Strider Web Back-end Assessment - 2.0

- Instructions
  - Test version: 2.0
  - Aim to spend no more than 8 hours on this test. If you go over 8 hours, write in the README how many hours you spent in total. Do not include pre-work to get your environment set up in this total.
  - Do not post this test anywhere on the public internet (including Github, Gitlab, etc.)
  - We recommend using the programming language and frameworks you are most familiar with to solve the problem
  - Make sure to include instructions in the README for the reviewer to be able to set up your project in his or her local environment.
  - Reviewers will only see what is in your Zip file, so make sure that any comments that you want to make about the project are left in the Readme.
  - The code you write should be as production-ready as possible given the time constraints. That means you should add comments where appropriate, write automated tests as you see fit, and write understandable & well-architected code that others can understand.
  - **Submission**
    - Save your whole project as a ZIP file, with title in the format "[firstName]-[lastName]-[test-version]-web-back-end.zip" e.g., "John-Smith-1-7-web-back-end.zip"
    - Upload your project to Dropbox or Google Drive, and send the link to the zip file in an email with title and recipient below
      - Title: "[firstName] [lastName] Web Back-end Submission"
      - Send to: submissions@onstrider.com
    - Make sure permissions on the file/folder are set to "Anyone with this link can view"
- What help you can get
  - You may use the internet, Google, StackOverflow, libraries, etc to solve the project
  - You cannot reference other people's solutions for the same or similar test
  - You cannot get outside help from anyone
  - You can ask questions about requirements, submission, etc to your Strider recruiter
- Evaluation criteria
  - Fulfilling all of the requirements correctly
  - Following test instructions well
  - Architecture choices
  - Cleanliness of code
  - Scalability of solution
  - Use of best practices
  - Quality of comments and README
  - Quality of commentary on critique and discussion
  - **Efficiency of database queries**

Project Description

Overview

The Project Manager you work with wants to build a new product, a new social media application called Posterr. Posterr is very similar to Twitter, but it has far fewer features.

Posterr only has two pages, the homepage and the user profile page, which are described below. Other data and actions are also detailed below.

Pages

Homepage

- The homepage, by default, will show a feed of all posts (including reposts and quote posts).
- There is a toggle switch "All / following" that allows you to switch between seeing all posts and just posts by those you follow.
- New posts can be written from this page.

User profile page

- Shows data about the user:
    - Username
    - Date joined Posterr, formatted as such: "March 25, 2021"
    - Number of followers
    - Number following
    - Count of number of posts the user has made (including reposts and quote posts)
- Shows a feed of all posts the user has made (including reposts and quote posts)
- Shows whether you follow the user or not
- Follow/unfollow actions:
    - You can follow the user by clicking "Follow" on their profile
    - You can unfollow the user by clicking "Unfollow" on their profile
- New posts can be written from this page

More Details

Users

- Only alphanumeric characters can be used for username
- Maximum 14 characters for username
- Do not build authentication
- Do not build CRUD for users

- When/if necessary to make your application function, you may hard-code the user. For example, you may need to do this to implement creating new posts, following, etc

Posts

Posts are the equivalent of Twitter's tweets. They are text only, user generated content. Users can write original posts and interact with other users' posts by reposting or quote-posting. For this project, you should implement all three — original posts, reposts, and quote-posting

- Users are not allowed to post more than 5 posts in one day (including reposts and quote posts)
- Posts can have a maximum of 777 characters
- Users cannot update or delete their posts
- Reposting: Users can repost other users' posts (like Twitter Retweet)
- Quote-post: Users can repost other user's posts and leave a comment along with it (like Twitter Quote Tweet)

Following

- Users should be able to follow other users
- Users cannot follow themselves
- Following and unfollowing will be done only on the user profile page

Extra feature: Search

Only work on this extra feature if you have enough time to complete the required features and get through all three phases of the interview.

- Implement a search feature that allows users to efficiently search through all posts
- This search feature should not return reposts
- This search feature should return quote posts, but only if the search matches the additional text added (do **not** return matches from the original post that was quoted on top of)
- (For Phase 2) This search feature should return reply-to-posts

Phase 1, coding

Estimated time: 7 hours

- Build out a RESTful API and corresponding backend system to handle the features detailed above. This RESTful API would communicate with single page JS app. This API you build should enable all the features on both of the pages.
- Do not implement additional features beyond what is explained in the overview.
- Write some automated tests for this project.

- Do not build a front-end.

Phase 2, planning

Estimated time: 30 minutes

The Product Manager wants to implement a new feature called "reply-to-post" (it's a lot like Twitter's These are regular posts that use "@ mentioning" at the beginning to indicate that it is a reply directly to a post. Reply posts should only be shown in a new, secondary feed on the user profile called "Posts and Replies" where all original posts and reply posts are shown. They should not be shown in the homepage feed.

What you need to do:

- Write down questions you have for the Product Manager about implementation.
- Write about how you would solve this problem in as much detail as possible. Write about all of the changes to database/front-end/api/etc that you expect. You should write down any assumptions you are making from any questions for the Product Manager that you previously mentioned.

This should be added as a section called "Planning" in the README.

Phase 3, self-critique & scaling

Estimated time: 30 minutes

In any project, it is always a challenge to get the code perfectly how you'd want it. Here is what you need to do for this section:

- Reflect on this project, and write what you would improve if you had more time.
- Write about scaling. If this project were to grow and have many users and posts, which parts do you think would fail first? In a real-life situation, what steps would you take to scale this product? What other types of technology and infrastructure might you need to use?

This should be added as a section called "Critique" in the README.