Routes.py

Lines through 1 to 5 import all necessary files and libraries and make sure you install all modules using apt-get command if you are on a linux machine and pip command if you are on a windows machine. In line 7 we pass our app to a Bootstrap() function to make some bootstrap templates available to serve as a blueprint that includes all bootstrap resources with predefined blocks where users can place their content.

routes.py forms.py dashboard.html index.html

```
1  from appfolder import appFlask
2  from flask import Flask, render_template, redirect, url_for, request
3  from flask_bootstrap import Bootstrap
4  from appfolder.forms import LoginForm, Settings
5  import RPi.GPIO as GPIO
6
7  Bootstrap(appFlask)
8
```

Following that we create three routes for our app: index page, dashboard, and logout. Nowadays, web applications have user-friendly URLs for users' convenience. We use the route() decorator to bind a function to a corresponding URL. As the user types in the IP address for the web interface, he gets to the index page where he is prompted to log in to the system to start it. That is done by our first route('/', methods=['GET', 'POST']), which accepts HTTP methods GET and POST. We bind a function called index() to this decorator. We instantiate a LoginForm() object called form to be able to get the user's credentials when he logs in. validate_on_submit() function returns True if the form has been submitted either through GET or POST. In the if statement following that we check the values for username and password that were predefined for the sake of simplicity in this template. Finally, if the user enters the right credentials, he is taken to a dashboard.

```
9  @appFlask.route('/', methods=['GET', 'POST'])
10 def index():
11     form = LoginForm()
12     error = None
13     if form.validate_on_submit():
14         if form.username.data != 'admin' or form.password.data != '12345678':
15             error = 'Invalid Credentials. Please try again.'
16         else:
17             return redirect(url_for('dashboard'))
18     return render_template("index.html", form=form, error=error)
```

Our second decorator route('/dashboard', methods=['GET', 'POST']) is the route for the dashboard. We bind a function called dashboard() to this decorator. This part of our code actually deals with the hardware. We take advantage of the fact that a Raspberry Pi allows interfacing sensors and actuators through the general purpose I/O pins. Because we imported RPi.GPIO as GPIO in line 5, we can now use this object to interact with the hardware. In our function dashboard(), the first thing we do is to set up the mode for GPIO pins. We use BCM which is a Broadcom chip-specific numbering scheme. After that, we define two channels (pins) for both of our sensors assuming that they would be deployed in two different rooms. We set our signals to be 0 at the beginning. We set our channels to act as input pins in lines 28 and 29. We then instantiate an object from Settings class to get the settings of the device through the form. In the dashboard, whenever the user turns the power on, our sensors start waiting for the input. Then sensors start listening in the infinite while loop and as soon as one of the sensors detect sound or water, they break out of the infinite while loop and send the signal to the dashboard specifying the exact location of where a burglar triggered the sensor.

routes.py forms.py dashboard.html index.html

```
19
20
21  @appFlask.route('/dashboard', methods=['GET', 'POST'])
22  def dashboard():
23      GPIO.setmode(GPIO.BCM)
24      channel = 21
25      channel2 = 20
26      signal = 0
27      signal2 = 0
28      GPIO.setup(channel, GPIO.IN)
29      GPIO.setup(channel2, GPIO.IN)
30      form = Settings()
31      message = False
32      if form.validate_on_submit():
33          GPIO.add_event_detect(channel, GPIO.RISING)
34          GPIO.add_event_detect(channel2, GPIO.RISING)
35          if form.power.data != True:
36              message = False
37          else:
38              message = True
39      while message:
40          if GPIO.event_detected(channel):
41              signal = 1
42              message = False
43          elif GPIO.event_detected(channel2):
44              signal2 = 1
45              message = False
46      GPIO.remove_event_detect(channel)
47      GPIO.remove_event_detect(channel2)
48      return render_template("dashboard.html", form=form, signal=signal, signal2=signal2)
```

The last function logout() is bound to the route('/logout') that redirects the user to the index page.

```
51  @appFlask.route('/logout')
52  def logout():
53      return redirect(url_for('index'))
54
```

Reference:

http://flask.pocoo.org/docs/0.12/quickstart/

https://explore-flask.readthedocs.io/en/latest/forms.html

https://pythonhosted.org/Flask-Bootstrap/basic-usage.html