Flask Python (4) - Forms

Users

If we are going to ask students to login, we to let them sign up and then in some way store their data. The next two tasks are connected with displaying forms to let users sign up and login.

forms.py

The following code provides us with two classes. One is a form to login to a website (*LoginForm*) and a form to let people sign up (*SignUpForm*). They have attributes for each form. Create the file **forms.py** in the site directory.

```
#### form.py ###

#### Creates two forms signing up and logging in ####


## Import wtforms and FlaskForm libraries to help us

from flask_wtf import FlaskForm

from wtforms import StringField, SubmitField, PasswordField

from wtforms.validators import DataRequired, Email, EqualTo


## Class for login form

class LoginForm(FlaskForm):

    email = StringField('email')

    password = PasswordField ('Password')

    submit = SubmitField('Log In')


# Class for our sign up for to allow students to register.

class SignUpForm(FlaskForm):

    fname = StringField('First name')

    surname = StringField('Surname')

    email = StringField('email')

    password = PasswordField ('Password')

    pass_confirm = PasswordField ('Confirm Password')

    code = StringField('Sign up code')

    submit = SubmitField('Register!')
```

Rendering the form

We now need to add some code to our *flask_app.py* file so that we can use the form class we made and provide a template for users to sign up to. Things to note here are the use of the POIT and GET methods. This allows data to be moved from one page to another. We will explore the difference later on but for now don't worry!

Step 1: At the top of the file you will need to alter your first line so you can redirect your code to a different web page:

```
from flask import Flask, render_template, url_for, redirect
```

Step 2; You will also need to add a new line

```
from forms import SignUpForm
```

This imports our `SignUpForm` class form *forms.py*.

Step 3: After you have created the app object you will need to add a secret key to your application so that the forms can pass encrypted data from the web page to application. After you created the flask app object you will need a new line of code:

```
app = Flask(__name__)

app.config['SECRET_KEY'] = 'mykey'
```

You can use any text as your key.

Step 4: Then create another route as follows:

```
@app.route('/signup', methods=["GET","POST"])
def signup():
    form = SignUpForm()
    if form.validate_on_submit():
        return redirect(url_for('index'))
    return render_template('signup.html',form=form)
```

The above code creates our signup route and uses the GET and POST methods (don't worry for the moment about these but they are needed to pass data). The function signup() creates an object called form (based on our class) and if it is a valid form it will return us to the index page and if not it will render the signup template. The render_template function also has data it is passing namely our object form (so we have access to the form data!)

signup.html

The next part of this is to produce the template *signup.html.* We do this in the templates folder. Here is the code:

```
{% extends "index.html" %}

{% block content %}

      <h2>SIGN UP</h2>

        <form method="POST">
```

```
            {{form.hidden_tag()}}
            {{form.fname.label}} {{form.fname}}<br>
            {{form.surname.label}}{{form.surname}}<br>
             {form.email.label}}{{form.email}}<br>
             {{form.password.label}}{{form.password}}<br>
            {{form.submit}}
        </form>
    {% endblock %}
```

Check it works

Step 1: Make sure the **Sign Up** link on your navigation bar works and takes you to the form.

Step 2: Fill in a form and check that the **Register!** button takes you back to the index page.

Step 3: The form now needs to be in the middle of the page and properly formatted using bootstrap.