Coursework 1 Report

Jack Sweeney
40209035@napier.ac.uk
Edinburgh Napier University - Advanced Web Technologies (SET09103)

Keywords – Python, Flask, development, web, project, Bootstrap, app, bands, report

1 Title

Favorite Bands Python Project

2 Introduction

For this assignment i was asked to design and develop and catalogue of entries in the form of a **Python Flask app**, this app had to make use of; *url hierarchy and navigation, requests, responses and redirects, templates and static files.*

3 Plan

To meet these requirements i decided to develop a short catalogue detailing my four favorite bands, the app would include *a gallery of album art, individual pages with artists' details and discography and a navbar at the top of each page for easy navigation.*

I also decided to add redirects to each page meaning that the user could abbreviate each artist in the address box and be navigated to the artist page they require.

4 Development

To develop this app i used **Levinux** along with **Putty** and the coding for the app was completed using **Vim** I also utilised **Bootstrap** to add some style to the html files. In total the app consists of 6 HTML template pages, 2 Python Flask files and a config file.

The 5 HTML template files are; *index.html, LCD.html, AF.html, RH.html, AM.html and 404.html.* These files are what the user interacts with, when the user makes a request the Python app responds with one or more of these pages. 404.html is where the user is sent if the page the have requested does not exist. The Python Flask files are,*index.py and testing.py, testing.py* is the testing harness I used to check that responses to error codes were as expected, *index.py* is the main page of the whole app, this processes the users demands and responds by directing them to the corresponding page of the app, i.e on startup the user will be taken to *index.html* or the *'/' route*

full details of the code used for this app is contained at the end of this report

5 Testing

After development i tested the app to ensure it was functioning to the best of it's ability. I tested **200** and **404** error codes on the app by creating a test harness and comparing responses to calls sent to the '/' route. When tested on a 200 code the app functions as normal, sending the user to the index page. When tested on a 404 the app fails and the user is show *404.html* which is a blank page with "Sorry, i could not find the page you are looking for".

Figures 1,2,3,4 and 5 show this testing in more detail

6 Enhancements

If i were building a similar project in the future there would be a few things i would do differently in regards to the design and development of the app, for example I would allocate more time to the development stage of the project as towards the end of this app's construction I felt myself rushing to get certain features completed, added to this more time would allow me to add more features to the app, for example i had the idea of adding more artists and incorporating a search bar into the homepage to allow people to search for specific pages or albums.

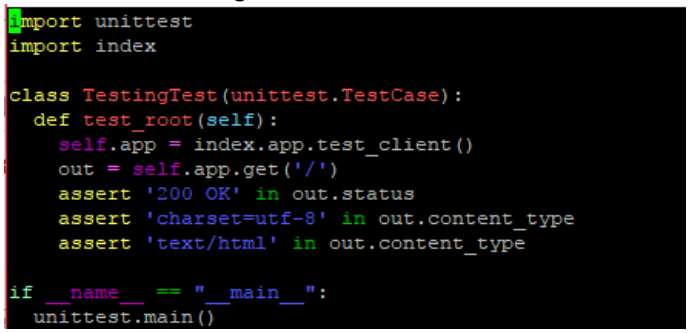
7 Evaluations

In conclusion i feel that this app meets the technical requirements of the assignment while also being relatively stylish, i can admit, however, that it is not without it's faults, i could have been more thorough with testing and i could have written more redirects but i feel that this could be a suitable prototype to outline features and designs for a future project.

I feel I have picked up a lot of valuable skills in regards to coding by undertaking this project, I have learned the many new things about Python Flask including; *Requests, Responses, Redirects and how to implement HTML templates within Python apps*. I have also improved my knowledge and skills in regards to working on the command line. Overall, though I feel my work ethic could have been vastly improved, I did not anticipate the work that would be required in building this app, I should have set aside more time to carry out this project, that being said I feel that the positives far outweigh the negatives. I think that the hardest aspect of building this application was the learning curve that was involved, the way it had to be coded was unlike any project I have been involved in previously but i'm very happy with the way it turned out and the skills i have learned.

