ATT
CSCI3060U Project Phase #3
Front End Requirements Testing

Taylor Smith
Alexandar Mihaylov
Talha Zia

DEPENDENCIES

Our code uses regex which only properly works with g++ 4.9 and above

g++-4.9 (Regex)

```
sudo add-apt-repository ppa:ubuntu-toolchain-r/test
sudo apt-get update
sudo apt-get install g++-4.9
```

Update the alternatives

```
sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-4.9 60
sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-4.8 40
```

Switch with

```
sudo update-alternatives --config g++
```

Running g++-4.9

```
g++-4.9 -std=c++11 sample.cpp
```

FAILURE LOG

Global

Test Number	What is Tested	Nature of Failure	Error in Code	Actions taken to Fix
GLOBAL01	VARIOUS	no functions match account names because they are in different formats in the accounts file and the test cases	error in tests not in code	Capitalize all account names in the test files.
GLOBAL02	VARIOUS	No transaction files match because of improper formatting.	error in tests not in code	Add a trailing whitespace to the misc section of the expected output transaction file.
GLOBAL03	VARIOUS	No output files match because of incorrect prompts		Update output files to display proper prompts.

Login

Test Number	What is Tested	Nature of Failure	Error in Code	Actions taken to Fix
login01	Check all commands **do not work** before	Input file had an	Error was in the tests	Updated test files to

	user logs in: logout, withdrawal, transfer, paybill, deposit, create, delete, disable, changeplan, and check for gibberish	unneeded line. Output file didn't consider the redisplay of the welcome prompt.	not in the code.	represent the prompts that are actually displayed.
login02	Check that you can login as a **standard** user, logout successfully and an "empty" transaction file is generated.	Logout was not properly indicating the account type logging out.	The logout handler was using a fixed value for the misc section of the tf.	Updated logout handler to check if user is an admin or standard user. Also updated the output test file.
login03	Check that you can login as an **admin** user, logout successfully and an "empty" transaction file is generated.	Passes after global changes applied	Error was in the tests not in the code.	Test passed after the global changes were applied
login04	Check that after logging in as a **standard** user you **cannot** login again.	Not printing an already logged in error message	Command validator was not properly checking if user was already logged in and tried to log in again.	Updated command validator to print error message when already logged in
login05	Check that after logging in as an **admin** user you **cannot** login again.	Passes after global changes	Error was in the tests not in the code.	Test passed after the global changes

				were applied
login06	Check that a **standard** user can login and then use **all** unprivileged commands.	Test needed to be changed because we needed to give in a bad input	Error was in the test not in the code	Test passed after global changes were applied
login07	Check that a standard user can login and **cannot** use **all** privileged commands.	Command validator was not accurately printing privileged access prompt	No check to see if standard user, tries to access a privileged command	Implement that check in CommandValidator.
login08	Check that an admin user **can** use **all** privileged commands			
login09	Bad input before login, and after login fails gracefully	No check for non valid transactions .	Command validator was not checking for transactions not supported	Write a check to look for unsupported commands
login10	User enters an account holder name that does not exist in the current accounts file, user is given an error	Extra printed line in the output file	extra println that was used for debugging	removed the extra println.

Logout

Test Number	**What is Tested**	**Nature of Failure**	**Error in Code**	**Actions taken to Fix**

logout01	Check that a user cannot logout before they have logged in	Passes after global changes	Error was in the tests not in the code.	Test passed after the global changes were applied
logout02	Once a user has logged in and performed **0 transactions**, then logs out, check that there was nothing regarding transactions written to the transaction file	Passes after global changes	Error was in the tests not in the code.	Test passed after the global changes were applied
logout03	Once a user has logged in and performed **1 transaction**, then logs out, assert that transaction was written to the transaction file	Withdrawal was asking for user's name even when logged in as a standard user.	No check if user is standard or Admin in withdrawlHandler	Implement the check in withdrawalHandler for if a user is standard or admin.
logout04	Once a user has logged in and performed **2 or more transactions**, then logs out, assert that transactions were written to the transaction file	Transfer was asking for user's name even when logged in as a standard user. No lookup of second username in transfer.	No check if user is standard or Admin in transferHandler. Second account name was hardcoded for prototype.	Implement the check in transferHandler for if a user is standard or admin. Dynamically adjust the second account name.
logout05	User performs a login and logout session, then is able to log in once again followed by a successfull logout	Passes after global changes	Error was in the tests not in the code.	Test passed after the global changes were applied

