(CSCI 3060/SOFE 3980U - Winter 2016
Course Project Assignment #1 - Front End Requirements

Denesh Parthipan (100536986), Luisa Rojas-Garcia (100518772), Truyen Truong (100516976)

Note: The test environment for all test cases is Linux Operating System.

Test Case Name: Login

Description: Starts a Front end session. Should ask for the kind of session.

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**		EXPECTED OUTPUT

1
(1)
	Check if command is `login`	`login`
		`admin`
		`logout`
			`login`
			`standard`
			`acc_holder`
			`logout`
				If command is `login`, continue.
				If it is not an existing command, show error message. User may use the `help` command.

2 (2)	Check if there is an account already in session.			If there is a user already logged in, display an error message. Otherwise, prompt user to enter the account mode.
3 (3)	Check if account mode (`admin` or `standard`) is valid.			If `admin` is the input, login is successful. Inform the user they are now logged in and start reading in the current bank accounts file and as well as start accepting transactions. If `standard` is the input, prompt for `acc_holder`. If the input is neither `admin` nor `standard`, show an error message. User may attempt to login again.
4 (4)	Check if the `acc_holder` exists in the Bank System in the case of a standard user only. `acc_holder` May not be "END_OF_FILE".			If `acc_holder` exists, login is successful. Inform the user they are now logged in and start reading in the current bank accounts file and as well as start accepting transactions.

Test Case Name: Logout

Description: Ends a Front End session.

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**	EXPECTED OUTPUT

1
(5)
	Check if command is `logout`	`login`
		`admin`
		`logout`
			If command is `logout`, continue.
			If it is not an existing command, show error message. User may use the `help` command.

2 (6)	Check if there is an account in session.		If there is a user already logged in, successfully logout. Otherwise, display an error message.

Test Case Name: Create
Description: Create a new bank account with an initial balance. Privileged transaction.
Pre-requirement: User <u>must</u> be logged into an account. Refer to Login test case.

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**	EXPECTED OUTPUT

1
(7)
	Check if the command is `create.`	`login`
		`admin`
		`create`
		`new_acc_holder`
		`init_balance`
		`logout`
			If command is `create`, continue.
			If it is not an existing command, show error message. User may use the `help` command.

2 (8)	Check that user is logged in.		If user is logged in, continue. Otherwise, display an error message.
3 (9)	Check that current logged in user is an administrator.		If account mode is `admin`, prompt user for `new_acc_holder`. Otherwise, display an error message.
4 (10)	Check that `new_acc_holder` is a valid name: - No longer than 20 characters. - Only letters, numbers and hyphens. - Can not be “admin” or “END_OF_FILE” - No underscores		If `new_acc_holder` is valid, then continue. If `new_acc_holder` is longer than 20 characters, truncate account name and then continue. Otherwise, display an error message. Prompt user for `new_acc_holder` again.
5 (11)	Check that `new_acc_holder` is unique; hasn’t been		If `new_acc_holder` is valid, prompt user for `init_balance`.

	used before.		Otherwise, display an error message. Prompt the user for `new_acc_holder` again.
6 (12)	Check that `init_balance` is a valid amount. - No more than 99999.99		If `init_balance` is valid, create new account. Inform the user they have successfully created an account and save the information provided to the bank account transaction file.

Test Case Name: Delete

Description: Delete a bank account. Privileged transaction.

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**		EXPECTED OUTPUT

1
(13)
	Check that the command is `delete`.	`login`
		`admin`
		`delete`
		`acc_holder`
		`acc_number`
		`yes`
		`logout`
			`login`
			`admin`
			`delete`
			`acc_holder`
			`acc_number`
			`no`
			`logout`
				If command is `delete`, continue.
				If it is not an existing command, show error message. User may use the `help` command.

2 (14)	Check that user is logged in.			If user is logged in, continue. Otherwise, display an error message.
3 (15)	Check that current logged in user is an administrator.			If account mode is `admin`, prompt user for `acc_holder`. Otherwise, display an error message.
4 (16)	Check that `acc_holder` exists in the Bank System.			If `acc_holder` exists, prompt user for `acc_number`. Otherwise, display an error message. Prompt user for `acc_holder` again.
5 (17)	Check that `acc_number` corresponds to `acc_holder`.			If they match, prompt user for deletion confirmation prompt. Otherwise, display an error message. Prompt user for `acc_number` again.
6 (18)	Check if the input for the confirmation prompt is valid (`yes`/`y` or `no`/`n`)			If `yes` or `y`, then account is deleted successfully and the user is informed of this. save the information provided to the bank account transaction file. If `no` or `n`, then the deletion process gets cancelled.

				If none of the above, display an error message stating the 4 options available. User is prompted for confirmation again.

