Queen's University

Assignment:
Quinterac part 5

Course:
CMPE 327

Prepared for:
Lecturer: Ding, Steven

Group Number:

23

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

Due Date:
November 25th, 2019

Note

Both White-Box testing methods (decision coverage and statement coverage) is done using code injection. A separate backend file was cloned to perform code injection, the filename is "test_backend.py". We decided to do source level implementation testing using the coverage array method. We have a global array called "wbCounter[]" and whenever a test case occurs we increment the value in the array by 1.

Withdraw Method

1. Decision coverage is used for the "withdraw" method (starts at line 24)
2. For decision coverage we made a test case for each decision in the program. The first case enters the error statement when the amount exceeds the total balance, and the second case enters the success statement when the amount doesn't exceed the total balance. The "wbCounter1" array value is incremented by 1 for their respected indices.

```
# transaction withdraw
def withdraw(acc1, amount):
    if(Master_Dict[acc1]["balance"] < amount):
        wbCounter1[0] += 1
        print("FATAL ERROR: withdraw insufficient funds")
    else:
        wbCounter1[1] += 1
        Master_Dict[acc1]["balance"] = Master_Dict[acc1]["balance"] - amount
    return 0
```

3. The test inputs that were used to enter each if statements are the following:

Decision	amount input	test	amount
TRUE	600000	T1	600000
FALSE	400000	T2	400000

*Note only "amount" is considered as the input because the "acc1" parameter is assumed to be correct as the account number is verified in the frontend. The default balance for every account is assumed to be 500000.

To run these tests the following lines were included in the master transaction summary file.

WDR 1234567 600000 0000000 ***
WDR 1234567 500000 0000000 ***

4. Finally, the array was reviewed by a print statement to see if the test cases successfully ran. The withdraw test case showed both values of the array to be “1” meaning both test cases successfully executed. There were no errors found.
   Counter Array 1 is the white-box record array for the withdraw method.

```
FATAL ERROR: withdraw insufficient funds
Counter Array 1:  [1, 1]
```

Create Account Method

1. The white box used for the account create transactions was statement coverage.
2. This method creates a test case for each statement in the program. The purpose of the statement coverage test is to make sure that every statement in the program can be executed at least once. The program create_acc takes in two inputs: acc, and acc_name, for account number and account name. Both inputs are taken from the Merged transaction summary file. The program consists of two statements, where the first statement adds a new account entry into the back office’s master dictionary with the new account number, name and default balance, and the second statement just returning a zero value. One test case was created for this test set, since both statements of the program will be executed regardless of the test inputs. A counter was injected into the program to log if when either statement is executed.

```
# transaction create account
def create_acc(acc, acc_name):
    Master_Dict[acc] = {'balance': 500000, 'name': acc_name}
    wbCounter2[0] += 1
    return 0
```

3.

Statement	Inputs	Test	Input
1	7777777, JamesBond	T1	777777, JamesBond
2			

The test input from the transaction summary file can be seen in the figure below.

NEW 7777777 000 0000000 JamesBond

4. The one test case for the create account transaction ran successfully with no failures. The output of the test result can be seen in the figure below. As seen in the figure below the test counter array was incremented by 1, indicating that the statement in the program was executed successfully. This result was expected since there were no design constraints on the program, and since the contents of the program test inputs were already verified by the front end before being sent to the back office.

```
Counter Array 2:  [1]
```