Withdrawal Test

Test Number	What is Tested	Nature of Failure	Error in Code	Actions taken to Fix
withdrawal01	The account holder's name and account number match and withdrawal proceeds as intended	Passes after global changes applied	Missing validator to check the format of the amount entered.	Had to implement a amount validation function to make sure the amount entered is in the proper format
withdrawal02	The account holder's name and account number **do not** match, error thrown and user is sent back to main menu	Passes after global changes applied	Missing validator to check if the account holder's name matches the account number	Had to implement an isValidAccount function to make sure the account number matches the account name
withdrawal03	User supplies a **negative** value to withdrawal and is rejected	Passes after global changes applied	Error was in the test not in the code	Test passed after the global changes were applied
withdrawal04	The user supplies random input (ex: #badinput!) into the amount to withdraw that is not in the form **\d+\.\d{2}** and is **rejected**	Passes after global changes applied	Error was in the test not in the code	Test passed after the global changes were applied
withdrawal05	User supplies 1.00 value to withdrawal and	Passes after global changes are applied	Missing paper currency validtor to check if the	Had to implement isPaperCurren

	is rejected since it is not a paper currency value		amount entered matches paper currency	cy function to make sure valid paper currency numbers can be input
withdrawal06	User supplies 5.00 value to withdrawal and is accepted since it is a multiple of 5	Passes after global changes are applied	Error was in the test not in the code	Test passed after the global changes were applied
withdrawal07	A **standard user** supplies 400.00 dollars to withdrawal and is accepted since it is under the 500 limit	Passes after global changes are applied	Error was in the test not in the code	Test passed after the global changes were applied
withdrawal08	A **standard user** supplies 500.00 dollars to withdrawal and is accepted since it is under the 500 limit (account has more than $500.10 in it)	Passes after global changes are applied	Error was in the test not in the code	Test passed after the global changes were applied
withdrawal09	A **standard user** supplies 505.00 dollars to withdrawal and is **rejected** since it is above the 500 limit	Lack of validation + global changes	Did not have a function to check if the amount is less than 500	Implemented WithdrawalHandler::isUnderWithdrawalLimit that checks if the amount is under the limit
withdrawal10	An **admin user** supplies 505.00 dollars to withdrawal and is **accepted** since admins do not have limits	Was getting an error saying that the amount is greater than the limit	missing check for when the user is an admin	add a condition to the check && !current_status.is_admin

withdrawal11	A **standard user** supplies 400.00 dollars to withdrawal then after attempting to withdrawal another 105.00 in the same day, is **rejected** and given an error since the combined amount is > 500.00	Was succeeding even though the combined withdrawals exceeded the daily limit	Did not have a function that aggregated daily withdrawals and checked if they are under the daily limit	made a AccountsDatabase::updateWithdrawnAmount function that updates the daily withdrawn amount on every withdrawal
withdrawal12	An **admin** supplies 400.00 dollars to withdrawal then after attempting to withdrawal another 105.00 in the same day, is **accepted**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
withdrawal13	A **standard user** supplies 400.00 dollars to withdrawal then after attempting to withdrawal another 100.00 in the same day, is **accepted** (account has more >= 500.20 in it)	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
withdrawal14	Standard student account withdrawals 5.00 from an account and is **accepted**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
withdrawal15	Standard student account withdrawals 5.00 from an account with 5.05 in it and is **accepted**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
withdrawal16	Standard student account withdrawals 5.00 from an account	The transaction was accepted when it should	There was no check to see if the withdrawal +	isWithdrawalPossible was implemented to ensure that

