Lab 7: Testing and Deployment

February 2023

Deployment

Software deployment refers to the comprehensive set of procedures, tasks, and actions that are necessary to ensure that a software system or upgrade is accessible and ready for use by its targeted audience. This involves a series of activities such as planning, testing, configuration, installation, and maintenance, aimed at guaranteeing that the software is properly deployed and functional. The objective of software deployment is to make certain that end-users can take advantage of the latest features and bug fixes in the software, while ensuring that the process is seamless and does not disrupt the user experience. You may use many platforms to deploy your project. We will use pythonanywhere.

- Create an account in pythonanywhere.com, select beginner (free account!)
- Go to option "Web" and select "Add a new web app", select next and then "Flask", python 3.7, then press "next"
- Go to option "Files" and upload all files to "mysite" directory

Testing *(Not mandatory)*

Writing unit tests for a Flask application is an important step in ensuring that the application is working correctly and that any changes made to the codebase do not break existing functionality. In this tutorial, we will be using the unittest library that comes with Python, and the Flask test client to test our views, and we will be deploying the application on pythonanywhere, a platform that makes it easy to deploy and run web applications.

First, let's start by creating a new Flask project, create a new directory for your project and within that directory, create a file called app.py and add the following code:

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"
```

This will create a basic Flask app that will display "Hello World!" when you run it.

Next, let's create a new file called test.py in the same directory as your app.py file. This file will contain the unit tests for your application. In this file, we will import the necessary modules, create a test client, and write test cases for our views.

```
import unittest
from app import app

class FlaskTestCase(unittest.TestCase):

    def setUp(self):
        self.app = app.test_client()
        self.app.testing = True

    def test_hello(self):
        response = self.app.get('/')
        self.assertEqual(response.status_code, 200)
        self.assertEqual(response.data, b'Hello
        World!')

        if name == 'main':
        unittest.main()
```

In this code, we have created a test case for the 'hello' view that we created earlier. The 'setUp' method is called before each test case, and it creates a test client and sets the 'testing' flag to 'True'. The 'test_hello' method sends a GET request to the '/' endpoint and asserts that the response has a status code of 200 and that the response data is "Hello World!".

To run these tests, you can simply run the 'test.py' file using the following command:

```
python test.py
```