Homework 4: Social Network

Due date: February 13, 2017 at 11:59pm

This homework is the first in a series of homeworks in which you will build an increasingly sophisticated nano-blogging site. This site will eventually be a featureful, interactive web application including user registration and authentication, email integration for user verification, photo upload, and quasi-real-time updates.

For this assignment, you will implement the base features for the site in a Django application, focusing on the user authentication and general design. We recommend reading this (relatively short) document in its entirety before starting on your homework.

The learning goals for this assignment are to:

- Demonstrate mastery of many learning goals from hw1, hw2, and hw3 including:
    - High--quality, incremental development using the Git version control system.
    - Basic features of HTML and CSS and/or CSS libraries.
    - The high--level architecture of an MVC application, including basic features of the Django framework.
    - Manual validation of HTTP request parameters by a web application.
- Gain experience using iterative development, similar to what you would encounter using a modern agile software development process.
- Demonstrate a basic understanding of persistent data storage using the relational model, and gain experience using a modern ORM tool.
- Demonstrate a basic understanding of user authentication using the Django framework.

Specification

This section describes the main features for the social network site that you will implement during this assignment:

- Non-logged-in users may register for the site. Registering users must provide user name, first name and last name. Registering for the site leaves the user logged in as the newly registered user. (This function is similar to what is implemented in the course example.)
- Registered users may log in using their username and password.

- Logged-in users are able to post a short (160 characters or less) message. Posts, when displayed, show the following information:
    - the contents of the post,
    - the user who posted it (linking to the user's profile), and
    - the date and time the post was written.

- Logged-in users may view a 'global stream', displaying all posts that have been posted in reverse-chronological order (most recent first).

- Logged-in users may view profiles of other users (or their own profiles) when clicking on links provided with posted messages in the global stream. Profile pages contain information about a user (e.g. first name and last name) as well as all of the posts that user has made, in reverse-chronological order.

- Logged-in users are able to log out.

Implementation hints

You may end up with the something like the following pages:

1. A **login** page; displays a form to the user requesting username and password, linking to the registration page if the user wants to register instead.
2. A **registration** page; displays a form requesting a username, first name, last name, and password (with a password confirmation field).
3. A **global stream** page; lists posts from all users.
4. A **profile** page for a user; shows information about a user as well as all posts from that user.

Note that the built-in Django authentication system has a User model class[1] that which already has fields for first name and last name. Also note that Django models support dates/times fields, including sorting by date/time, and that, in `settings.py`, you can set which time your server uses.

Requirements

Your application must also follow these requirements:

- The empty URL (i.e. http://localhost:8000/) must route to the first page of your application.

- Your application should not use any hard-coded absolute paths (e.g. C:\Users\Bovik\foo.txt or http://localhost:8000).

[1] See https://docs.djangoproject.com/en/1.10/topics/auth/default/#user-objects for more info.

- Your application should run with **Django 1.10.x**.
- Your application should use the default Django database configuration based on a SQLite database file (named `db.sqlite3`) in your project directory.
- Your application should not crash as a result of any input sent to the server-side or because of any actions the user performs.
- You must have some design and general theme used throughout your application using (using Bootstrap or any other CSS framework is fine, as long as you cite it). The CSS must be separated out into (one or more) static file(s).
- Cite all external resources used and any additional notes you would like to convey to your grader in the `README.md` file. Be sure to let us know in `README.md` which versions of Python and Django you are using.

Grading criteria

For substantial credit your solution must clearly demonstrate the learning goals for this assignment, which are described above in the introduction.

Committing your work [10pt]

Specification fulfillment [20pt]

Validation [20pt]

Routing and configuration [10pt]

Coverage of technologies [40pt]

Turning in your work

Your submission must be turned in via Git and should consist of a Django application in the **`hw4`** directory. Name your project **webapps** and the application **socialnetwork**. Follow the same steps as for the last homework to create the initial Django project and application. The directory structure will look somewhat like the following (some files/directories omitted):

```
[YOUR-ANDREW-ID]/hw4/
    |-- webapps/
        |-- settings.py
        |-- urls.py
        |-- [etc.]
    |-- socialnetwork/
        |-- static/
        |-- templates/
        |-- models.py
        |-- views.py
        |-- [etc.]
    |-- manage.py
    |-- db.sqlite3
    |-- README.md
```