Queen’s University

Assignment:
Quinterac part 1

Course:
CMPE 327

Prepared for:
Lecturer: Ding, Steven

Prepared by:
Adams Liu: 10187602
Eric Chau: 10141119
Ian Chang: 10190262

Due Date:
October 6, 2019

Testing structure

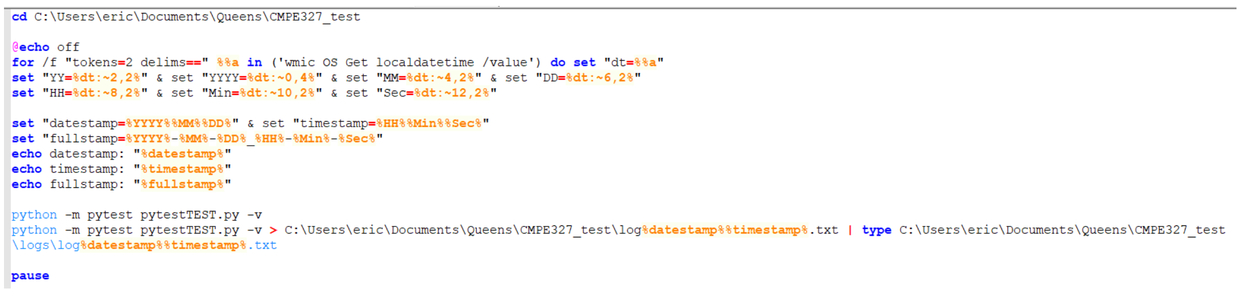
The available features in the Quinterac system are as listed: login, logout, deposit, withdraw, transfer, createacct (create account), and deleteacct (delete account). The test cases for the system will be tested in the same order as the list above, with login/logout, deposit, withdraw, transfer, createacct, and lastly with deleteacct. A login transaction is required to make any other transaction, and a logout transaction is needed to complete all previous transactions made in a session. Therefore, before any other transactions can be tested, the login and logout transaction must be tested first. Since the rest of the transactions have no functional dependencies with one another, the ordering of the other transaction does not matter as much. Following the login and logout test cases are the deposit/withdraw/transfer test cases, and then lastly ending with test cases createacct/deleteacct test cases.

*Note the createacct/deletacct transactions can only be tested under the “agent” user.

Test environment setup

For our test cases, we intend the user to be testing on the Windows OS thus we will be using batch files. The batch file requires the directory in which the python test file is located, as well as the directory the user wants the log files to be stored in. This must be manually modified in the batch file for each system that will be used for testing (since each system has different directory names). In addition, the batch file assumes that the user has pytest installed on the system (not just in a virtual environment, like pycharm’s terminal uses). If it does not, this can easily be done through a bash terminal (e.g. Windows powershell) and using the command “pip install -U pytest”. The installation can be verified using the command “pytest –version”.

After all of which is set up, tester can simply execute the batch file and the batch script will be run. Since Windows batch files uses the command prompt as its default UI, the script first changes into the directory which the python file is located (as specified above). Environment variables are then set to utilize the user’s machine’s time and date. The script then runs the python file and pipes the test summary into a log file using the time and date variables initially set up. This way after executing numerous tests, the log files would (by default) be sorted in order of date and time tested.

