

TEST RUN PLAN

QBASIC



|  |  |
| --- | --- |
| Name | Student Number |
| Muhammad Usman Majeed | 10086980 |
| Jessica Nahulan | 10029341 |
| Johan Cornelissen | 10098176 |

October 3, 2017

Contents

[Test Run Organization: 2](#_Toc494569797)

[Directories overview: 2](#_Toc494569798)

[Test Runs: 3](#_Toc494569799)

[Test automation: 3](#_Toc494569800)

[Output validation: 3](#_Toc494569801)

[Results reporting and organization: 3](#_Toc494569802)

# Test Run Organization:

## Directories overview:

TRANSFER

VALID ACCOUNT LIST FILE

INPUT TEST

Output

Input

QBASIC

Testing

WITHDRAW

DEPOSIT

LOGOUT

CREATEACCT

TRANSACTION SUMMARY FILE

LOGIN

DELETEACCT

*The above diagram shows the high-level description of the structure for our directory organization.*

Tests will be organized into folder per feature directories with each feature having its own respective input and output subdirectories. The input folders will contain all input going into the system such as the valid accounts list file and the user input (transaction codes and user input parameters). The output folders will contain all output both generated by the front end in the form of the Transaction Summary File and output messages presented to the user including prompts, try again, and error messages.

# Test Runs:

## Test automation:

We will be using windows batch files for our testing, as this will be the simplest possible solution for our project. It is also an approach which all team members have experience with and as we are following extreme programming practices, it makes the most sense to take this path. The batch scripts will run the test cases using all the inputs from the respective features directories and outputting into the respective output subdirectories.

The higher-level view of our test automation plan:

Python Environment

Concurrent Interface Engine

Object repository

Test Template

Business logic Comment

Business Logic

Product Library

Framework Library

Test Runner

Parser

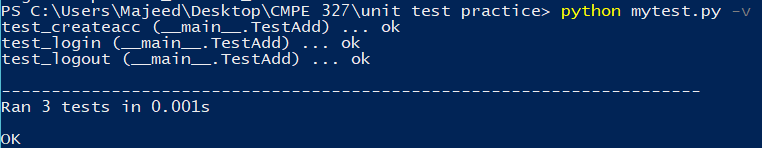
Python

Element Variable

*The above diagram shows the high-level description of our test automation.*

## Output validation:

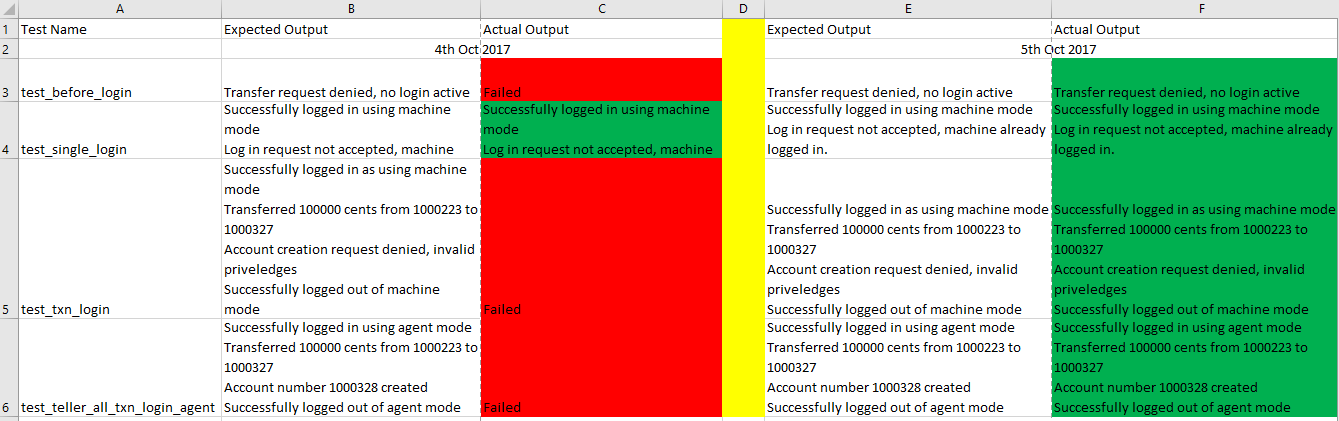
For output validation, we will be using the python unit test framework to complete all white-box testing, which will help produce a comprehensive list of which functions were failed or passed and how long it took for each test to run. For black-box testing, the team will be using a separate python script to verify the standard output and transaction files created match the expected output.



*The above image shows the output of test validation.*

## Results reporting and organization:

The results from the test automation and python test validations will be stored in an excel spreadsheet. This will help us keep track of all the test fails and passes as well as keep track of which day/time the test was run and by whom. This information can be compared with future results to see if there is any regression during the development. The excel sheets will also help anyone to easily get an overview of what the current state of the system’s operation is and which functions/constraints/features are able to run successfully and which are still in progress.



*The above image is an example of a spreadsheet for test reporting and future runs comparison*