Test Case Name: Disable

Description: Disable a bank account. Privileged transaction.

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**		EXPECTED OUTPUT

1
(19)
	Check that the command is `disable`.	`login`
		`admin`
		`disable`
		`acc_holder`
		`acc_number`
		`yes`
		`logout`
			`login`
			`admin`
			`disable`
			`acc_holder`
			`acc_number`
			`no`
			`logout`
				If command is `disable`, continue.
				If it is not an existing command, show error message. User may use the `help` command.

2 (20)	Check that user is logged in.			If user is logged in, continue. Otherwise, display an error message.
3 (21)	Check that current logged in user is an administrator.			If account mode is `admin`, prompt user for `acc_holder`. Otherwise, display an error message.
4 (22)	Check that `acc_holder` exists in the Bank System.			If `acc_holder` exists, prompt user for `acc_number`. Otherwise, display an error message. Prompt user for `acc_holder` again.
5 (23)	Check that `acc_number` corresponds to `acc_holder`.			If they match, prompt user for disablement confirmation prompt. Otherwise, display an error message. Prompt user for `acc_number` again.
6 (24)	Check that the account hasn't already been disabled.			If the account is not disabled, continue. Otherwise, display an error message and abort disablement.
7 (25)	Check if the input for the confirmation prompt is valid (`yes`/y or `no`/`n`)			If `yes` or `y`, then account is disabled successfully and the user is informed of this. Save the information provided to the bank account transaction file *immediately*.

				Change bank account from active (A) to disabled (D). If `no` or `n`, then the disablement process gets cancelled. If none of the above, display an error message stating the 4 options available. User is prompted for confirmation again.

Test Case Name: Enable
Description: Enable a bank account. Privileged transaction.

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**		EXPECTED OUTPUT

1
(26)
	Check that the command is `enable`.	`login`
		`admin`
		`enable`
		`acc_holder`
		`acc_number`
		`yes`
		`logout`
			`login`
			`admin`
			`disable`
			`acc_holder`
			`acc_number`
			`no`
			`logout`
				If command is `enable`, continue.
				If it is not an existing command, show error message. User may use the `help` command.

2 (27)	Check that user is logged in.			If user is logged in, continue. Otherwise, display an error message.
3 (28)	Check that current logged in user is an administrator.			If account mode is `admin`, prompt user for `acc_holder`. Otherwise, display an error message.
4 (29)	Check that `acc_holder` exists in the Bank System.			If `acc_holder` exists, prompt user for `acc_number`. Otherwise, display an error message. Prompt user for `acc_holder` again.
5 (30)	Check that `acc_number` corresponds to `acc_holder`.			If they match, prompt user for disablement confirmation prompt. Otherwise, display an error message. Prompt user for `acc_number` again.
6 (31)	Check that the account is actually disabled.			If the account is disabled, continue. Otherwise, display an error message and abort enablement.
7	Check if the input for			If `yes` or `y`, then account is enabled

(32)	the confirmation prompt is valid (`yes`/`y` or `no`/`n`)			successfully and the user is informed of this. Save the information provided to the bank account transaction file *immediately*. Change bank account from disabled (D) to active (A). If `no` or `n`, then the enablement process gets cancelled. If none of the above, display an error message stating the 4 options available. User is prompted for confirmation again.

Test Case Name: Change Plan
Description: Change the transaction payment plan for an account. Privileged transaction.

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**	EXPECTED OUTPUT

1
(33)
	Check that the command is `changeplan`.	`login`
		`admin`
		`changeplan`
		`acc_holder`
		`acc_number`
		`logout`
			If command is `changeplan`, continue.
			If it is not an existing command, show error message. User may use the `help` command.

2 (34)	Check that user is logged in.		If user is logged in, continue. Otherwise, display an error message.
3 (35)	Check that current logged in user is an administrator.		If account mode is `admin`, prompt user for `acc_holder`. Otherwise, display an error message.
4 (36)	Check that `acc_holder` exists in the Bank System.		If `acc_holder` exists, prompt user for `acc_number`. Otherwise, display an error message. Prompt user for `acc_holder` again.
5 (37)	Check that `acc_number` corresponds to `acc_holder`.		If they match, changes plans successfully and the user is informed of this. Save the information provided to the bank account transaction file *immediately*. Change account payment plan from non-student (N - default) to student

			(S) or vice versa. Otherwise, display an error message. Prompt user for `acc_number` again.

Test Case Name: Deposit
Description: Deposit money into an account.

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**		EXPECTED OUTPUT

1
(38)
	Check that the command is `deposit`.	`login`
		`admin`
		`deposit`
		`acc_holder`
		`acc_number`
		`amount_dep`
		`logout`
			`login`
			`standard`
			`deposit`
			`acc_number`
			`amount_dep`
			`logout`
				If command is `deposit`, continue.
				If it is not an existing command, show error message. User may use the `help` command.

