CSCI 3060U Front End Reqs - Kershan Arulneswaran, Andrew Hunter, David Jeanes, Kim Yap (The Piggy Bank)

Table of Test Cases

Transaction	Test Name	Intention
misc	misc_01	Test if an invalid transaction is entered
	misc_02	Test if a blank transaction is entered
login	login_01	Test if user can actually login
	login_02	Test if something other than ‘standard’ or ‘admin’ is entered when prompted
	login_02a	Test if the ‘standard’ or ‘admin’ prompt is blank
	login_02b	Test if an invalid username is entered
	login_02c	Test if a blank username is entered
	login_03	Test if user can log in as a standard user (account holder)
	login_03a (part of login_03)	Test if account holder’s name is visible
	login_04	Test if user can log in as a bank administrator
	login_05	Test if user cannot log in during session if they are already logged in
logout	logout_01	Test if user is logged in
	logout_02	Test if information is written to transaction file after logout
	logout_03	Test if user can logout as admin
	logout_04	Test if user cannot make

		transactions after logging out
	logout_05	Test that a user can login after logging out
withdrawal	withdrawal_01	Test if the user attempts a withdraw without being logged in
	withdrawal_02	Test if an invalid account name is entered for withdrawal as an admin
	withdrawal_03	Test if a blank account name is entered for withdrawal as an admin
	withdrawal_04	Test if an invalid account number is entered
	withdrawal_05a	Test if a blank account number is entered
	withdrawal_05b	Test if a non number is entered as a bank account number
	withdrawal_05c	Test if the entered account is disabled
	withdrawal_06	Test if the user attempts to withdrawal $0
	withdrawal_07	Test if the user attempts to withdrawal a negative amount
	withdrawal_08	Test if the user enters a non number for their withdrawal amount
	withdrawal_09	Test if a blank withdrawal amount is entered
	withdrawal_10	Test if the bank account does not belong to the logged in user
	withdrawal_11	Test if more than $500 dollars is tried to be withdrawn in a single session
	withdrawal_12	Test if exactly $500 is tried to be withdrawn in a single session

	withdrawal_13	Test if $500.01 is tried to be withdrawn in a single session
	withdrawal_14	Test if the user tries to withdraw more money than they have in the account
	withdrawal_15	Test if the withdraw will leave the account with exactly $0
	withdrawal_16	Test if the withdraw will leave the account with -$0.01
	withdrawal_17	Test if the withdraw will leave the account with $0.01
	withdrawal_18	Test if a valid amount of money is withdrawn from an account by an admin
transfer	**Requirement**	**Should ask for the account holder's name (if logged in as admin)**
	transfer_01	Test if transfer is started without logging in
	transfer_02	Test for Valid Account Holder Name entered if admin
	transfer_03	Test for Blank Account Holder Name entered if admin (Deny)
	transfer_04	Test if Invalid random format entered (Deny)
	Requirement	**Should ask for the account number that money will be transferred from (as a text line)**
	transfer_05	Test for Valid Account Number under Account Holder Name entered when logged in as admin

	transfer_06	Test for Valid Account Number under username entered when logged in as user
	transfer_07	Test if can access Valid Account number from system but from other username when logged in as admin (Deny)
	transfer_08	Test if can access Valid Account number from system but from other username when logged in as user (Deny)
	transfer_09	Test for Blank Account Number entered if admin (Deny)
	transfer_10	Test for Blank Account Number entered if user (Deny)
	transfer_11	Test if Invalid random format entered for account number if admin (Deny)
	transfer_12	Test if Invalid random format entered for account number if user(Deny)