	with 5.00 in it and is **rejected**	have been rejected	fee was more than the limit	the withdrawal is possible
withdrawal17	admin account withdrawals 5.00 from an account with 5.00 in it and is **accepted**	Was rejected when it should have been accepted	There was no check to see if the user is an admin	added check && current_status.is_admin
withdrawal18	Standard non-student account withdrawals 5.00 from an account and is **accepted**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
withdrawal19	Standard non-student account withdrawals 5.00 from an account with 5.10 in it and is **accepted**	The transaction was rejected when it should have been accepted	The issues was that there was a floating point rounding error and instead of a 0 it was a very small number e-8	added a roundf(x*100)/100.00 to fix the floating point rounding issue
withdrawal20	Standard non-student account withdrawals 5.00 from an account with 5.05 in it and is **rejected**	Test passes when it should have failed	Error not in code but in cbaf file	added a standard non-student account with balance of 5.05
withdrawal21	admin non-student account withdrawals 5.00 from an account with 5.05 in it and is **accepted**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong

Transfer

Test Number	**What is Tested**	**Nature of Failure**	**Error in Code**	**Actions taken to Fix**
transfer01	The first account (sender) number	Succeed when it should be giving an invalid input	No check to see if the given input	implemented AccountsDatabase::number

	and account nameholder **match**		is a valid account number	Exists function to check for valid account number
transfer02	The first account (sender) number and account nameholder **do not** match and an error is thrown	Get invalid input instead of account # not matching the number	No check to see if acc # matches account name	Used isValidAccount to check if they are infact the same account
transfer03	The second account (receiver) supplied by the user exists and user proceeds as normal	Get invalid transaction command	Was not checking if that amount entered is a of valid format	used validateAmountFormat to check if the amount is actually in the correct format
transfer04	The second account (receiver) supplied by the user **does not** exist	Bad input instead of account holder does not match	Did not have the correct account number in the test case	fixed the account number to be the right one
transfer05	User supplies a **negative** value to transfer and is **rejected**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
transfer06	The user supplies random input (ex: #badinput!) into the amount to transfer that is not in the form **\d{5}\.\d{2}** and is **rejected**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
transfer07	A **standard user** supplies 1.00 dollars to transfer and is accepted	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong

	since it is under the 1000.00 limit			
transfer08	A **standard user** supplies 1000.00 dollars to transfer and is accepted since it is under the 1000.00 limit (account has more than 1000.10 in it)	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
transfer09	A **standard user** supplies 1001.00 dollars to transfer and is **rejected** since it is above the 1000.00 limit	The test was accepted and rather than being rejected	There was no check as to the amount that a user can transfer	implemented a function isUnderTransferLimit() that checks if the amount given is under the transfer limit
transfer10	An **admin user** supplies 1001.00 dollars to transfer and is **accepted** since admins do not have limits	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
transfer11	A **standard user** supplies 900.00 dollars to transfer then after attempting to transfer another 101.00 in the same day, is **rejected** and given an error since the combined amount is > 1000.00	The test was succeeding even though we expected a rejection	There was no check or any variable to keep track of how much money was transferred for the day	Implemented isTransferPossible and updateTransferedAmount to check if the transfer should be allowed, and to update the daily transaction withdrawn on success respectively

transfer12	An **admin** supplies 900.00 dollars to transfer then after attempting to transfer another 101.00 in the same day, is **accepted**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
transfer13	A **standard user** supplies 900.00 dollars to transfer then after attempting to transfer another 100.00 in the same day, is **accepted** (account has more >= 1000.10 in it)	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
transfer14	Standard student account transfers 5.00 from an account and is **accepted**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
transfer15	Standard student account transfers 5.00 from an account with 5.05 in it and is **accepted**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
transfer16	Standard student account transfers 5.00 from an account with 5.00 in it and is **rejected**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
transfer17	admin transfers 5.00 from a student account with 5.00 in it and is **accepted**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong

