to containing ASCII characters 33 through 126, inclusive.
- **R4:** You must support messages to containing ASCII characters 32 through 126, inclusive.
- **R5:** You must support user names and passwords up to 50 characters long.
- **R6:** You must support messages up to 500 characters long.

2 Undefined Behaviors

The following behaviors and situations are not defined and will not be graded. Your application can do whatever you want.

- Visiting the `index.php` or `login.php` page when already logged in.
- Visiting the `logout.php` page when not already logged in.
- Visiting the `view.php` page without specifying the `user` parameter in the URL.
- User names and passwords longer than 50 characters and messages longer than 500 characters.
- User names, passwords, and messages shorter than 1 character.

3 Application Skeleton

The application consists of five PHP scripts: the front page (`index.php`), the login script (`login.php`), view script (`view.php`), the post script (`post.php`), and the logout script (`logout.php`). These five files will be provided in the archive `hw2skel.tgz` available from the class Web page. You will turn in these five files, together with the database schema.

4 Database

The application must use a PostgreSQL database to store the data. The database must be called `chattr`. You will need to connect to the database from PHP. You can find a reference on using PostgreSQL from PHP here:

http://php.net/manual/en/book.pgsql.php

You must create the database tables you need for your application. You must submit an SQL archive of your database with your code. You can do this using the following command (which you must run as operating system user `postgres`):

```
pg_dump -s chattr > db.sql
```

This will dump only the table definitions but not the data. If your application depends on certain data already being in the database, you will need to run `pg_dump` without the `-s` option or edit `db.sql` manually.

5 VM Image

We have created a VirtualBox VM image configured with Apache 2.2.22, PHP 5.4.4, and PostgreSQL 9.1.13. You can download the VM image from:

https://drive.google.com/open?id=0B8j90Kz8o96VemZ5TnhlbHFaYTA

You should also download the PGP signature for the compressed VM image (`hw2vm.zip.sig`) from the class Web page and verify that it matches the file you downloaded: with `hw2vm.zip.sig` in the same directory as `hw2vm.zip`, run:

```
gpg --verify hw2vm.zip.sig
```

The VM is configured with two users: `student` and `root` with passwords `hacktheplanet` and `hackallthethings`, respectively. The VM is configured with the following services:

Host Port	Service
2222	SSH
5432	PostgreSQL
8080	Apache (with PHP)

We have already created a database called `chattr` in PostgreSQL. It has been configured so that the `student` account is the owner and can create the necessary tables. In addition, we have created a database account for the Chattr application named `chattr` with password `toomanysecrets`.

The HTTP document root for the VM is `/var/www/`. You can work directly on the VM or over SSH. To copy files from your computer to the VM, you can use:

```
scp -P 2222 -r /path/to/files/ root@127.0.0.1:/var/www
```

To copy files from the VM to your computer:

```
scp -P 2222 root@127.0.0.1:/var/www /destination/path
```

6 Submitting Your Solution

Your solution to this assignment consists of the five PHP files you modified (`index.php`, `login.php`, `view.php`, `post.php`, `logout.php`) and the database schema (`db.sql`). You may submit additional PHP files if your solution relies on them. Your solution must by submitted via email to `cs127f1@ieng6.ucsd.edu` by October 10, 2017, 10:00 P.M. PDT. It must be a `gzip`-compressed `tar` archive, signed with your PGP key and encrypted to the `cs127f1@ieng6.ucsd.edu` PGP key, which is provided on the CSE 127 Web page and has key fingerprint:

```
E1BF 1E04 1104 28DA 4F89 6543 B033 B3DC 10D3 7DBD.
```

You must send a plain email with the encrypted and signed archive file as an attachment. The email must have the subject "Homework 2 Submission" and the attachment must be named "`{PID}-hw2.tgz.asc`" (where `{PID}` is your PID). To create a `gzip`-compressed `tar` archive, copy the files you wish to submit to single directory, change into that directory, and issue the command:

```
tar -zcvf /path/to/archive/{PID}-hw2.tgz *.php *.sql
```

This will create an archive in the directory `/path/to/archive/` containing all the PHP and SQL files in the current working directory. To sign your submission with GPG:

```
gpg --encrypt --sign --armor -r cs127f1@ieng6.ucsd.edu {PID}-hw2.tgz
```

This will produce a file named "`{PID}-hw2.tgz.asc`" in the same directory. You will need to have imported the `cs127f1@ieng6.ucsd.edu` public key into your GPG keyring first.

7 Grading

This programming assignment that must be completed ***individually*** by each student. You may *not* discuss your solution with other students until seven days after the assignment deadline. Your solution will be graded on application functionality, as described in Section 1. We will use the same VM we provided you, so make sure that your solution works on the original image of the VM. You may consult any online references you wish. If you use any code you find online, please document it in a `README` file submitted with your solution (see Section 6).