PIXLEE

MINI-PROJECT: INSTAGRAM COLLECTION

Write an app that takes a hashtag, start date, and end date, and collects submissions from Instagram

- You'll want to use the following endpoint: https://api.instagram.com/v1/tags/{tag-name}/media/recent?access_token=ACCESS-TOKEN
- You can test with an Instagram sandbox access token. When ready, please ask us for a valid access token that allows access to the tags endpoint.
- More information is provided here: https://instagram.com/developer/endpoints/tags/

The app

- The app should paginate through the endpoint and collect content whose tag time is in between the start and end dates
- The photo's tag time is defined as the time the hashtag was tagged with the photo. (Ordinarily, the photo created time)
- If the caption does not contain the desired hashtag, but the submitter of the photo posts a comment with the desired hashtag afterwards, the photo will be included in the pagination
- The tag time in the above case is the time the comment was posted
- The app should be available as a web service
    - Backend: Provide an API that accepts a POST to create a collection and GET to retrieve content
    - Frontend: Create a web page that provides users with an interface to create collections and view them
- The data should then be stored in a permanent data store. (Recommendation is to spin up a Postgres server)
- The data stored is at the discretion of the applicant (but the tag time should be included)

To complete the assignment, you have a choice between adding features on a frontend perspective, or a backend perspective:

- FRONTEND: write a web visualization for the collected content in a js framework of your choice. The user should be able to view the photos, the Instagram username, navigate to the native Instagram link of the photo, and play videos. The user should be able to paginate or load more to view further photos.
- BACKEND: build robust fault tolerance. If the system dies, how do you recover from the last point the collection occurred? How to minimize the amount of API hits, and thus minimize the chance of hitting the rate limit on the token? How to handle multiple, parallel processing of many collection requests? How to elegantly handle the case of your end date being in the future, or a lack of an end date?

SUBMISSION:

- The code should be put on github with a README that provides instructions on how to run the application
- There should be a live example that is provided to us (Recommend you use heroku)
- Please write up a short summary explaining
    - what you did (frontend or backend)
    - why you made some choices you made (what features did you add and why?)
    - what you would improve next if you had more time (what's the next most important feature?)