INFO3180 - Lab 4 (20 Marks)

File Uploads

Due: February 28, 2021 at 11:55pm

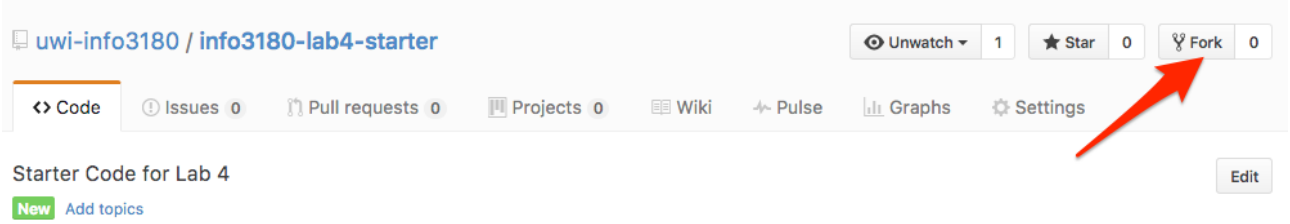
As always your lecture slides and the various demonstrations done at the end of your labs and in your tutorials will be helpful in completing your labs. The following resources may also be helpful:

- Flask-WTF File Uploads - https://flask-wtf.readthedocs.io/en/stable/form.html#module-flask_wtf.file
- CSS Grid - https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout/Basic_Concepts_of_Grid_Layout

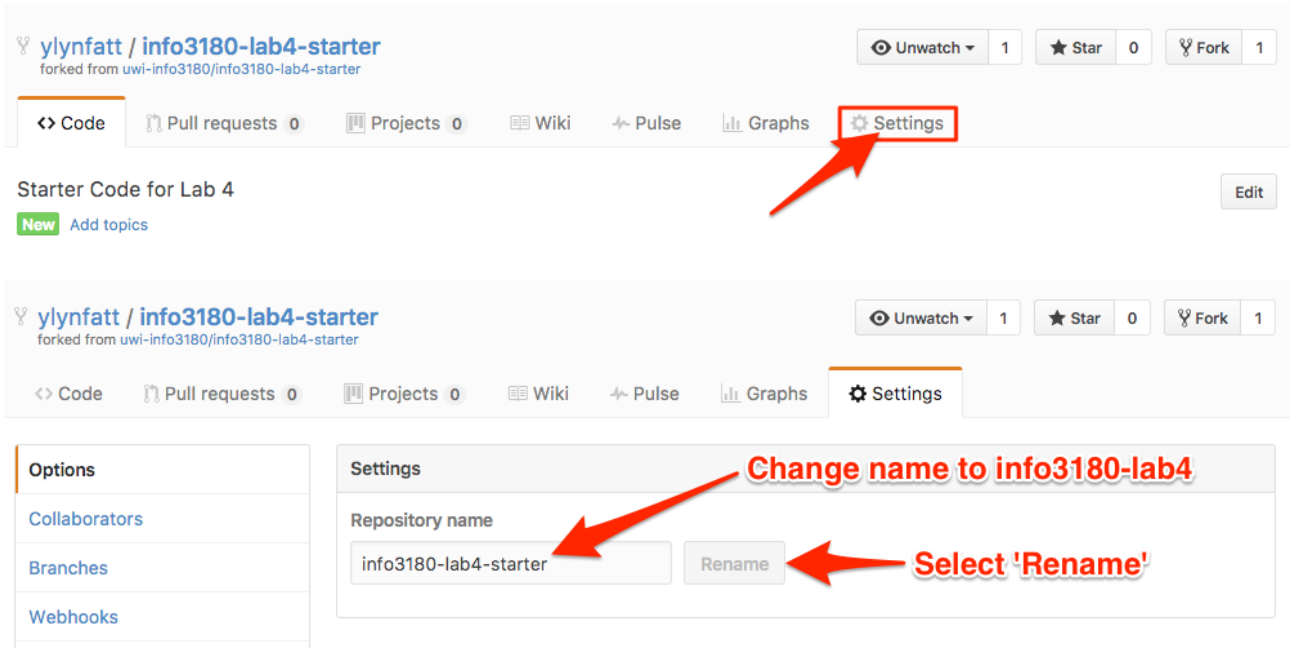
Exercise 1 - File Uploads

Start with the starter code at: https://github.com/uwi-info3180/info3180-lab4-starter

Fork the repository to your own account



In github.com rename it by going to settings and changing the name to info3180-lab4



To start working on your code clone it from your newly forked repository for example:

```
git clone https://github.com/{yourusername}/info3180-lab4
```

Setup your upload folder and Create your Upload form

Note: Ensure you create your virtual environment as you have done in previous labs and activate it. Also ensure you take a look at the dependencies in your **`requirements.txt`** file and install these libraries for the application from your **`requirements.txt`** file by using the **`pip install -r requirements.txt`** command.

1. Add a configuration option in your **`config.py`** file for your **`UPLOAD_FOLDER`**. This should point to your "**`uploads`**" folder so that Flask can know what folder to store your uploaded files. **Note:** The folder has already been created for you in the root of your application.