2 (39)	Check that user is logged in.			If user is logged in, continue. Otherwise, display an error message.
3 (40)	Check if account mode (`admin` or `standard`) is valid.			If account mode is `admin`, prompt user for `acc_holder`. if account mode is standard, skip `acc_holder` and prompt user for `acc_number`. Otherwise, display an error message.
4 (41)	Check that `acc_holder` exists in the Bank System.			If `acc_holder` exists, prompt user for `acc_number`. Otherwise, display an error message. Prompt user for `acc_holder` again.
5 (42)	For administrators, check that `acc_number` corresponds to `acc_holder`. For standard users, check that the `acc_number` corresponds to the currently logged in `acc_holder`.			If they match, then account changes plans successfully and the user is informed of this. Save the information provided to the bank account transaction file. Change account payment plan from student (SP) to non-student (NP) or vice versa. Otherwise, display an error message. Prompt user for `acc_number` again.
6	Check that `amount_dep`			If `amount_dep` is valid, then continue.

(43)	is valid. - Must be in the form of #####.##. No dollar (or any other) signs. - `amount_dep` + balance - fee <= 99999.99 - Only positive numerical values *bigger* than 0.			Otherwise, display an error message. Prompt user for `amount_dep` again.
7 (44)	Check that the account is enabled/active.			If the account is enabled/active, then the amount specified is deposited into the account and the user is informed of this. Save the information provided to the bank account transaction file. Otherwise, Otherwise, display an error message. Deposit process gets aborted.

Test Case Name: Withdrawal

Description: Withdraws money from an account.

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**		EXPECTED OUTPUT

1
(45)
	Check that user is logged in.	`login`
		`admin`
		`withdrawal`
		`acc_holder`
		`acc_number`
		`amount_wit`
		`logout`
			`login`
			`standard`
			`withdrawal`
			`acc_number`
			`amount_wit`
			`logout`
				If user is logged in, continue.
				Otherwise, display an error message.

2 (46)	Check if account mode (`admin` or `standard`) is valid.			If account mode is `admin`, prompt user for `acc_holder`. if account mode is standard, skip `acc_holder` and prompt user for `acc_number`. Otherwise, display an error message.
3 (47)	Check that `acc_holder` exists in the Bank System.			If `acc_holder` exists, prompt user for `acc_number`. Otherwise, display an error message. Prompt user for `acc_holder` again.
4 (48)	For administrators, check that `acc_number` corresponds to `acc_holder`.			If they match, then continue. Otherwise, display an error message. Prompt user for

	For standard users, check that the `acc_number` corresponds to the currently logged in `acc_holder`.			`acc_number` again.
5 (49)	Check that the account is enabled/active.			If the account is enabled/active, then continue. Otherwise, display an error message. Withdrawal gets aborted.
6 (50)	Check that the account was not created in this same day.			If the account is was not created in this same day, then the user is prompted for `amout_wit.` Otherwise, display an error message. Withdrawal gets aborted.
7 (51)	Check that `amount_wit` is valid. - Must be in the form of #####.00. No dollar (or any other) signs. - No decimals (pennies) - Multiple of 5 - Only positive numerical values *bigger* than 0. - No funds that were deposited in this same day.			If `amount_wit` is valid, then continue. Otherwise, display an error message. Prompt user for `acc_wit` again.
8 (52)	Check that the total amount to be withdrawn is valid. - `amount_wit` + fee <= balance			If the total is valid, then the amount specified is withdrawn from the account and the user is informed of this. Save the information provided to the bank account transaction file. Otherwise, display an error message. Prompt user for `amount_wit` again.

Test Case Name: Transfer

Description: Transfer money between two accounts.

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**		EXPECTED OUTPUT
1 (53)	Check that user is logged in.	`login` `admin`	`login` `standard`	If user is logged in, continue.

		`transfer` **`acc_holder`** **`acc_num_f`** **`acc_num_t`** **`amount_tra`** `logout`	**`transfer`** **`acc_num_f`** **`acc_num_t`** **`amount_tra`** `logout`	Otherwise, display an error message.
2 (54)	Check if account mode (`admin` or `standard`) is valid.			If account mode is `admin`, prompt user for `acc_holder`. if account mode is standard, skip `acc_holder` and prompt user for `acc_number`. Otherwise, display an error message.
3 (55)	Check that `acc_holder` exists in the Bank System.			If `acc_holder` exists, prompt user for `acc_num_f`. Otherwise, display an error message. Prompt user for `acc_holder` again.
4 (56)	For administrators, check that `acc_num_f` corresponds to `acc_holder`. For standard users, check that the `acc_num_f` corresponds to the currently logged in `acc_holder`.			If they match, then the user is prompted for `acc_num_t`. Otherwise, display an error message. Prompt user for `acc_num_f` again.
5 (57)	Check that `acc_num_t` exists in the Bank System.			If `acc_num_t` exists, then continue. Otherwise, display an error message. Prompt user for `acc_num_t` again.
6 (58)	Check that the current account is enabled/active.			If the current account is enabled/active, continue. Otherwise, display an error message. Transfer gets aborted.
7 (59)	Check that the current account was not created in this same day.			If the current account was not created in this same day, then continue. Otherwise, display an error message. Transfer gets aborted.
8 (60)	Check that the destination account is enabled/active.			If the destination account is enabled/active, then continue. Otherwise, display an error message. Transfer gets aborted.

