Assignment 2
100 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.

1 Required Functionality

When a user goes to the Chattr site, they will see a login screen:

![Login Screen](image1.png)

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

If a user already has an account and enters their user name and password correctly, they will see a page of their Chattr posts:

![Post Page](image2.png)

Figure 2: Message page generated by `view.php` when a user is logged in.
The “Logout” button must take the user to the logout.php page, which will log him out and return him to the front page (Figure 1). If the user name and password are not correct, a user will see the login failure page:

![Chattr](image)

**Login Failed!**

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 would 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 will be created with the requested password and the user will be logged in (see Figure 2). If the requested user name is already taken, the user will see:

![Chattr](image)

**User idfoster already exists!**

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

Finally, whether logged in or not, a user can view other users’ posts by navigating to /view.php?user=XYZ where XYZ is the user name:

![Chattr](image)

When | Who | What
---|---|---
2014-10-02 22:08:35 | jmaskiew | lol hack the planet

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

To view his own page, a user may set the user parameter in the URL to his own user name, or omit the user parameter altogether. 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:

![Chattr](image)

**User klevchen does not exist!**

Figure 6: Message page generated by view.php when viewing the page of a non-existent user.
Note that the option to post a message is only 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 is not shown (Figure 5). Viewing another user’s posts must not require the user to be logged in.

2 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.

3 Database

The application must use a PostgreSQL SQL 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:


You must create the database tables you need for your application. You must create a SQL script to create your schema and submit it with your code. You can do this using the command (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.

4 VM Image

We have created a VirtualBox VM image configured with Apache 2.2.22, PHP 5.4.4, and PostgreSQL 9.1.13. The image will be distributed via BitTorrent; the torrent file hw2vm.torrent will be available on the class Web page. The credentials for the VM are student:hacktheplanet and root:hackallthethings. The VM is configured with the following services:

<table>
<thead>
<tr>
<th>Host Port</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>2222</td>
<td>SSH</td>
</tr>
<tr>
<td>5432</td>
<td>PostgreSQL</td>
</tr>
<tr>
<td>8080</td>
<td>Apache (with PHP)</td>
</tr>
</tbody>
</table>

PostgreSQL has been setup so that the local student account has access to the Chattr database. In addition, we’ve created a database account for the application: chattr:toomanysecrets.

The HTTP document root for the VM is located in /var/www/. You can work directly on the VM or over SSH. The VM image can be downloaded via a torrent available on the class website.

To copy files from your computer to the VM:

```
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
```
5 Submitting the Solution

Your solution to this assignment consists 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 be submitted via email to cs127f1@ieng6.ucsd.edu by October 16, 2014, 11:00 A.M. Pacific time. 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:

ED49 BC3B 8992 A1E0 D2DD 66DC A1EF 6BAE 7864 D1BD.

You may either send a signed and encrypted email, or send a plain email with the encrypted archive and separate signature.

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/ {YOUR EMAIL}-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 {YOUR EMAIL}-hw2.tgz
```

You will need to have imported the cs127f1@ieng6.ucsd.edu public key into your GPG keyring first.

6 Grading

Your solution will be graded on application functionality, as described in Section 1.

This is an individual programming assignment. You may not discuss it with other students until seven days after the assignment deadline. 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 5).

7 Change History

This is revision 3 of the assignment. This revision clarified that viewing another user’s posts is done by navigating to the view.php page with the user GET parameter set to the intended user’s name.

Revision 2 of this assignment removed the “Back” and “Users” links that appeared in several screen-shots on the first revision (Figures 1, 4, and 3). You will not be penalized if these links appear in your solution, nor will you be graded on the function of these links if they are present.