proj2
=====

Twitter Clone:

App link:

http://fritter-zhar.rhcloud.com/

App description:

My app lets users log in to their accounts, post new freets, view freets posted by all users, and edit/delete the only the freets that they posted.

Design Challenges:

There were some problems I ran into with implementation of the edit/delete function. To connect to an edit/delete page to change a specific post, I needed a way to identify the post to access it in the database. I explored two options, one to use a href link and two to use a form. Originally, for each post the user wrote I included a href link that included the _id of the post. I used the _id because it is unique to each post. Then, when the user clicks the link, the app opens an edit page that uses the _id from the link to find the post in the database to make changes. However, one bad part of this implementation was that the _id was now visible on the link, publically exposing the database field. I alleviated this problem by wrapping the displayed post in a form along with a edit/delete button. Inside the form, I was able to store the post _id in a hidden input tag so I could transfer it to the edit page. (https://github.com/6170-fa14/rujiazha_proj2/blob/master/views/users/index.ejs#L27).

Another problem is to know which user is making changes to posts or creating new posts. I considered taking the username from the html page. However, this is not a secure practice because html elements can be changed externally. Therefore, I created a user field in the current session when a user logs in. (https://github.com/6170-fa14/rujiazha_proj2/blob/master/routes/index.js#L29) I can access this user field to know which user is making changes. Additionally, having this field also allows me to easily distinguish between posts made by the user and posts made by others, which I need to know which posts to allow for edits. (https://github.com/6170-fa14/rujiazha_proj2/blob/master/routes/users.js#L17).

In order to ensure that only users with accounts can use the app, I included a function, checklogin, that checked if a user is logged in. If no user is logged in, then nothing is accessible but the homepage and the make new account page. I add a user field to the current session when a user logs in, so I check if the a user is logged in by checking in req.session if the user field is present. If someone is trying to access an internal page without first logging in, the app redirects them to the home page

instead of throwing an error, making the app easier to use. I originally wrote out the function for all get methods, but for modularity I set the function externally and referred to it from every method instead. (https://github.com/6170-fa14/rujiazha_proj2/blob/master/routes/users.js#L5)

Also, I noticed that a user remains logged in after closing and reopening the app on the same browser. In order to ensure account security, and allow the user the choice continue to be logged in after closing the app, I added the log out function. I implemented logout by calling req.session.destroy so that the app will revert to the path it takes when no user has been specified as a field. (https://github.com/6170-fa14/rujiazha_proj2/blob/master/routes/users.js#L22)

In the app, I wanted to avoid redundancies in usernames, and I wanted passwords to be secure. Therefore, in the signup page I made sure that a user could not take a username already in the database and that the password had to be 8 characters. Additionally, I checked for accidental blank submissions by not allowing the username field to be blank. (https://github.com/6170-fa14/rujiazha_proj2/blob/master/routes/index.js#L24)

PROJECT 2, PART 2

Grading Directions:

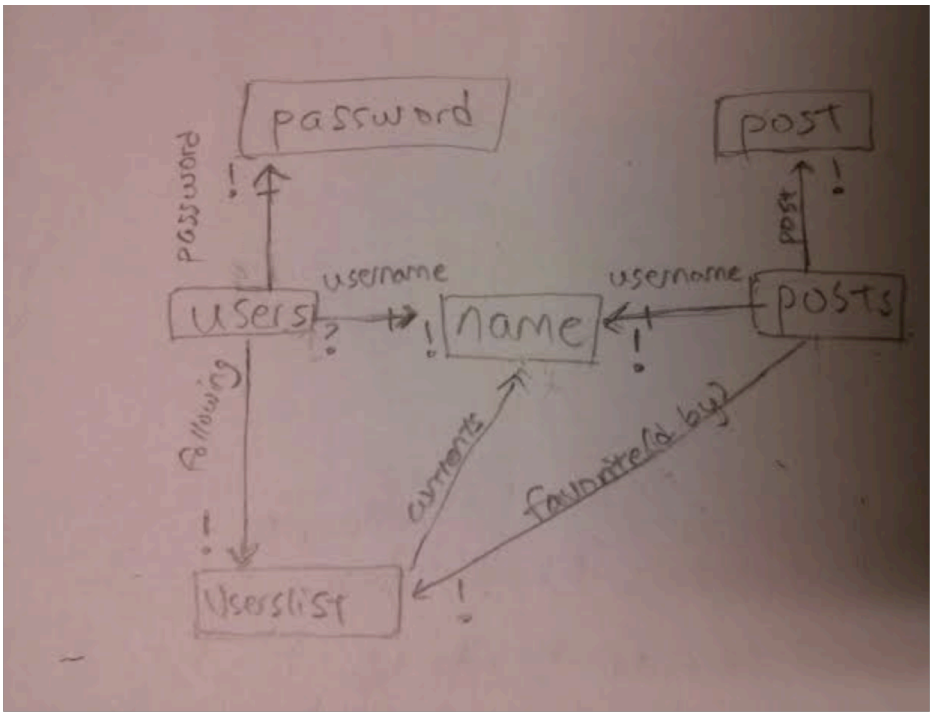
My additional feature allows users to favorite posts they can see and see how many favorites each post has. A user can both view posts of users they follow and favorite or unfavorite each post, including their own. Each post also displays the number of favorites it has, followed by the name of the users that favorited it.

I allow a user's tweet to submit only if there is something written. This way, there will not be unintentional blank posts. https://github.com/6170-fa14/rujiazha_proj2/blob/master/routes/users.js#L78

I modified the checklogin function to return true or false. By doing so, I can put in an if-else statement in each method to make sure that code to be executed when a user is logged in is only run if a user is logged in. The precaution taken here is necessary because the code after res.redirect() can be run before redirect is executed. https://github.com/6170-fa14/rujiazha_proj2/blob/master/routes/users.js#L8

In my follow users page, I categorized users into the users that are followed from the users that are not followed, so it is easier for the user to see which users to follow/unfollow. Upon following a user, the page immediately updates and places the user in the appropriate category. https://github.com/6170-fa14/rujiazha_proj2/blob/master/routes/users.js#L56

Data Model:



Design Challenges:

One challenge was on form submissions for following or unfollowing users. I could have the action for each form point to a different handler in users.js, or I could have the actions of both forms have the same name. Initially I had two different actions point to different handlers, but because both follow and unfollowing a user involve changing the same field in the users collection, the handlers were very similar and had repetitive code. To avoid this issue of repetitiveness, I used the same action. For each form, I added the name tag to the submit button of either "follow" or "unfollow". This way, I can use the same handler for both follow and unfollow, and I can check which kind of submit button was pressed for where the code differs. https://github.com/6170-fa14/rujiazha_proj2/blob/master/routes/users.js#L63

Another challenge was determining the structure for storing the favorited information. I could either store the favorites of each user under the users collection or under the posts collection. To store under the users collection would mean that the whole users collection would be searched for each post to figure out which users favorited each post. Storing under the posts collection would make it much easier to access the users that favorited that post. Therefore, I chose to store the users that favorited a post under the posts collection. https://github.com/6170-fa14/rujiazha_proj2/blob/master/routes/users.js#L21