transfer18	Standard non-student account transfers 5.00 from an account and is **accepted**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
transfer19	Standard non-student account transfers 5.00 from an account with 5.10 in it and is **accepted**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
transfer20	Standard non-student account transfers 5.00 from an account with 5.00 in it and is **rejected**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
transfer21	admin account transfers 5.00 from a non-student account with 5.00 in it and is **accepted**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
transfer22	Transfers must be strictly greater than 0. Should be **rejected**	Was succeeding when a rejection was expected	There was no check to see if the value was strictly above 0	Add the condiditon if (stof(amount) <= 0) right after we check if it is a valid amount

Pay Bill

Test Number	What is Tested	Nature of Failure	Error in Code	Actions taken to Fix

paybill01	Account name and number **match** from the current accounts file	Succeeded to completion even though a #badinput is entered to prevent it from doing so	No check to see if a valid company name was entered	isValidCompan() function was implemented to do that check
paybill02	Account name and number **do not match** from the current accounts file, user is sent back to the menu	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
paybill03	User provides “EC” as the company name once prompted and is **able** to proceed with paybill	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
paybill04	User provides “CQ” as the company name once prompted and is **able** to proceed with paybill	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
paybill05	User provides “TV” as the company name once prompted and is **able** to proceed with paybill	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
paybill06	User provides “AA” as the company name once prompted and is **unable** to proceed with paybill since “AA” is not in the	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong

	list of companies, user is shown an error and taken back to the main menu			
paybill07	User supplies a **negative** value to pay and is **rejected**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
paybill08	The user supplies random input (ex: #badinput!) into the amount to pay that is not in the form **\d+\.\d{2}** and is **rejected**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
paybill09	A **standard user** supplies 1.00 dollar to pay and is accepted since it is under the 2000.00 limit	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
paybill10	A **standard user** supplies 2000.00 dollars to pay and is accepted since it is under the 2000.00 limit (account has more than $2000.10 in it)	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
paybill11	A **standard user** supplies 2001.00 dollars to pay and is **rejected** since it is above the 2000.00 limit	Succeeds even though the amount exceeds the limit, the transaction is accepted when it should be rejected	There was no check performed to see if the amount was greater than the allowed company max	isUnderPaybillLimit() function had to be implemented to perform the check

paybill12	An **admin user** supplies 2001.00 dollars to pay and is **accepted** since admins do not have limits	Failed since there was an type in the account number in the expected tf file	Error was in the test not in the code	Fixed the typo so that it accurately represented the right account
paybill13	A **standard user** supplies 1900.00 dollars to pay then after attempting to pay another 101.00 to the **same** bill holder in the same day, is **rejected** and given an error since the combined amount is > 2000.00	The test was succeeding even though we expected a rejection	There was no check or any variable to keep track of how much money was paid for the day	Implemented isPaybillPossible and updatePaybillAmount to check if the paybill amount should be allowed, and to update the daily paid amount on success respectively
paybill14	A **standard user** supplies 1900.00 dollars to pay then after attempting to pay another 101.00 to the **another** bill holder in the same day, is **accepted**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
paybill15	An **admin** supplies 1900.00 dollars to pay then after attempting to pay another 101.00 to the same bill holder in the same day, is **accepted**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong

paybill16	A **standard user** supplies 1900.00 dollars to pay then after attempting to pay another 100.00 in the same day, is **accepted** (account has more >= 2000.10 in it)	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
paybill17	Standard student account pays 5.00 from an account with sufficient funds is **accepted**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
paybill18	Standard student account pays 5.00 from an account with 5.05 in it and is **accepted**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
paybill19	Standard student account pays 5.00 from an account with 5.00 in it and is **rejected**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
paybill20	admin account pays 5.00 from a student account with 5.00 in it and is **accepted**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
paybill21	Standard non-student account pays 5.00 from an account with sufficient funds and is **accepted**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong

paybill22	Standard non-student account pays 5.00 from an account with 5.10 in it and is **accepted**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
paybill23	Standard non-student account pays 5.00 from an account with 5.00 in it and is **rejected**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
paybill24	admin non-student account pays 5.00 from an account with 5.00 in it and is **accepted**	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong

