

Homework 4: Social Network UI (and Login)

Due date: September 27, 2021 at 11:59pm

This homework is the first in a series of homework assignments in which you will build an increasingly sophisticated nano-blogging site. This site will eventually be a featureful, interactive web application photo upload, and quasi-real-time updates.

This is a sketch of the main page of this application. (Your page should look different.)

The sketch shows a web application interface for 'Blog Master'. At the top is a blue header bar with the text 'Blog Master' on the left and a link 'Jeff Eppinger' on the right. Below the header, on the right side, are three links: 'Global', 'Follower', and 'Logout'. The main content area is titled 'Global Stream'. It features a 'New Post' section with a text input field and a 'Submit' button. Below this, there are two posts. The first post is by 'Jeff Eppinger' with the text 'I waited in line for 15 minutes to get my lunch –' and a timestamp '8/25/2021 12:10 PM'. It has a comment by 'Farnam Jahanian' that says 'Just come to my place. I'll feed you.' with a timestamp '8/25/2021 1:07 PM'. The second post is by 'James Garrett' with the text 'Can we have BBQ?' and a timestamp '8/25/2021 1:13 PM'. Below the posts, there is a 'Comment:' section with a text input field and a 'Submit' button. At the bottom, there is another post by 'Sujata Telang' with the text 'We need more seats in the library. The other day, I went in to find a book. No books, and no seats!' and a timestamp '8/24/2021 11:34 AM'. Below this post, there is another 'Comment:' section with a text input field and a 'Submit' button.

For this assignment, you will create the HTML pages for your site and implement a basic login function using Django's authentication package.

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 Django features.
- Gain experience using iterative development, similar to what you would encounter using a modern agile software development process.
- Demonstrate a basic understanding objects to represent forms.
- Demonstrate a basic understanding of hierarchical templates.
- Demonstrate a basic understanding of user authentication using the Django framework.

Specification

- All pages must have a common page layout that includes a header containing a site name and navigation links with id attributes as shown below. Each page must also have a page name.
- Login Page:** When first visiting the site, a login page is shown. Registered users log in using username and password and will then be shown the global stream page. There must be an href link to the register page. The page must have id attributes on elements, as shown below. (Django Forms will generate ids in this format for the form fields – you write the other ones.)

The sketch shows the Login Page layout. It features a blue header bar with the site name "Blog Master" and a "Register" link. The main content area has a "Login" title and a form with "Username:" and "Password:" labels, input fields, and a "Submit" button. Callout boxes indicate the following id attributes: id_site_name for the header, id_header_div for the header container, id_page_name for the page title, id_register_link for the Register link, id_username for the username input field, id_password for the password input field, and id_login_button for the Submit button.

- Register Page:** Non-logged-in users may register for the site. Registering users provide user name, password, confirm password, e-mail address, first name and last name. The register page will contain an href link to go back to the login page. Registering for the site causes the newly registered user to be logged in, displaying the global stream page. The register page must have id attributes on elements as shown in the sketch, below.

The sketch shows the Register Page layout. It features a blue header bar with the site name "Blog Master" and a "Login" link. The main content area has a "Register" title and a form with "Username:", "Password:", "Confirm:", "E-mail:", "First Name:", and "Last Name:" labels, input fields, and a "Submit" button. Callout boxes indicate the following id attributes: id_site_name for the header, id_header_div for the header container, id_page_name for the page title, id_login_link for the Login link, id_username for the username input field, id_password for the password input field, id_confirm_password for the confirm password input field, id_email for the email input field, id_first_name for the first name input field, id_last_name for the last name input field, and id_register_button for the Submit button.

- **Global Stream Page:** Logged-in users first are shown the global stream page which displays posts and comments made by all users:
 - There will be a text input and button on the global stream page to allow the logged-in user to create a new post. The ids for these elements are shown in the sketch.
 - Each post will have its own `<div>` element with `id="id_post_n"`, where n is the database record number for this post. Since this homework shows dummy data for posts, you can make up any integer value for n for now. Every post in your dummy data needs to have its own unique value for n .
 - With each post there will be a text input and button to allow the logged in user to comment on that post. Ids for these elements are: `id="id_comment_input_text_n"`, `id="id_comment_button_n"`, where n is the same as above.
 - Each comment will have its own `<div>` element with `id="id_comment_k"`, where k is the database record number for this comment. Since this homework shows dummy data for comments, you can make up any integer value for k for now. Every comment in your dummy data needs to have its own unique value for k . (You can have comment `<div>` elements inside their post's `<div>` elements, if you like.)