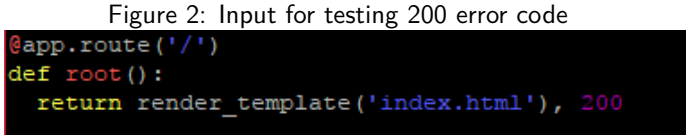
8 Figures

Figure 1: Test Harness



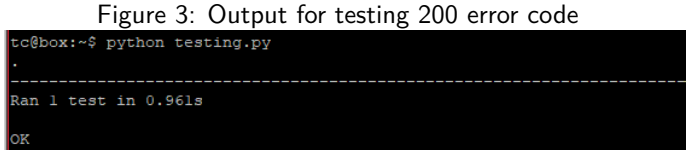
```
import unittest
import index

class TestingTest(unittest.TestCase):
  def test_root(self):
    self.app = index.app.test_client()
    out = self.app.get('/')
    assert '200 OK' in out.status
    assert 'charset=utf-8' in out.content_type
    assert 'text/html' in out.content_type

if __name__ == "__main__":
  unittest.main()
```

Figure 2: Input for testing 200 error code



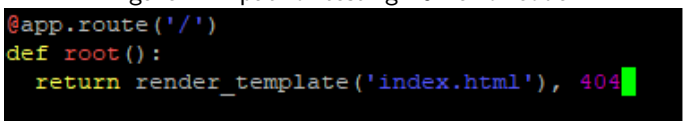
```
@app.route('/')
def root():
  return render_template('index.html'), 200
```

Figure 3: Output for testing 200 error code



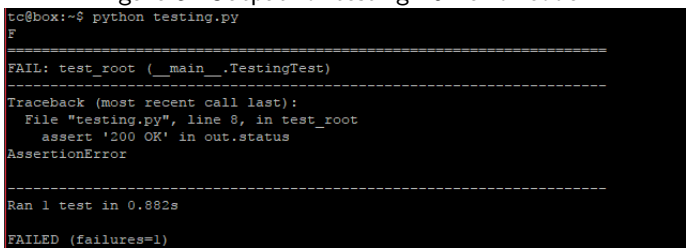
```
tc@box:~$ python testing.py
.
----------------------------------------------------------------------
Ran 1 test in 0.961s

OK
```

Figure 4: Input for testing 404 error code



```
@app.route('/')
def root():
  return render_template('index.html'), 404
```

Figure 5: Output for testing 404 error code



```
tc@box:~$ python testing.py
F
======================================================================
FAIL: test_root (__main__.TestingTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "testing.py", line 8, in test_root
    assert '200 OK' in out.status
AssertionError

----------------------------------------------------------------------
Ran 1 test in 0.882s

FAILED (failures=1)
```

9 Examples of Code Used

Listing 1: index.py

```
import ConfigParser
from flask import Flask, render_template, redirect, url_for
app = Flask(__name__)

@app.route('/')
def root():
  return render_template('index.html'), 200

@app.route('/LCDSoundsystem')
def lcd():
  return render_template('LCD.html'), 200

@app.route("/LCD")
def relcd():
  return redirect(url_for('lcd')), 301

@app.route('/ArcadeFire')
def af():
  return render_template('AF.html'), 200

@app.route('/AF')
def reaf():
  return redirect(url_for('af')), 301

@app.route('/Radiohead')
def rh():
  return render_template('RH.html'), 200

@app.route('/RH')
def rerh():
  return redirect(url_for('rh')), 301

@app.route('/ArcticMonkeys')
def am():
  return render_template('AM.html'), 200

@app.route('/AM')
def ream():
  return redirect(url_for('am')), 301

@app.errorhandler(404)
def page_not_found(error):
  return render_template('404.html'), 404

@app.route('/config/')
def config():
  str = []
  str.append('Debug:'+app.config['DEBUG'])
  str.append('port:'+app.config['port'])
  str.append('url:'+app.config['url'])
  str.append('ip_address:'+app.config['ip_address'])
  return '/t'.join(str)

def init(app):
    config = ConfigParser.ConfigParser()
    try:
        config_location = "etc/defaults.cfg"
        config.read(config_location)

        app.config['DEBUG'] = config.get("config", "debug")
        app.config['ip_address'] = config.get("config", "ip_address"↩
       )
        app.config['port'] = config.get("config", "port")
        app.config['url'] = config.get("config", "url")
    except:
        print "Could not read configs from: ", config_location

if __name__ == '__main__':
    init(app)
    app.run(
        host=app.config['ip_address'],
        port=int(app.config['port']))
```