Team 6 Testing Framework

Experience:

The experience for this deliverable was much nicer than the previous, since, misunderstanding what was needed for the previous deliverable, writing the script for executing test cases is something that we had already taken care of. Needless to say this was unpleasant when deliverable 2 was rolling around, but has since made the project outlook very good, with a testing framework which will easily accommodate additional test cases as the project continues

Architecture:

The script clears the temp folder, creates, and subsequently iterates through a list of all text files in the testCases folder. These test cases are written in a specific format to facilitate parsing by the script as follows:

Line1: Description of the test
Line2: Path to the .py file the tested function resides in
Line3: Function being tested
Line4: Arguments
Line5: Expected output

After parsing, the script makes the method call along with the given argument and compares the results to the expected output given in the test case for each test in the testCases folder. The output of all of these tests is written to an html file.

Script:

```
#!/usr/bin/python

import subprocess
import os.path
import sys
from importlib import import_module

def getTestCases():
    os.chdir("../testCases/")
    allFiles = os.listdir(".")
    return [TestCase(fileName) for fileName in allFiles if ".txt" in
fileName]

def executeTest(testCase):
    importStatement = convertPathToImport(testCase.modulePath)
    sys.path.append(testCase.modulePath)
    module = __import__(importStatement, fromlist=[testCase.methodName])
    methodToTest = getattr(module, testCase.methodName.replace("()",""))
    result = methodToTest(testCase.inputValue)
    testCase.actualResult = result

    #actual test
    if result == testCase.expectedResult:
        testCase.testPassed = True
    return testCase
```

```
def convertPathToImport(path):
    path = path.replace("/", ".")
    path = path.replace(".py","")
    return path.split(".")[-1]

class TestCase:
    def __init__(self, fileName):
        file = open(fileName, 'r')
        fileContents = file.read().split("\n")
        self.id = fileContents[0]
        self.description = fileContents[1]
        self.modulePath = fileContents[2].strip()
        self.methodName = fileContents[3].strip()
        self.inputValue = fileContents[4]
        self.expectedResult = fileContents[5]

        self.actualResult = ""
        self.testPassed = False

        file.close()


def clearTempFolder():
    #go back a directory, remove everyting from temp,
    #if nothing is in temp, the warning is suppressed by sending it to null
    subprocess.call("rm ../temp/* 2>/dev/null", shell=True)

def generateHtml(testResults):
    html = "<html>"
    html += "<body>"
    html += "<h1>Test Results</h1>"
    html += testResults
    html += "</body>"
    return html + "</html>"

def writeHtmlFile(html):
    os.chdir("..")
    os.chdir("./reports/")
    filename = "test_results.html"
    output = open(filename,'w')
    output.write(html)
    subprocess.call("xdg-open " + filename, shell=True)

def main():
    clearTempFolder()
    testCases = getTestCases()
    outputString = ""
    for testCase in testCases:
        executeTest(testCase)
        if testCase.testPassed:
            outputString += '<p style = "color:green">'
            outputString += "Test case "+testCase.id+" passed!"
            outputString += '</p>'
```

```
        else:
            outputString += '<p style = "color:red">'
            outputString += "Test case "+testCase.id+" FAILED!"
            outputString += '</p>'
        outputString += "<ul>"
        outputString += "<li>\tInput: " + testCase.inputValue + "</li>"
        outputString += "<li>\tExpected Result: " + testCase.expectedResult
+ "</li>"
        outputString += "<li>\tActual Output: " +
str(testCase.actualResult) + "</li>"
        outputString += "</ul>"
    htmlBody = generateHtml(outputString)
    writeHtmlFile(htmlBody)


main()
```

Test Cases:

1
this will test the upper method, converting all lower case letters to upper case
jython/lib-python/2.7/string.py
upper()
caT
CAT

2
this will test the strip method, taking a string and removing any trailing or leading whitespace
jython/lib-python/2.7/string.py
strip()
    this is a test for trim
this is a test for trim

3
this will test the lower method
jython/lib-python/2.7/string.py
lower()
CAT
cat

4
this will test the swapcase method changing lower case letters to upper case and upper case letters to lower case
jython/lib-python/2.7/string.py
swapcase()
teSTING fOr Cases
TEsting FoR cASES

5
this method will test the capwords method, which takes in a string and returns that string with all of the words capitalized
jython/lib-python/2.7/string.py
capwords()
this iS a tEst.
This Is A Test.