Deposit

Test Number	**What is Tested**	**Nature of Failure**	**Error in Code**	**Actions taken to Fix**
deposit01	Account name and number **match** from the current accounts file	invalid program path due to features still needing to be implemented.	Amount wasn't being checked, neither was if the name and number match.	Implemented check for correct amount syntax and checks for name matching number
deposit02	Account name and number **do not match** from the current accounts file, user is sent back to the menu	Test input was wrong	Error was in the tes not in the code.	Updated the name in the input and the transaction file to represent the change.

deposit03	Standard account deposits 0.01 into an account with $0.00 balance, and is **rejected** since it does not cover the 0.1/0.05 fee, user is taken back to menu	Succeeded when it should have failed.	There was no condition to check if the current balance - fee < 0.	Implemented two functions, isDepositPossible and updateDepositAmount,
deposit04	admin account deposits 0.01 into an account with $0.00 balance, and is **accepted**	In “misc” variable there was a leftover ‘a’ from the prototype.	leftover variable from prototype	removed the leftover variable.
deposit05	User deposits 1.00 into their account of 5.00 balance and within the same session attempts to withdrawal 5.00 but is **rejected**	Passes after global changes were applied	Error was in the test not in the code	Expected transaction file was wrong.
deposit06	A deposit of 1.00 is added to an account with a balance of 99999.99 it should be **rejected** since it exceeds the maximum value an account can hold.	We didn’t have a check when the maximum amount was succeeded.	No validation for max account balance.	Implemented a validation for maximum account balance.

Create

Test Number	**What is Tested**	**Nature of Failure**	**Error in Code**	**Actions taken to Fix**

create01	Once prompted to enter account holder name, enter in a string of characters “A” that are of length 20. Check that the system created the account holder with 20 “A”s	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong, extra letter in the name
create02	Once prompted to enter account holder name, enter in a string of characters “A” that are of length 21. Check that the system created the account holder and truncated it to only 20 “A”s	An account with more than 20 char name was created.	No check to ensure maximum string length.	Implemente d a check to shorten the strong if it is too long.
create03	When prompted to enter in an initial balance, input 99999.99 and the transaction should continue without any error	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong, extra letter in the name
create04	When prompted to enter in an initial balance, input 100000. and the transaction should **reject** it, display an error and send the	Account was created with initial balance higher than limit	No check if the user was inputting too much as an initial balance	Checks balance and errors if its too high.

	user back to main menu			
create05	Create an account from start to finish, and within the same session attempt to withdrawal, transfer from/to, paybill, and deposit to the created account	No longer a valid test as accounts cannot be used in the same day.		

Delete

Test Number	What is Tested	Nature of Failure	Error in Code	Actions taken to Fix
delete01	Once the user is prompted to enter the account name to be deleted, fill it with one that exists in the current bank accounts file and the transaction should **proceed** The user enters in a bank account number that **matches** the account holder name based on the current bank accounts file The account is deleted.	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong

delete02	Once the user is prompted to enter the account name to be deleted, fill it with one that **does not** exist in the current bank accounts file and the transaction should be **rejected**	Transaction happens when it shouldn't	DisableHandler doesn't check if the accountholder name is a valid one	update to properly take in name and make sure it is a valid one.
delete03	The user enters in a bank account number that **does not match** the account holder name based on the current bank accounts file	The transaction passes when it shouldn't.	There is no check to ensure that Account name and number match	Implement a check in the disable handler to ensure account name and number match.
delete04	Once an account deletion has been deleted, and within the same session attempt to withdrawal, transfer from/to, paybill, and deposit to the created account, this should result in a **rejection** and an error by the system	Account was usable after being deleted	Account wasn't being deleted.	Deleted the account

