Assignment: User Signup

Your assignment is simply to create a user signup form where users provide a username, password, and, optionally, an email address. You must have an additional field to verify the password (the user must retype it exactly). Then you will validate the input and either redirect them to a welcome page, or provide error information if any input is invalid.

Project Setup

To get started on this assignment, first create a repo (local and remote) called `user-signup` and then activate your virtual environment using `source activate flask-env`. If you need reminders on how to set up a new repository, review the instructions for **Web Caesar (../web-caesar/#git-repository-setup)**.

Next, get your project file structure set up by adding a `main.py` file and adding a directory for the template files you will make below.

```
(flask-env) $ touch main.py
(flask-env) $ mkdir templates
```

Your Task

Your page does not have to look exactly the same as **our version (https://launchcode-demos.appspot.com/signup)**, but you must have all the same functionality. Here's a summary of what is required:

1. If the user's form submission is not valid, you should reject it and re-render the form with some feedback to inform the user of what they did wrong. The following things should trigger an error:

    - The user leaves any of the following fields empty: username, password, verify password.
    - The user's username or password is not valid -- for example, it contains a space character or it consists of less than 3 characters or more than 20 characters (e.g., a username or password of "me" would be invalid).
    - The user's password and password-confirmation do not match.
    - The user provides an email, but it's not a valid email. Note: the email field may be left empty, but if there is content in it, then it must be validated. The criteria for a valid email address in this assignment are that it has a single `@`, a single `.`, contains no spaces, and is between 3 and 20 characters long.

2. Each feedback message should be next to the field that it refers to.

3. For the username and email fields, you should preserve what the user typed, so they don't have to retype it. With the password fields, you should clear them, for security reasons.

4. If all the input is valid, then you should show the user a welcome page that uses the username input to display a welcome message of: "Welcome, [username]!"

5. Use templates (one for the index/home page and one for the welcome page) to render the HTML for your web app.

✶ Note

While we've covered how to specify different input types than just `text` (e.g., `password` and `email`), for this assignment **do not use the email input type**. Instead, just use `text`, which does not do any client-side validation. This will enable us to check that the server side validation is working by letting errors through the client side. You should, however, use `type='password'` for the password and password verification inputs, to hide the characters typed (this input type does not include any additional validation).

Submit

To turn in your assignment and get credit, follow the **submission instructions (../)**.

 Bonus Missions

1. Use a **Regular Expression (https://docs.python.org/3/library/re.html)** (also called a regex) to validate the input fields. See this article about **Regular Expressions (https://en.wikipedia.org/wiki/Regular_expression)** for more information.

2. If you haven't already, refactor your templates so that you are using a base template for the boilerplate HTML that is needed for both the home page and the welcome page, and template extensions for the additional HTML your app uses.