Homework 3: Data Models and Authentication

Due date: September 20, 2018

This homework is the second in a series of homeworks in which you will build an increasingly sophisticated nano-blogging site, **grumblr**. 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 **grumblr** in a Django application, focusing on the user authentication and general design. We recommend reading this document in its entirety before starting on your homework.

The learning goals for this assignment are:

- Demonstrate mastery of many learning goals from #hw1 and #hw2, 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 **grumblr** that you will implement during this assignment:

- Non-logged-in users may register for the site. Registering users must provide user name, first name, last name, and password. Registering for the site leaves the user logged in as the newly registered user.
- Registered users may log in using their username and password.
- Logged-in users are able to post a short (42 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.

Your solution might be a site with 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.

Your solution may vary slightly from this basic design as long as you include all the details from the specification (above) and the requirements (below). Note that the built-in Django authentication system includes a User model class[1] that already has fields for a user's first and last name.

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 hard-code absolute paths or urls (e.g. C:\Users\Bovik\foo.txt or http://localhost:8000).
- Your application should run with Python 3 and Django 2.1.x. This is the configuration that the graders will use when evaluating your work.
- 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 or as a result of any user actions.
- You must have some design and general theme used throughout your application using CSS (using Twitter Bootstrap or Materialize or any other CSS framework is fine). The CSS must be implemented in (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.

Grading criteria

For substantial credit your solution must clearly demonstrate the learning goals for this assignment, which are described above in the introduction. We will grade your work approximately as follows:

- Incremental development using Git [10pt]
- Fulfilling the **grumblr** specification [20pt]
  Proper input validation [20pt]
- Request routing and configuration in Django [10pt]
- Appropriate use of web application technologies in the Django framework [40pt]
- Design [0pt]

[1] See https://docs.djangoproject.com/en/2.1/topics/auth/default/

Turning in your work

Your submission should be turned in via Git and should consist of a Django application in the `homework/3/` directory. Name your project **webapps** and the application **grumblr**. The directory structure *might* look somewhat like (with some files/directories omitted):

```
[YOUR-ANDREW-ID]/homework/3/
    webapps/
        settings.py
        urls.py
        [etc.]
    grumblr/
        static/
            grumblr/
        templates/
            grumblr/
        models.py
        views.py
        [etc.]
    manage.py
    db.sqlite3
    README.md
```

Randall Munroe, *Tags*. https://xkcd.com/1144/