Disable

Test Number	What is Tested	Nature of Failure	Error in Code	Actions taken to Fix
disable01	Once prompted to enter accountholder name, provide one that **exists** and the user should be able to proceed with the transaction. Provide an account number that matches the accountholder name provided, transaction should proceed as intended	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong
disable02	Once prompted to enter accountholder name, provide one that **does not** exist and the transaction should be **rejected** and the user taken to the main menu	Transaction happens when it shouldn't	DisableHandler doesn't check if the accountholder name is a valid one	update to properly take in name and make sure it is a valid one.
disable03	Provide an account number that **does not** matche the accountholder name provided, transaction should be **rejected** and	The transaction passes when it shouldn't.	There is no check to ensure that Account name and number match	Implement a check in the disable handler to ensure account name and number match.

	user taken to the main menu			
disable04	After an account has been disabled, within the same session attempt to withdrawal, transfer, paybill, deposit, from that account. All transactions should be **rejected** gracefully with their respective errors.	Input file didn't correctly navigate the program because of no checks for disabled accounts.	Wasn't correctly disabling the account or checking if an account was disabled in the other transactions.	Implements checks in other transaction s against disabled accounts. Properly disable accounts.
disable05	An account that exists and is already disabled is attempted to be disabled again.	Account that was already disabled was being re-disabled	No check to see if the account being disabled was already disabled.	Implement ed a check in the disable handler to see if the account given is already disabled.

Enable

Test Number	**What is Tested**	**Nature of Failure**	**Error in Code**	**Actions taken to Fix**
enable01	Once prompted to enter accountholder name, provide one that **exists** and the user should be able to	Account name in expected transaction files was wrong. And the account number in the input file was wrong.	The error was in the tests not in the code	Adjusted the tests to the proper name and number.

	proceed with the transaction. Provide an account number that matches the accountholder name provided, transaction should proceed as intended			
enable02	Once prompted to enter accountholder name, provide one that **does not** exist and the transaction should be **rejected** and the user taken to the main menu	Transaction happens when it shouldn't	EnableHandler doesn't check if the accountholder name is a valid one	update to properly take in name and make sure it is a valid one.
enable03	Provide an account number that **does not** match the accountholder name provided, transaction should be **rejected** and user taken to the main menu	The transaction passes when it shouldn't.	There is no check to ensure that Account name and number match	Implement a check in the enable handler to ensure account name and number match.
enable04	After an account has been enabled, within the same session attempt to withdrawal, transfer, paybill, deposit, from	Account number in input file was the wrong one and some of the misc info was set to A when it shouldn't have been.	Error in tests not in code.	Account number changed to the correct ones and incorrect flags removed.

	that account. All transactions should work as intended as if the account was never disabled.			
enable05	An account that exists and is already enabled is attempted to be enabled again.	Account that was already enabled was being re-enabled	No check to see if the account being enabled was already enabled.	Implemented a check in the enable handler to see if the account given is already enabled.

Change Plan

Test Number	What is Tested	Nature of Failure	Error in Code	Actions taken to Fix
changeplan01	Once prompted to enter accountholder name, provide one that **exists** and the user should be able to proceed with the transaction. Provide an account number that matches the accountholder name provided, transaction should proceed as intended	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong

changeplan02	Once prompted to enter accountholder name, provide one that **does not** exist and the transaction should be **rejected** and the user taken to the main menu	Logout wasn't called because the transaction was completed.	Didn't check if the name existed	Implemented a check in changeplan handler to ensure name exists in the database and properly take in the name.
changeplan03	Provide an account number that **does not** match the accountholder name provided, transaction should be **rejected** and user taken to the main menu	The transaction passes when it shouldn't.	There is no check to ensure that Account name and number match	Implement a check in the changeplan handler to ensure account name and number match.
changeplan04	After a successful changeplan of an account, within the same session run the changeplan transaction again on the same account	Passes after global changes are applied	Error was in the test not in the code	Expected transaction file was wrong