Homework 4: Pennstagram

Due Friday, October 19th at 11:59 PM on Canvas. Make sure to start early!

About

In this homework assignment, we will be building a REST API and its corresponding frontend (i.e. HTML pages) for Pennstagram, a collaborative web application for uploading images with captions.

This assignment will be graded manually, so specific implementation details are up to you. However, we will be assessing API design considerations that we discussed in lecture.

Initializing a Flask application

For the first part of this assignment, you will have to build a Flask application from scratch (you are allowed to consult the code provided with lecture).

TODO: Add a route available at "/api" that returns a "hello world" to the user. The methods for this route can either be GET or POST.
**TODO:** Display an HTML page at "/" that has the title "Pennstagram" in an <h1> tag.

### Retrieving images

All interfaces with the application should be done through routes that start with "api" (e.g. /api/images).

**TODO:** Create a GET route for retrieving all the images. Add an optional URL parameter for limiting the number of images to retrieving (e.g. retrieve the last 5 images).

### Creating a database

We should have persistent, server-side storage for our images.

**TODO:** Create a database that holds an array of JSON objects pertaining to each image that is uploaded.

Images that are uploaded should have:

1. a caption
2. a URL for the image source
3. a like count
4. username of the uploader
Uploading images

Image hosting is not something we’ll build for this project. Instead, we’ll rely on a service named Imgur for hosting. We can upload images and simply copy/paste the image URL by right clicking on the uploaded image.

Building a dynamic frontend

We want our users to be able to see the images they post. Build a frontend in /templates that allow us to see all the uploaded images one by one. *Hint: you will need a for loop within your template.*

Uploading will be done through curl or through a GUI debugger like Postman. You do NOT need to build a GUI interface for uploading.

**TODO:** Display the images that are uploaded into the database as well as their associated metadata (e.g. caption, uploader).

**TODO:** Implement a like button using an `<a>` tag with an `href` attribute that makes a redirect to some GET request. You will have to create another route for handling “liking” a photo. *(Hint: think about how we made an edit using a GET request and URL parameters in lecture).*
Rendering images

Displaying images in HTML requires the image source - an image location on the internet. Once we have an image URL, we can render the image by creating an `img` tag in HTML, supplying an attribute named `src` (source).

```html
<img src="{{IMAGE URL GOES HERE}}" />
```

Submission

Create a `README.md` file explaining how to run your server and documenting the routes you created (along with their HTTP method). Upload a `.zip` file onto Canvas containing all the code for your project.