2. Create a **`forms.py`** file and create a Flask-WTF form class named "**`UploadForm`**" with a file upload field only. Ensure that you create the necessary form validation to make the file upload field **required** and allow **only image files** (e.g. jpg and png files) to be uploaded.
3. Import that form in your **`views.py`** file.
4. Alter the **`upload`** view function so that:
    a. You instantiate the form class you created in Step 2 and pass it to your **`upload.html`** template file via the **`render_template`** function when a **`GET`** request is made.
    b. You validate the form data on submit (using the appropriate Flask-WTF function) when a **POST** request is made.
    c. If the validation passes, save the file to your upload folder using the configuration option you created in the first step. **Note:** Also ensure you use the **`secure_filename()`** function before saving the file. (Look at the top of your *views.py* file to ensure that it has already been imported.)

5. Open the **`upload.html`** file and ensure you add the appropriate **`method`** and **`enctype`** attributes to the **`<form>`** tag to allow for file uploads. Also display the file upload field and it's associated label using the correct variables from the Flask-WTF form you created.

    Note: You also need to ensure you print out the Flask-WTF **`csrf_token`** hidden field or else your validation may fail.

6. Test that your file uploads work by starting the development server for your Flask app, go to the **`/upload`** route and log in. Upload a file and

then check your uploads directory to see if it is there.

Note: The username and password to use to login for this exercise is in your **`config.py`** file as configuration options. This is **not** a best practice and is being done just because we don't have a database or a users table in this lab. In your next lab you will create a proper login system and store this information in a database.

7. Commit your code to your repository.

Exercise 2: Listing your uploaded files

Here's an example python script for iterating over some files in a specific directory. **Note:** You will need to make modifications to get this to work in your lab, as it will not work as you see it here.

```
import os
rootdir = os.getcwd()
print rootdir
for subdir, dirs, files in os.walk(rootdir + '/some/
folder'):
    for file in files:
        print os.path.join(subdir, file)
```

1. Use the code from the example above to help you create a helper function called **`get_uploaded_images()`** in your app which iterates over the contents of the **`uploads`** folder and stores the filenames in a python list which the function will return. **Hint:** Yes, you will need to make some modifications to the code above, it won't work as is.

2. Create a route called "**/uploads/<filename>**" and a view function called **get_image()**, it should take an argument called "**filename**". You will use this view function to *return* a specific image from your upload folder using the **send_from_directory()** function. e.g. **send_from_directory(app.config['UPLOAD_FOLDER'], filename)**

3. Next, create another new route called "**/files**" and its respective **files()** view function. You should also render a template named "**files.html**" which lists the image files uploaded in the **uploads** folder as an HTML list (ie. using an unordered or ordered list) of images.

   Note: You should use the function you created in Step 1 to get the image and pass the list of image filenames to your template. You will then need to use the **url_for()** function to help print the proper url in your **<img />** tags that uses the route you created in Step 2.
4. Open **header.html** and add a new menu item to your navigation called "**View Images**" and link it to the "**files**" route that you created in Step 2.
5. Commit your code to your Github repository.
6. Ensure that the **/files** route is only accessible to users who are logged in. **Hint:** Take a look at the **upload** view function to see how we did this.
7. Commit your code.
8. Now see if you can modify your **files.html** template file to display and style your list of images to look like the screenshot (see Figure 1).

Figure 1: Images should be displayed in a Grid

Hint: You may want to try using CSS Grid or Flexbox to help with this layout. 😉 **DO NOT use an HTML table.**

Note: If you didn't do so before, add, commit and push your code to your Github repository

```
git add .
git commit -m 'your commit message'
git push origin master
```

Submission

Submit your code via the "Lab 4 Submission" link on OurVLE. You should submit the following link:

1. Your Github repository URL for your Flask app e.g. https://github.com/{yourusername}/info3180-lab4

Grading

1. Added config option for UPLOAD_FOLDER in config.py (1 mark)
2. Create UploadForm class and file upload field defined with validation rules. (2 marks)
3. Instantiate UploadForm class in the appropriate place and pass to render_template function. (1 mark)
4. Use form.validate_on_submit() or form.validate() to check form validation (1 mark)
5. Use app.config['UPLOAD_FOLDER'] in code to save file, also ensure you use secure_filename() (2 marks)
6. The form should have the correct 'method' and 'enctype' attribute set. Also the form should have a file field and CSRF Token. (3 marks)
7. Your Image File should successfully upload (1 mark)
8. The get_uploaded_images() function should be defined and it should return a Python list of image filenames. (2 marks)
9. A Files route, associated view function and template file should be created (2 marks)

10. A "/upload/<filename>" route with associated get_image() view function that uses the send_from_directory() function to return a specific image from the uploads folder. (2 mark)
11. The Files route should only be accessible when logged in. (1 mark)
12. The Files route should look like screenshot (see Figure 1) and use CSS Grid or Flexbox to define the grid layout. (2 marks)

    Note: You will lose marks if you use an HTML table. **Do NOT use an HTML table to layout the pictures.**