	Requirement	**Should ask for the account number that money will be transferred to (as a text line)**
	transfer_13	Test for Blank Account Holder Name entered if admin (Deny)
	transfer_14	Test if Invalid random format entered if admin (Deny)
	transfer_15	Test for Blank Account Holder Name entered if user (Deny)
	transfer_16	Test if Invalid random format entered if user (Deny)
	transfer_17	Test for Valid Account Number in System
	transfer_18	Test if the same account number under user if admin(Deny)
	transfer_19	Test if the same account number under user if user(Deny)
	Requirement	**Then should ask for the amount to transfer**
	transfer_20	Test for alphanumeric characters (Deny)
	transfer_21	Test for number decimal less than 0.01 (Deny)
	transfer_22	Test for **Negative** transfer amount. (Deny)
	transfer_23	Test for amount of transferer meeting minimum balance of 0.00 (Cannot transfer more than what is in account) for user
	transfer_24	Test for amount of transferer meeting minimum balance of 0.00 (Cannot transfer more than what is in account) for admin

	transfer_25	Test for 1000.01 (Deny for user)
	transfer_26	Test for 1000.01 (Allow for Admin)
	transfer_27	Test for 1000 (Allow for User)
	transfer_28	Test for 1000 (Allow for Admin)
	transfer_29	Test for multiple transfers in one session exceeding 1000 (Deny for User)
	Requirement	**Should save this information for the bank account transaction file(Saved as two lines on the transaction file the first being the transferer and the second being the transferee)**
	transfer_30	Test for first line transaction format
	transfer_31	Test for second line transaction format
	transfer_32	Test for both lines saved in transaction file (transfer transaction log should always come in pairs)
		Bank account that the funds are being transferred from must be a valid account for the account holder currently logged in. (Constraint)
		Bank account that the funds are being transferred to must be a valid account in the Bank System. (Constraint)

		o Maximum amount that can be transferred in current session is $1000.00 in standard mode (Constraint)
		o Account balance of both accounts must be at least $0.00 after transfer. (Constraint)
paybill	paybill_01	Test if user is logged in and can pay bills
	paybill_01a	Test if user cannot pay bills with an inactive account
	paybill_02	Test if user is an admin and can pay bills
	paybill_02a	Test if admin shares the account name
	paybill_03	Test if user shares a text line
	paybill_03a	Test if user shares an account number in that text line
	paybill_04	Test if company that will be paid exists
	paybill_04a	Test if company tag is either one of these strings: **EC, CQ, FI**
	paybill_05	Test if the amount the user wants to pay is less or equal to $0 ($0.00)
	paybill_05a	Test if the amount the user wants to pay is greater than $2000 ($2000.01)
	paybill_05b	Test if the amount the user wants to pay is equal to $0.01
	paybill_05c	Test if the amount the user wants to pay is equal to $2000

	paybill_06	Test if transaction is terminated if user selects ‘n’
	paybill_07	Test if transaction passes if user selects ‘Y’
	paybill_07a (same file)	Test if user has a balance
	paybill_07b (same file)	Test if user’s balance after input is negative, terminates otherwise
	paybill_07c (same file)	Test if user’s balance after input is non-negative
	paybill_07d (same file)	Test if transaction went through
deposit	deposit_01	Test if the user attempts a deposit while not logged in
	deposit_02	Test if an invalid account name is entered as an admin
	deposit_03	Test if a blank account name is entered as an admin
	deposit_04	Test if an invalid account number is entered
	deposit_05a	Test if a blank account number is entered
	deposit_05b	Test if a non-number account number is entered
	deposit_05c	Test if the entered account is disabled
	deposit_06	Test if the user attempts to deposit $0
	deposit_07	Test if the user attempts to deposit $0.01
	deposit_08	Test if the user attempts to deposit a negative amount
	deposit_09	Test if the user enters a non number for their deposit amount
	deposit_10	Test if a blank deposit amount is entered

	deposit_11	Test if the bank account does not belong to the logged in user
	deposit_12	Test if a valid amount of money is deposited to an account by a standard user
	deposit_13	Test if a valid amount of money is deposited to an account by an admin
	deposit_14	Test that the deposited funds cannot be used in this banking session
	deposit_15	Test if the deposit will leave the account with $99999.99
	deposit_16	Test if the deposit will leave the account with $100000.00 which exceeds the maximum amount an account can hold
create	create_01	Test if the user attempts create when not logged in
	create_02	Test if the user attempts a create transaction as a non admin
	create_03	Test if the given account holder name is blank
	create_04a	Test if the given account holder name is longer than 20 characters (21 characters)
	create_04b	Test account creation where the given account holder name is exactly 20 characters
	create_05	Test if the given account name is END_OF_FILE
	create_09	Test that the given account number is unique
	create_10	Test if the initial balance is left blank
	create_11	Test if the initial balance is negative

