Nathanael Han
CSCI E33-a

Final Project:

Poetica

For my final project I made a web application called Poetica. Poetica is a forum/social network site where users can share their poetry and view the poetry of other writers.

The features on my website include:

- A home page where all the posted poems can be viewed
- A page that lists the poets registered on the site
- A page where users can post a poem using a rich text editor
- Like/unlike functionality
- Users can delete their posts
- A page that displays the poems a user has liked
- Clicking each poem title leads to a page displaying that individual poem
    - On this page users can comment on the displayed poem.
- Profile pictures for each poet
- Profile pages for each user, displaying poems posted by the user
    - The profile page allows a user to post a "status" on their own page.
- Mobile navigation
- Pagination for the poets, liked, poems, and profile pages.
- A dark mode toggle button

Requirements:

[Django Ckeditor]

Log in credentials:

Superuser:

username: user1
email: user1@gmail.com
password: cse33spring

Other accounts:

password: cse33spring

Files:

Finalproject directory:

- In settings.py I configured the appearance of the Ckeditor

Users directory:

This app includes the user authentication functionality.

- views.py: contains the views for login, logout and registration
- templates:
    - registration.html
        - displays the form fields to create a new account
    - login.html
        - displays the form to log in to an account

Poetica directory/app:

Static files:

Assets folder: contains svg and png files used on the site

Stylesheets: dark_mode.css and styles.css

Poetica.js:

- Defines stylesheets — later used in the currentMode( ) and changeColor( ) functions.

- Initializes AOS (Animate on Scroll Library)
  - I used this library to animate the posts as the page is scrolled
  - One problem I ran into with using this library was that when I refreshed the page, the posts would disappear.
    - I was able to find a fix for this online to correctly load the AOS library (https://github.com/michalsnik/aos/issues/239)

- When DOM is loaded:

  - Mobile navigation element is selected
  - On click, display is toggled

  - Dark mode switch is selected
  - If the dark mode state isn't yet saved, it is given a default setting of false
  - The currentMode( ) function is called to check the theme setting and change the stylesheet accordingly

- Functions:

  - currentMode( ): uses conditionals to check the theme setting, then modifies the logo and stylesheet included in the html

  - changeColor( ): when the switch is toggled, the localStorage setting is updated, then the logo and stylesheet are modified in the html.

  - likePoem( ): references the view to retrieve the number of likes on a poem. After the like button is pressed, the function hides the like button and displays the unlike button. Lastly, the current number of likes is updated on the html.

    - unlikePoem( ): references the view to retrieve the number of likes on a poem. After the like button is pressed, the function hides the unlike button and displays the like button. Then the current number of likes is updated in the html.

Templates:

<u>layout.html:</u> the basic template for the pages on the site.

- References poetica.js
- In the head of this document, I include a style tag to hide the html. When the stylesheets are loaded, the html is set to visible. This prevents the pages from briefly displaying without styling, when the dark mode switch is toggled. (reference: https://bojanvidanovic.com/blog/css-solutions-fouc/)
- Includes the navigation links
    - Some links are only displayed if a user is logged in.
- Includes the bootstrap toggle switch for dark mode.

<u>index.html:</u> displays each poem posted to the website. Uses a for loop to iterate through the list of poems.

<u>Poem display:</u>

If the poster of a given poem has a profile picture, that photo is displayed. If not, a default graphic is shown. Each div also displays the poem's title (as a link to the poem's individual page) and the poet's name (as a link to the user's profile page). The poem's text is displayed with the "safe" tag, which allows the output from the rich text editor to be displayed correctly. If a user is logged in, the like/unlike buttons are displayed, along with the current number of likes the poem has.

If the current user is in the list of users who have liked the poem, the page shows the unlike button and hides the like button. If the user has not liked the poem, the unlike button is hidden and the like button is shown.

If the poem displayed was posted by the current user, the delete button is displayed.

Pagination:

The poems are paginated in groups of 8 posts. I implemented this using Django's pagination class.

likes.html: Displays the poems a user has liked.

poem_page.html: Displays an individual poem. On this page, if a user is logged in, a form to post a comment is displayed. In the comment_section div, using a for loop, the comments are displayed along with the user's profile picture and name.

poets.html: Lists the poets that are registered on the site (their username, profile picture, first name, and last name). Each link references to the user's profile page. Uses a for loop to iterate through the users in the database. The page is paginated by groups of 20.

post_poem.html: Displays the rich text editor for a user to post their poem.

profile.html: Displays the user's profile picture, username, name, and the poems they have posted. This page also provides a form to write a "status" message.

Models.py

Contains four models: User, Poem, Comment, and Status.

Poem model:
- body field, uses RichTextField field type from Ckeditor.
- Referencing the timestamp of the post, poems are ordered in reverse, with the most recent posts first.

Admin.py

- Registers the Poem, Comment, and Status models

Forms.py

- Contains the model forms for posting a poem, writing a comment, and writing a status message.

Urls.py

- Contains the URL patterns for the website.

Views.py

Functions:

index:

- Using pagination, provides all the poems in the database to index.html.
- Renders the homepage.

post poem:

- Renders the page with a form to post a poem.
- If the user submits a post:
    - Processes the form submission, saving the posted poem and its author in the database.
    - Redirects back to the homepage.

poem_page:

- Renders a page displaying an individual poem
    - The poem IDs are passed in from the URL templates tags.
- Provides the poem, the poem's comments, and the comment form as context
- If a comment is submitted:
    - The comment, the poem commented on, and the commenter are saved to the Comment model.
    - The view redirects to the poem's page.

like_poem:

- Adds a like on a poem for a current user.
- Provides the current number of likes on the poem using JsonResponse — used in poetica.js to update the count.

unlike_poem:

- Removes the like on a poem from the user.
- Provides the current number of likes on the poem using JsonResponse — used in poetica.js to update the count.

delete_poem:

- When delete button is submitted, the poem is deleted from the model.
    - Redirects to the homepage

      * This feature could use some work — when a post is deleted, the page will redirect from the current page back to the index page. If I had more time, I would have liked to use JS to hide the post, and remain on the same page.

poets:

- Renders the page displaying the list of poets registered on the site.
- Using pagination, provides this list of users as context to poets.html.

liked_poems:

- Renders likes.html
- Using pagination, provides the poems liked by the current user as context.

profile:

- Renders the profile page
    - The poet's ID is passed in from the URL templates tags.
- Provides the poems written by the current user to the profile page as well as the form to update the user's status.
- If a status message is submitted
    - Saves status and the current user in the Status table.
    - Redirects back to the profile page.
- If the user hasn't posted any status messages, a default message is provided to the profile page.
- Otherwise, the most recent status message is provided as context.