9 (61)	Check that the destination account was not created in this same day.			If it was not created in this same day, then prompt the user for `amount_tra`. Otherwise, display an error message. Transaction gets aborted.
10 (62)	Check that `amount_tra` is valid. - Must be in the form of #####.##. No dollar (or any other) signs. - No more than 1000.00 for current session (for standard users only) 1000.00 <= total transferred in this session + `amount_tra` - Only positive numerical values *bigger* than 0. - No funds that were deposited in this same day			If `amount_tra` is valid, then continue. Otherwise, display an error message. Prompt user for `amount_tra` again.
11 (63)	Check that the total amount to be transferred is valid. - `amount_tra` + fee <= balance			If the total is valid, then the amount specified is transferred from `acc_num_f` into `acc_num_t` and the user is informed of this. Save the information provided to the bank account transaction file. Otherwise, display an error message. Prompt user for `amount_tra` again.

Test Case Name: Paybill

Description: Pay bill with money from an account.

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**		EXPECTED OUTPUT

1
(64)
	Check that user is logged in.	`login`
		`admin`
		`paybill`
		`acc_holder`
		`acc_num`
		`company`
		`amount_pay`
		`logout`
			`login`
			`standard`
			`paybill`
			`acc_num`
			`company`
			`amount_pay`
			`logout`
				If user is logged in, continue.
				Otherwise, display an error message.

2 (65)	Check if account mode (`admin` or `standard`) is valid.			If account mode is `admin`, prompt user for `acc_holder`. if account mode is standard, skip `acc_holder` and prompt user for `acc_number`.

				Otherwise, display an error message.
3 (66)	Check that `acc_holder` exists in the Bank System.			If `acc_holder` exists, prompt user for `acc_num`. Otherwise, display an error message. Prompt user for `acc_holder` again.
4 (67)	For administrators, check that `acc_num` corresponds to `acc_holder`. For standard users, check that the `acc_num` corresponds to the currently logged in `acc_holder`.			If they match, then continue. Otherwise, display an error message. Prompt user for `acc_num` again.
5 (68)	Check that the account is enabled/active			If the account is enabled/active, then continue. Otherwise, display an error message. Bill payment gets aborted.
6 (69)	Check that the account was not created in this same day.			If the account was not created in this same day, then prompt the user for `company`. Otherwise, display an error message. Bill payment gets aborted.

7
(70)
	Check that `company` is valid. Must be one of the following:
	`- The Bright Light Electric Company (EC)`
	`- Credit Card Company Q (CQ)`
	`- Low Definition TV, In. (TV)`
	May also use the two-letter abbreviations.
				If `company` is valid, prompt the user for `amount_pay`.
				Otherwise, display an error message. Prompt user for `company` again.

8 (71)	Check that `amount_pay` is valid. - No more than 2000.00 (for standard users only, per day per company) 2000.00 <= total paid			If `amount_pay` is valid, then continue. Otherwise, display an error message. Prompt user for `amount_pay` again.

	in this same day + amount_pay - Only positive numerical values - No funds that were deposited this same day.			
9 (72)	Check that the total amount to be paid is valid. - amount_pay + fee <= balance			If the total is valid, then the amount specified is paid to the corresponding company and the user is informed of this. Save the information provided to the bank account transaction file. Otherwise, display an error message. Prompt user for amount_pay again.

Test Case Name: Help
Description: Displays all available commands with their corresponding descriptions and arguments

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**	EXPECTED OUTPUT
1 (73)	Check if command is help	**help**	If command is help, display command details. If it is not an existing command, show error message. User may use the help command.

1 (1)
Test Case Name: Empty Strings
Description: Some users may input empty strings at any point during program execution.
Expected Output: No error message. However, the prompt to which the empty string was inputted to gets displayed again.

1 (1)
Test Case Name: Cancel
Description: A user may choose to cancel a transaction at any point during the process.

Expected Output: The transaction gets aborted. If fees were to be deducted, they are returned to the corresponding account.