Assignment 5

Your own Twitter

In this last assignment, we will be putting together everything we learnt so far to build a social media microblog app, similar to Twitter. This will be especially useful to acquaint yourself with the use of databases which have covered in the last lecture. Instead of guiding you through each individual step, this exercise is more similar to a real-life situation: you have a list of routes and features that you want to include in your app, but you have to piece together how to implement them yourself. Some hints such as how to implement the tables in your database are included of course.

Furthermore, I have added some ideas for extra features that you can try to implement on top of the core functionality and which are a great opportunity to practice your skills learnt so far. These are not required for grading.

Routes:

- **Registration page:** Users can sign up here. They need to at least specify (a) first name (b) last name (c) user handle (d) email (e) password. Username and password must be unique to avoid any confusion
    - Use the wtform password field, so that passwords aren' t displayed as they are entered
    - The registration page has a small link below the submit button that links to the sign in page. It could say something like *"Already have an account? Sign up here."*
    - Password have to be hashed (using bcrypt)
    - *Optional but recommended:* use flash messages from lecture 6 to display a message if a user has tried to register with a username or the email that is already contained in the database
- **Sign in page**: Users sign in here using their email and password.
    - If they enter the correct credentials, then they are redirected to the homepage/feed
    - If the email is not in the database, then the user is redirect to the registration page
    - If the password is not correct, then the page stays the same
    - Use the wtform password field, so that passwords aren' t displayed as they are entered
    - Use bcrypt to check whether they matched the saved hashes

    - The sign in page has a small link below the submit button that sends the user to the registration page.
    - *Optional but recommended:* use flash messages from lecture 6 to display a message if it was the password that was incorrect or if the email isn’t found
- **New post page:** Users can create short posts that are shared with all other users on the network
    - Create a check in the route that ensures that the post is at maximum 140 characters long. The post is only accepted if that criterion is met. If the post is accepted, them the user gets redirected to the homepage
    - If the user isn’t logged in and they try to open the page, then they are redirected to the sign-in page
    - New posts are always associated with the user that created them in the database (more on that below)
    - *Optional but recommended:* display a flash message notifying the user if the message they tried to post is too long.
- **Profile Page:** This is a dynamic page URL that corresponds to “/profile/<username>”. It displays: (a) username (b) Full name
    - If the user handle is not taken / doesn’t exist in the database, then the user is redirected to the homepage / feed
    - If the user isn’t logged in and they try to open the page, then they are redirected to the sign-in page
- **Homepage/feed:** This is the main page. All messages posted by users on the network are displayed here to be seen by all others.
    - Display all posts using a Jinja2 loop. Each iteration of the loop should be a div item that contains the user handle and the post. Users can click on the user handle to get redirected to the user’s profile page. Posts should be posted by the time when they were created/posted
    - If the user isn’t logged in and they try to open the page, then they are redirected to the sign-in page

Database:

Your database should contain two tables. The specifications below are minimum requirements but you are invited to have fun and add other fields that you think could be interesting.

- **Users**: This table should contain but is not limited to (a) primary key (b) first name (c) last name (d) user handle (e) email (f) description
  - The Users table should have a relationship to the Posts table, including a back reference
  - Make sure you implement a function that prints out the values of an item of the table. This will be helpful for debugging. The string does not necessarily have to have the same formatting as the one we did in class.
- **Posts:** This table should contain but is not limited to (a) primary key (b) post content (c) length of post in characters (d) time when it was posted (e) user id: this should be a foreign key to the Users table
  - Make sure you implement a function that prints out the values of an item of the table. This will be helpful for debugging. The string does not necessarily have to have the same formatting as the one we did in class.

Other features:

- **Design:** Your website should use **Bootstrap**. The design doesn't have to be special, but there should be no bare html - just like you'd expect from a decent website. This especially includes components like: display of posts in the feed; forms for login, registration and new posts; the profile page.
- **Navigation bar:** All your pages should contain a navigation bar that has a logo, linking to the homepage/feed as well as following routes via text: (a) homepage (b) new post (c) sign-in (d) registration (e) my profile (f) logout
    - Depending on whether the user is signed in or not she/he should only see the navigation bar elements c & d or e & f. Implement this using Jinja2 conditionals checking for *current_user.is_authenticated*.
    - Consider adding the navigation bar in the end as you might have to repaste the code to to each of the html files (see next bullet on this)
    - *Optional but recommended:* Base templates are a feature of Flask and Jinja2 that allow you to write html that is used frequently in a single html page that is then automatically adopted and extended by other pages of your website. A great example of this is the navigation bar, which generally stays the same on each page. Have a look at this page starting at "Template Inheritance"
    - *Optional but recommended:* Try to format your navigation bar as you would expect on a page like twitter: Logo and general links should be on the left side; registration and login or logout should be on the right side.
- **Database initialisation and Population script:** Create a python script that is not executed by your flask app (but should be in the same folder) that includes some SQL Alchemy commands for initiating and populating the database. This feature is optional but it will save you time and let you easily verify whether features are working

Optional features to tinker around with:

These features are not required for the full grade. I invite you to implement some of them to practice what we have covered in the course so far. Some will be useful for your final projects and they are a great opportunity for practicing. Some subtasks of each optional feature are recommendations, so you don' t have to implement each feature fully. The features are roughly ordered from easy to hard to give you a sense of how much time you might need.

- **Edit profile page:** *If users are on their own* profile page they see a button that says "edit profile" . If they click it they get taken to a new route with a form containing fields that are editable, but already are filled in with their *current* information. On successful submission the new data overwrites the old data and they get redirected to their profile page.
  - You will have to do some googling for some parts of this including how to set default values in a form. You can take a first look on options to update a value in a database here
- **Profile pictures:** Users can upload pictures of themselves that are displayed on their profile and also next to their name for any of their posts.
  - Save images in the static folder as we have done so far. Include the path to the file in your database.
  - Remember to consider cases where a user does not upload a profile picture. Either include something like a "None" value in your database or include a default profile picture like this one that you have added to your app initially
  - You might want to resize the pictures before saving them (or save them first, then open them, resize, and overwrite the saved file). Here' s a first pointer on how to resize images.
- **Post pictures:** Users can upload pictures with their posts.
  - See comments on Profile pictures
- **Delete post:** If a post is by the user that is currently logged in then she/he should see a small link where she/he can delete the post.
  - This link should be rendered dynamically with Jinja2. Check for the condition that the id of the user that created the post is the same as the id of the user logged in
  - Here' s a first pointer on how to delete elements from a database with SQLAlchemy
- **Like button:** Users can like/ "heart" / "clap" posts by other users in the feed
  - Create a new table in your database called "Likes" (or any other name of your choice). It should include (a) primary key (b) post id: this should be a

foreign key - don’ t forget to include a back reference in your posts table (c) user_id: this column should be included but can additionally also be also set as a foreign key with a back reference in the users table

    - Implement a checking system that makes sure that a user can’ t like a post twice
    - Include a small icon for likes in the feed, if clicked the page will refresh. Consider changing the icon displayed depending on whether the user has already liked the post or not.
    - Include a button on the users profile that redirects to a separate route where users can see a feed of posts that that user has liked.
- **Following other people:** Users can follow each other
    - Add a table called Followers that includes (a) primary key (b) id of person that follows (c) id of person being followed. (b) and (c) should be foreign keys to the Users table
    - Include metrics on “followers” and “following” on the profile of a user. Optionally make these clickable links which display a list of the clickable names of people following person x / being followed by person x.
    - Create a customised feed where you only see the posts of users that you follow. You could implement this by merging posts with the Followers table. Here’ s a first pointer how this could look like.