Assignment #4

Group 42

Alex Mason 20019045

Ted Mun 20001247

Marc De Veurteuil 20005645

Backend Architecture

The backend of our quinterac system will complete multiple tasks. The backend flow will be invoked when the program stops running. For the purposes of simplicity, one “day” will begin when the program is run and the “day” will end when the program ends, i.e. the quit command is used. To clarify, there can be multiple “sessions” in a day, i.e. a user can login and logout as many times as desired while the program is running. It is only when the quit command is called that the “day” will end, and the program will invoke the backend flow.

The backend architecture will accomplish the following tasks:

1. Apply the transactions from the day to the master accounts list and clear the transaction summary file
    a. Iterate through transactions in transaction summary file
    b. Apply transactions, one by one, to the master accounts list
    c. Handle any transactions that cannot be completed
    d. Update the master accounts list
2. Update the valid accounts file
    a. After the master accounts file is updated, update the valid accounts file to reflect the valid accounts in the master accounts file

Backend Flow

The system will start the backend processing after the “quit” command is entered into the command line. The user cannot be logged in and quit.

This causes the transaction_summary.txt file and master_accounts_list.txt file to be converted into a python list of Transaction objects and a dictionary of MasterAccount respectively.

The master_accounts_list dictionary is stored as a dictionary with the account number as the key and a MasterAccount object as the value.

The transaction_list iterated through and the transactions are processed using a transaction functions that has been created for each type of transaction.

The updateBackend() function is called, which sequentially calls getTransactionList() and getMasterAccountsList(). Each of two functions getTransactionList() and getMasterAccountsList() are found in transaction_summary.py and master_account.py

respectively. Each of these two functions do essentially the same thing, converting raw text file input from transaction_summary.txt and master_accounts_list.txt into a list object (transaction_list) and a dictionary object (master_accounts_list) respectively.

Next, updateBackend() iterates through the list of transactions recorded in the day. It updates the local master_accounts_list dictionary to reflect the transaction. This is accomplished by calling the backendController() function.

The backendController() function operates similarly to the controller() function in main. It takes a transaction and a master_account_list as input and calls the appropriate function to modify a copy of the passed master_accounts_list. It then returns the new version of the master_accounts_list with the appropriate transaction applied.

Once all of the transactions have been applied to the local master_accounts_list dictionary, they are written to the master_accounts_list.txt file. The new set of valid accounts are then generated using the generateValidAccountsList() function and written to the valid_account_list.txt file.

Backend Transaction Functions

Method	**Inputs**	**Outputs**	**Description**
deposit	Transaction, master_accounts_list	master_accounts_list	Checks that the deposit does not push the account over the allowed value then adds the amount from input to the balance
withdraw	Transaction, master_accounts_list	master_accounts_list	Checks that the account has enough funds to complete the withdrawal then removes the amount from input from the balance
transfer	Transaction, master_accounts_list	master_accounts_list	Uses the withdraw and deposit to implement. Withdraws from the source account and deposits in the destination account
create_account	Transaction, master_accounts_list	master_accounts_list	Adds an entry to the master accounts list file at the index of the account number
delete_account	Transaction,	master_accounts_list	Gets the account info from the

	master_accounts_list		master accounts list, checks that the balance is 0 and that the name input matches, then removes the account from the list
generate_valid_accounts_list	master_accounts_list	master_accounts_list	Overwrites the valid_accounts_list file with the master_accounts_list