Quality Assurance Assignment 5

Frontend Requirements Testing - Group 42

Alex Mason 20019045

Ted Munn 20001247

Marc De Verteuil 20005645

Source listing

The two pieces of functionality tested for the backend were the withdraw and account creation transactions. These pieces of code are found in the *backend.py* file. The four functions that work together to achieve this functionality are outlined below.

- backendController()
    - Takes a Transaction object and a master account list dictionary as input
    - Calls the appropriate function based on the transaction code and returns the updated master account list dict
- withdraw()
    - Takes a Transaction object and a master account list dictionary as input
    - Check that account has enough funds to complete the withdrawal
        - If there are enough funds, update the master account list dict
    - Return the master account list dict
- createAccount()
    - Takes a Transaction object and a master account list dictionary as input
    - Adds a new MasterAccount object to the master account list dictionary using the information in the Transaction argument
    - Returns the updated master account list dictionary
- updateBackend()
    - Gets a list and dictionary of transactions and accounts respectively by calling the getTransactionList() and getMasterAccountList() functions
    - Iterates through the transactions and calls the backend controller for each transaction
    - The updated master account list dictionary is written to the master_accounts_list.txt file using writeMasterAccountsFile()
    - The valid_account_list.txt file is updated using generateValidAccountList() and passing the updated master account list dictionary
    - Lastly the transaction_summary.txt file is cleared

Test cases

Withdraw

For the testing of the withdraw functionality, the path coverage white box method was used. The key condition to be covered here is in line 39 where the amount of funds in the account is checked to make sure the withdrawal is valid. The tests used to cover each direction this statement can take are outlined in the table below.

Test	**Purpose of test**	**Functions called in test**	**Key code segment**
test_withdraw_valid_amount	Test that the function handles a valid withdrawal amount.	withdraw() backendController()	if(current_account_balance - transaction.amount >= 0)
test_withdraw_invalid_amount	Test that the function handles an invalid withdrawal amount.	withdraw() backendController()	if(current_account_balance - transaction.amount >= 0)

Create Account

The statement coverage white box method was used for the testing of the create account functionality. Because of the relatively small number of lines used for the createAccount() method, statement coverage makes sense.

Test	**Purpose of test**	**Functions called in test**	**Key code segment**
test_create_account_success	Confirm the backend controller calls the correct function (createAccount()) and that the createAccount() function returns the correctly updated master_accounts_list dictionary	backendController() createAccount()	if transaction.transaction_code == "NEW": master_accounts_list[transaction.account_number_to] = MasterAccount(transaction.account_number_to, 0, transaction.account_name)

Test inputs

Note: The frontend does all of the validation for function input, therefore the backend assumes that passed transactions are valid.

Test	**Inputs**	**Description**
test_withdraw_valid_amount()	Transaction('WDR',1234567, 100, 0000000 , '***')	Tests that the if condition in withdraw handles a valid amount requesting to be withdrawn.
test_withdraw_invalid_amount()	Transaction('WDR',1234567, 10000, 0000000, '***')	Tests that the if condition in withdraw handles an invalid amount requesting to be withdrawn.
createAccount()	Transaction('NEW',1234567, 100, 0000000 , 'testaccount')	This transaction creates a new account. The front office handles validation so any transaction passed to the backend will be valid.

Test report

When creating an account you do not pass a balance to the new account, it always gets created with a balance of 0. Initially I used create account and asserted that the passed balance equalled the account balance but it failed. I changed the assertion to check the new account has a balance of zero, which reflects the actual system requirements