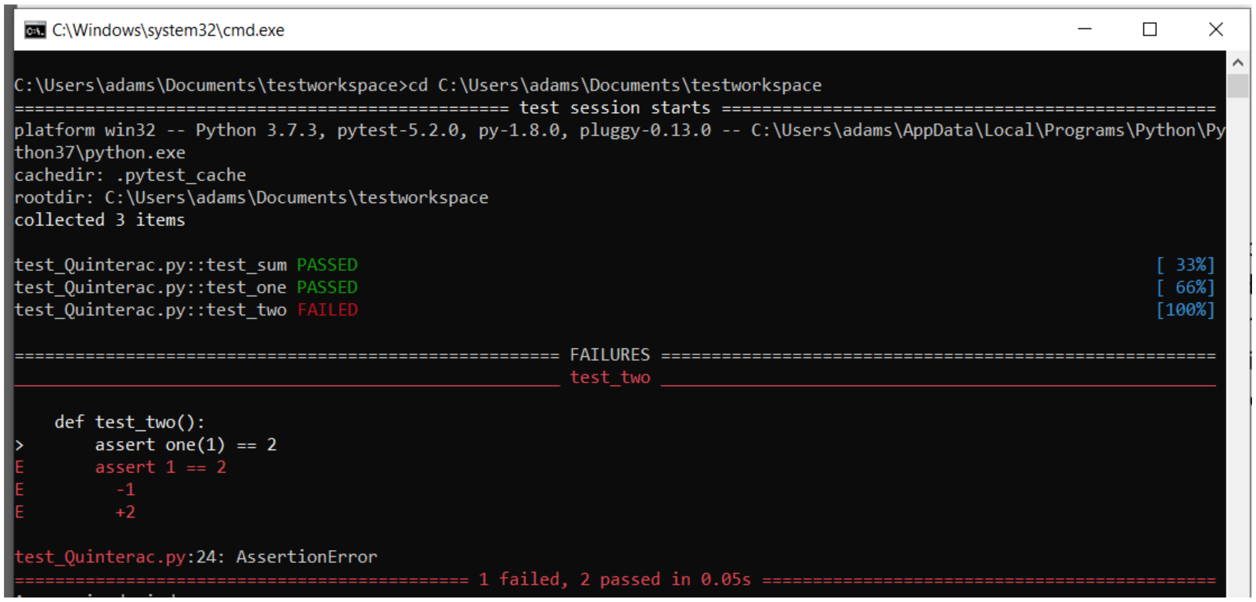


```
cd C:\Users\eric\Documents\Queens\CMPE327_test

@echo off
for /f "tokens=2 delims==" %%a in ('wmic OS Get localdatetime /value') do set "dt=%%a"
set "YY=%dt:~2,2%" & set "YYYY=%dt:~0,4%" & set "MM=%dt:~4,2%" & set "DD=%dt:~6,2%"
set "HH=%dt:~8,2%" & set "Min=%dt:~10,2%" & set "Sec=%dt:~12,2%"

set "datestamp=%YYYY%%MM%%DD%" & set "timestamp=%HH%%Min%%Sec%"
set "fullstamp=%YYYY%-%MM%-%DD%_%HH%-%Min%-%Sec%"
echo datestamp: "%datestamp%"
echo timestamp: "%timestamp%"
echo fullstamp: "%fullstamp%"

python -m pytest pytestTEST.py -v
python -m pytest pytestTEST.py -v > C:\Users\eric\Documents\Queens\CMPE327_test\log%datestamp%%timestamp%.txt | type C:\Users\eric\Documents\Queens\CMPE327_test
\logs\log%datestamp%%timestamp%.txt

pause
```

Output validation

Functions are tested using a pytest script which runs all test cases listed in the test case table. Test cases are automatically validated based on the returns of all test functions. Test functions include the transactions within the Quinterac system (eg. login, logout, atm, agent, deposit, etc) returning either a 1 or 0. If the test cases succeed that means the test function was able to go through all Quintearc transactions. The results will return a 0 indicating there are no errors and the test case passed, if the test case returns 1 then the test has case failed. All test cases are preprogrammed in the pytest script itself and will not require external input files to run the test cases.

The output results will be displayed on the windows command console. This result will display which test cases have passed, which test cases have failed and how they failed, as well as the time it took to run the pytest script.