	create_12	Test if the initial balance is exactly equal to $99999.99
	create_13	Test if the initial balance is exactly $100000.00
	create_14	Test if the initial balance is -$0.01
	create_15	Test a valid account creation with an initial balance of $0
	create_16	Test that the account cannot be used in this session
delete	delete_01	Test if user is logged in
	delete_02	Test if user is logged in as admin
	delete_03	Test if user is prompted to enter account holder's name
	delete_04	Test if account name is left blank
	delete_05	Test if account name entered has an existing account
	delete_06	Test if user is prompted to enter account number
	delete_07	Test if account number is left blank
	delete_08	Test if account number entered matches account name
	delete_09	Test if user is given the option to cancel the deletion before the deletion occurs
	delete_10	Test if account is set to deleted in the transaction file
	delete_11	Test if user cannot make any transactions to deleted account
disable	disable_01	Test if user is logged in
	disable_02	Test if user is logged in as admin
	disable_03	Test if user is prompted to enter

		account holder’s name
	disable_04	Test if account name is left blank
	disable_05	Test if name entered has an existing account
	disable_06	Test if user is prompted to enter account number
	disable_07	Test if account number is left blank
	disable_08	Test if account number matches the account name
	disable_09	Test if account is already disabled
	disable_10	Test if account is set to disabled after transaction
	disable_11	Test if disabled account cannot make any transactions
changeplan	changeplan_01	Test if user is logged in
	changeplan_02	Test if user is logged in as admin
	changeplan_03	Test if user is prompted to enter name
	changeplan_04	Test if name input is left blank
	changeplan_05	Test to verify that the name entered has an existing account
	changeplan_06	Test if user is prompted to enter account number
	changeplan_07	Test if account number is left blank
	changeplan_08	Test to verify that the account number entered matches the account name
	changeplan_09	Test if user can change from non-student plan to student plan

	changeplan_10	Test if user can change from student plan to non-student plan
	changeplan_11	Test if transaction file is changed to show user's new payment plan

Test Plan Document

Organization

Tests will be organized in the following directory structure:

```
run_tests.sh                        (shell script which runs all the tests)
tests/
    bank_accounts.txt               (current bank account file for all tests)
    00_misc/
        …
    01_login/                       (each transaction gets its own directory)*
        login_01/                   (the tests also each get their own directory)
            input.txt               (stdin to be piped into the program for this test)
            output.txt              (expected stdout for this test)
            transactions.txt        (expected bank transaction file for this test)
        login_02/
            input.txt
            output.txt
            transactions.txt
        …
    02_logout/
        …
    03_withdrawal/
        …
    04_transfer/
    05_paybill/
    06_depoist/
    07_create/
    08_delete/
    09_disable/
    10_changeplan/
    test_results/                   (the results of each test run are stored for reporting
                                     and comparison purposes)
        2025-02-03_11:16:32/        (each run has a folder named the time it was run)
            01_login/
                login_01/           (the directory format matches the tests themselves)
                    output.txt      (the stdout of the program for the test run)
```

transactions.txt (the output transaction file for the test run)

*Transaction directories are numbered by the order they are tested in.

Running Tests

Tests will be run by a shell script in the project's root directory named run_tests.sh. It will execute all the tests in order, piping in the input.txt and storing the stdout and transactions log in a new directory in the test_results directory (see above directory layout). From there the output.txt and the transactions.txt from the test run will be compared to their expected results using diff for both. If either of these files differ from their expected contents, the test fails. By default the script will continue to run all tests even if an earlier test fails but by using the -s command line argument the script will stop on test failure. To test only specific transactions the names of the transaction test directories can be passed to run_tests.sh as command line arguments.

Assumptions

Assume that the ‘end of session’ line in the transaction log does not include the account holder's name.

There were a number of points in the requirements which were clarified by the Big Bank CEO. Notably there is a jump from test create_05 to create_09 because of clarification that the account number during creation should be generated instead of entered.
In addition clarification was required on the format of the transaction file for transfer transactions as they operate on 2 accounts.