Blog Master Jeff Eppinger

[Global](#) [Follower](#) [Logout](#)

Global Stream

New Post:

Post by Jeff Eppinger – I waited in line for 15 minutes to get my lunch – 8/25/2021 12:10 PM

Comment by Farnam Jahanian – Just come to my place. I'll be there. -- 8/25/2021 1:07 PM

Comment by James Garrett – Can we have BBQ? -- 8/25/2021 1:13 PM

Comment:

g– We need more seats in the library. The library was closed. I went in to find a book. No books, and no seats! -- 8/24/2021 11:34 AM

Comment:

Within the <div> for each post and comment are the following three pieces of information:

- The first and last names of the user creating the post or comment will be shown in a link to that user's profile which must be an href with id="id_post_profile_n" or id="id_comment_profile_k", with *n* or *k* as above.
- The text of the post or comment must be in a with id="id_post_text_n", or id="id_comment_text_k", with *n* or *k* as above.
- The date and time of the post or comment must be in a separate with id="id_post_date_time_n", or id="id_comment_data_time_k", with *n* or *k* as above. The dates and times for posts and comments must be formatted m/d/y h:m as shown.

The screenshot shows the 'Blog Master' interface with a 'Global Stream' section. At the top right, there are links for 'Jeff Eppinger', 'Global', 'Follower', and 'Logout'. The 'Global Stream' section contains a 'New Post' form with a text input and a 'Submit' button. Below the form, there are three posts. Each post has a callout box pointing to its components: the user's name (id_post_profile_n), the date and time (id_post_date_time_n), and the text (id_post_text_n). The first post is by Jeff Eppinger, the second by Farnam Jahanian, and the third by Sujata Telang. Each post also has a 'Comment' form with a text input and a 'Submit' button. The comments have callout boxes for the user's name (id_comment_profile_k), the date and time (id_comment_date_time_k), and the text (id_comment_text_k). The first comment is by James Garrett.


- **Follower Stream Page:** This page displays only the posts made by users that the currently logged-in user is following. This page is otherwise the same as the global stream, except that the logged-in user cannot make new posts (but can still comment on posts). The page elements have the same ids as the global stream page. Note that users cannot follow themselves, so the logged-in user's posts do not appear here. No sketch of this page is shown.

- **My Profile Page:** The profile page for the currently logged-in user displays the name of the user, a short “bio” and the user’s profile picture. The user’s first and last names are shown in the page title. If a profile picture has been uploaded, the profile picture is in a `` with `id="id_user_picture"`. (If a profile picture has not been uploaded for the user, a default image is shown, also with `id="id_user_picture"`.)
 - This profile page will display a form to allow the bio text and a profile picture to be uploaded. The ids for these elements are shown below.
 - Also, on the profile page for the currently logged-in-user is a list of all the other users that this user is “following”. The list will link to each of these user’s profile pages. Each link must be in an href with `id="id_profile_link_x"` where *x* is the username of the followed user.

Blog Master
Jeff Eppinger
Global Follower Logout

Profile Page for Jeff Eppinger

id_page_name



id_user_picture

Jeff is a computer scientist. In the summer time he cultivates fig trees.

id_bio_input_text

Profile Picture: Choose File No file chosen

id_profile_picture

Submit

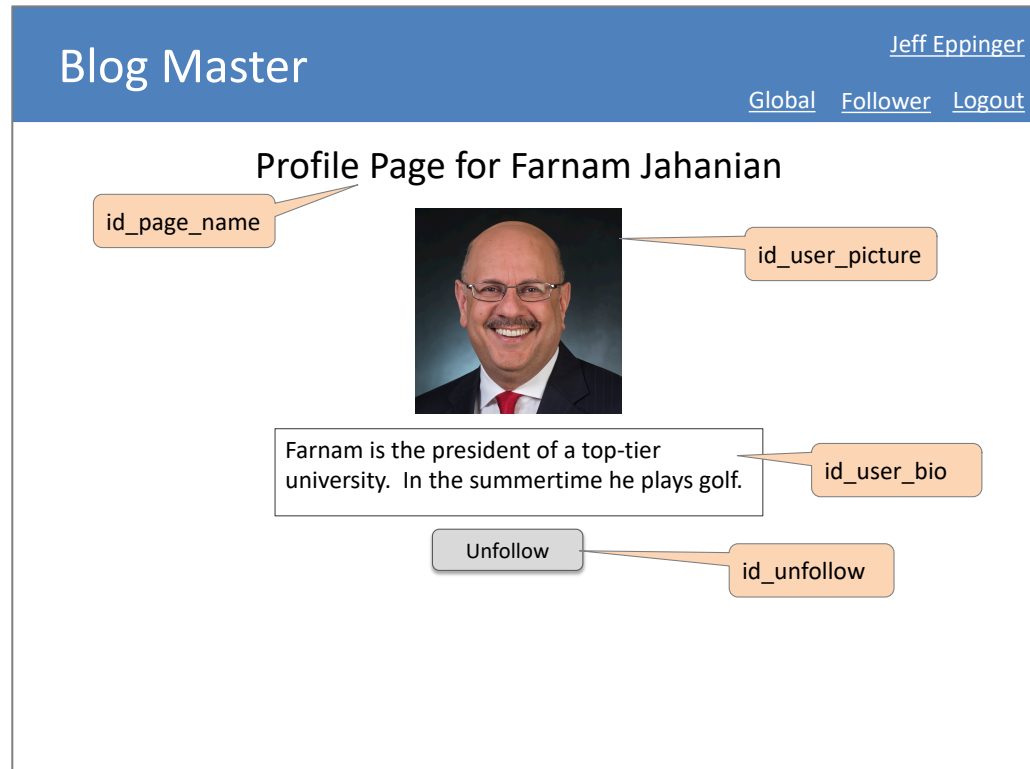
id_update_profile_button

You are following:

- James Garrett
- Farnam Jahanian
- Sujata Telang

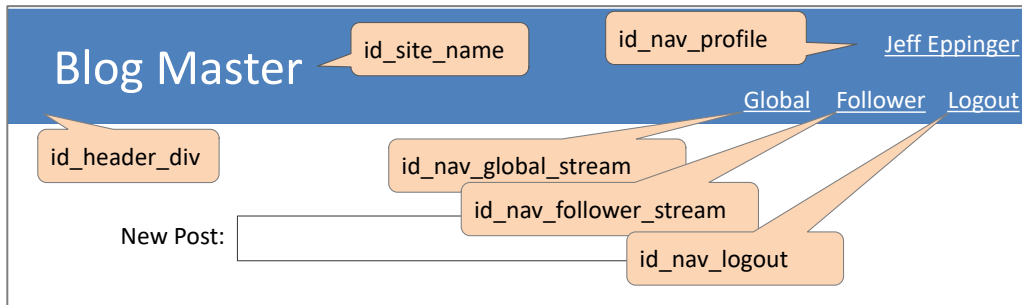
id_profile_link_x

- **Other Profile Page:** The “other” profile page, for a user other than the one that is logged in, also displays the name of the user, a short “bio” and the user’s profile picture, as in the “my” profile page, however, the user’s bio and picture cannot be edited.
 -
 - In addition, this page will also show a link to allow the logged-in user to follow or unfollow this other user. The id for the href to follow this other user is id="id_follow". The id for the href to unfollow this other user is id="id_unfollow". Note: only one of these links should be present on this page.



- Internally, you may use two different template pages for profiles, one “my” profile, another for “other” profile, or you may use one template page.

- All pages, other than login and register, must show the following navigation links:
 - the logged-in user's "my" profile page (id="id_nav_profile")
 - the global stream (id="id_nav_global_stream")
 - the follower stream (id="id_nav_follower_stream")
 - logout (id="id_nav_logout" – goes to login page when clicked)



- The first and last name of the logged-in user must be shown in the link to "my" profile page (id="id_nav_profile").
- Should there be validation errors for submitted form data (on the login and register pages), you must include the attribute `class="errorlist"` when formatting the error messages (as is done automatically, if using Django Forms).
- All pages should use template inheritance to provide the common features (the heading and the navigation)
- In HW#4, following links and buttons must work (so we can test all the pages):
 - The link to the register page (id_register_link)
 - The link to the login page (id_login_link)
 - The four navigation links (id_nav_*)
 - The submit buttons on the login and register pages (id_login_button and id_register_button)
 - The links to other user's profile pages (id_post_profile_*, id_comment_profile_*, and id_profile_link_*) – which all go to a dummy other user's profile page.

Implementation hints

You must make working [register](#) and [login](#) pages. Note that the built-in Django authentication system has a User model class¹ that which already has fields for username, e-mail address, first name and last name. You should use a Django form object to implement the register function.

You are not to make the other four pages “work”. You must display all the components of the UI, including at least one post and one comment on the two stream pages. These will be sample comments and posts. You are to provide sample pictures, bios, and follower lists on the profile pages. You may pass Python objects to your templated views to display the dummy data, or you may hardcode the dummy data into your pages.

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>).
- Your application must run with Django 3.
- You must use Django forms for processing the inputs from the Login and Register pages. These forms must be called `LoginForm` and `RegisterForm`.
- Your application should use the default Django database configuration based on a SQLite database file (named `db.sqlite3`) in your project directory for storing user information (which the Django authentication handles for you).
- You should not turn in the `db.sqlite3` file or files in the `migrations` folder. These files are specified in the `.gitignore` file that we pushed to the top of your repo, so unless you take some unusual action, these files will be ignored by git. When the grading script runs on your HW4 code, we will run “migrate” before “runserver” to create a clean `db.sqlite3` file.
- 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 [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 us in the `README.md` file, using the format described in the Homework #1 spec.

¹ See <https://docs.djangoproject.com/en/3.1/topics/auth/default/#user-objects> for more info.

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/  
    |-- forms.py  
    |-- views.py  
    |-- [etc.]  
|-- manage.py  
|-- README.md
```