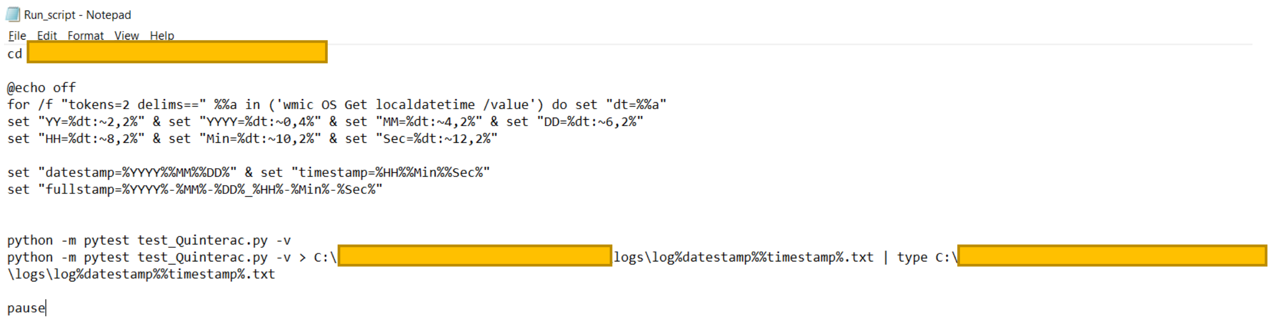


C:\Windows\system32\cmd.exe

```
C:\Users\adams\Documents\testworkspace>cd C:\Users\adams\Documents\testworkspace
================================================= test session starts =================================================
platform win32 -- Python 3.7.3, pytest-5.2.0, py-1.8.0, pluggy-0.13.0 -- C:\Users\adams\AppData\Local\Programs\Python\Py
thon37\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\adams\Documents\testworkspace
collected 3 items

test_Quinterac.py::test_sum PASSED                                                                               [ 33%]
test_Quinterac.py::test_one PASSED                                                                               [ 66%]
test_Quinterac.py::test_two FAILED                                                                               [100%]

====================================================== FAILURES =======================================================
______________________________________________________ test_two _______________________________________________________

    def test_two():
>       assert one(1) == 2
E       assert 1 == 2
E         -1
E         +2

test_Quinterac.py:24: AssertionError
============================================= 1 failed, 2 passed in 0.05s =============================================
```

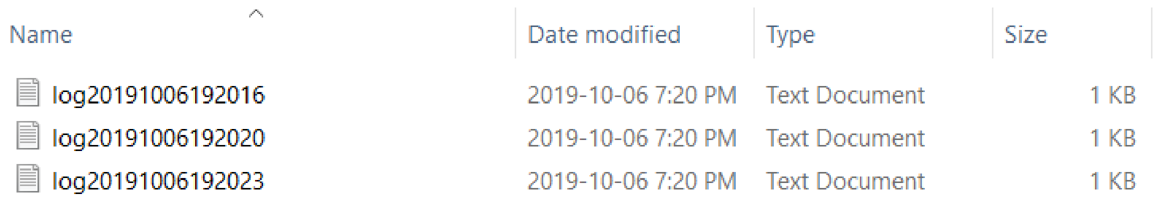
Result storage

In the batch file the script is set up so that the output of the python test will record a log file of the results. The batch file script needs to be edited so that the path points to the directory to where the log files will be stored. (This is highlighted in in yellow)



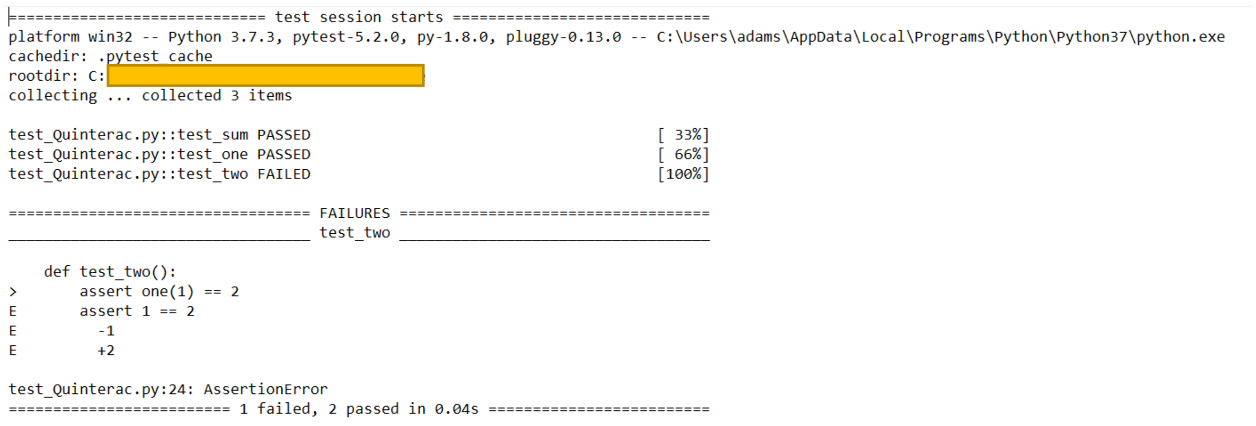
```
cd [redacted]

@echo off
for /f "tokens=2 delims==" %%a in ('wmic OS Get localdatetime /value') do set "dt=%%a"
set "YY=%dt:~2,2%" & set "YYYY=%dt:~0,4%" & set "MM=%dt:~4,2%" & set "DD=%dt:~6,2%"
set "HH=%dt:~8,2%" & set "Min=%dt:~10,2%" & set "Sec=%dt:~12,2%"

set "datestamp=%YYYY%%MM%%DD%" & set "timestamp=%HH%%Min%%Sec%"
set "fullstamp=%YYYY%-%MM%-%DD%_%HH%-%Min%-%Sec%"


python -m pytest test_Quinterac.py -v
python -m pytest test_Quinterac.py -v > C:\[redacted]logs\log%datestamp%%timestamp%.txt | type C:\[redacted]
\logs\log%datestamp%%timestamp%.txt

pause
```

The log file will be a copy of the terminal output of the python script. This will be in the “logs” folder and have the filename “log[YYYY][MM][DD][hh][mm][ss].txt” . The log files are named with the time and date the tests were completed, this organization and naming convention will help when comparing to past/future runs of the test results.



Name	Date modified	Type	Size
log20191006192016	2019-10-06 7:20 PM	Text Document	1 KB
log20191006192020	2019-10-06 7:20 PM	Text Document	1 KB
log20191006192023	2019-10-06 7:20 PM	Text Document	1 KB

A sample screenshot of how the logs look is shown below:



```
============================ test session starts ============================
platform win32 -- Python 3.7.3, pytest-5.2.0, py-1.8.0, pluggy-0.13.0 -- C:\Users\adams\AppData\Local\Programs\Python\Python37\python.exe
cachedir: .pytest_cache
rootdir: C:[redacted]
collecting ... collected 3 items

test_Quinterac.py::test_sum PASSED                                      [ 33%]
test_Quinterac.py::test_one PASSED                                      [ 66%]
test_Quinterac.py::test_two FAILED                                      [100%]

================================== FAILURES ==================================
__________________________________ test_two __________________________________

    def test_two():
>       assert one(1) == 2
E       assert 1 == 2
E         -1
E         +2

test_Quinterac.py:24: AssertionError
========================= 1 failed, 2 passed in 0.04s =========================
```

Additional notes

There are no specific testcases written for the transaction summary file and the valid account list file, since the valid account list file and the transaction summary file are produced by the back end of the system and its validity does not depend on the user input.

The transaction summary file and the valid account list file will be manually verified by the tester through comparison between the expected result and the actual results by hand. The tester would read through the actual results and verify whether it meets the front-end requirements demanded by the system. For example, when a new transaction summary file is outputted by the system, the tester would manually examine the file to check whether the formatting constraints (such as max character count per line or spacing limits) are met.

Unless the logout transaction is completed, no transaction summary file will be outputted from the system. If a test case ends without calling the logout transaction, no output file will be generated. After the logout transaction is called, it will produce a list of new transactions that were made in the system. However, if the logout transaction was called without calling transactions that involve an account number, the transaction summary file will just output an empty file with the end of session code (EOS).