CSCI 3060U - Software Quality and Assurance

Project Phase III - Front-End Requirements Testing(Failure Log List)

Wenbo Zhang, Xuan Zheng, Neel samirkumar Shah, Dev rajivkumar Thaker

Failure Tests List

Test number	What Was Tested	The nature of the failure	Error in Code	Fix Applied
Login01-07	Test login functions	The output mismatch the expected output by just a single “.”	One more “.” in the test code	Delete “.” showing in the code
Withdrawal06 - 07	Test withdraw function with standard user logged in, correct username and correct account number	Withdrawal function for standard users always does not work.	Only wrote code for admin users, missing else branch	Added else branch to handle standard user function
transfer06	Test transferring funds with zero amount (invalid input).	The program displayed “the transfer amount is missing…” instead of “Transfer amount must be greater than zero.”	Our code did not consider missing_input_check(...) and that a zero amount would also be marked as "missing", thus triggering the error.	We achieved this by adding code to check the integrity of the input file in banking_system(...) directly.
transfer10	Test transferring funds with invalid characters in the amount.	The program displayed “Invalid or missing transfer amount.” instead of “Invalid transfer amount. Amount must be	In our code function invalid_character_check() is run after the program has accepted the amount of value.	Adjusted the apply position of invalid_character_check() to check the input data before assigning the amount when the

		numeric.”		program is run.
transfer11	Test with missing input fields.	The program displayed an incorrect error about zero or numeric amounts instead of recognizing the receiver was missing.	The code performed zero_amount_check() or numeric checks before verifying if user2 was missing. Also, we used missing_input_check() incorrectly only after a zero check.	We avoided this by re-prioritizing our inspections to prioritize defect inspections.
transfer01-12	Test whole transfer feature	The transaction output still outputs the user's transaction information even if the transaction fails.	We did not connect the newly written transaction output function with the original process_transfer() judgment mechanism at this stage.	We link it with the original process_transfer() judgment mechanism, and no transaction log output will be recorded when any check fails.
paybill06	est paying a bill with zero amount (invalid input).	The program displayed “the transfer amount is missing…” instead of “Transfer amount must be greater than zero.”	Our code did not consider missing_input_check(...) and that a zero amount would also be marked as "missing", thus triggering the error.	We achieved this by adding code to check the integrity of the input file in banking_system(...) directly.
paybill08	Test paying a bill with missing or incomplete input.	The program gave an error “Payment amount must be greater than zero.” when it should have said “the paybill	The code performed zero_amount_check() or numeric checks before verifying if user2 was missing. Also, we used	We avoided this by re-prioritizing our inspections to prioritize defect inspections.

		amount is missing.”	missing_input_check() incorrectly only after a zero check.	
deposit01	Tests functionality of depositing money into valid(existing) accounts	The program gave an error “Error: Account number and holder name do not match.” When it should have said Deposit successful. Funds unavailable for this session.	There was an issue in the code where it was comparing the account_number given to the account_holder_name which lead to this discrepancy.	Code was fixed
deposit04	Verifies that deposit amount doesn’t exceed account balance limit	The program didn’t throw any error	The code was not testing this condition if the balance limit is exceeding during the deposit	Code was fixed
create03	Checks the functionality that initial balance is not blank	The program threw invalid initial balance error instead of initial balance cannot be blank	The program failed to recognize the blank character for balance, causing it to mistakenly interpret the next command (logout) as input, leading to the issue.	The issue was resolved by ensuring the program correctly handles blank balance values, preventing it from misinterpreting subsequent commands like logout.
create07	Checks the functionality that account holder name is not blank	The program threw invalid initial balance error instead of account holder	The program failed to recognize the blank character for account	The issue was resolved by ensuring the program correctly handles

		name cannot be blank	holder name, causing it to mistakenly interpret the next command (logout) as input, leading to the issue.	blank balance values, preventing it from misinterpreting subsequent commands like logout.
delete05	Checks the functionality that account number is not blank	The program threw invalid account number logout instead of account holder name cannot be blank	The program failed to recognize the blank character for account number, causing it to mistakenly interpret the next command (logout) as input, leading to the issue.	The issue was resolved by ensuring the program correctly handles blank balance values, preventing it from misinterpreting subsequent commands like logout.
delete06	Checks the functionality that account holder name is not blank	The program threw invalid account name logout instead of account holder name cannot be blank	The program failed to recognize the blank character for account holder name, causing it to mistakenly interpret the next command (logout) as input, leading to the issue.	The issue was resolved by ensuring the program correctly handles blank balance values, preventing it from misinterpreting subsequent commands like logout.
changeplan01	Toggle plan from SP to NP on a valid active account.	Error message: “The account number does not match the account holder name” even though it matched.	The code had leftover input() calls and was consuming the wrong tokens from the input file, causing a mismatch.	Removed interactive prompts; read tokens directly from the commands list. Also added a return status (1 or 0) to process_change

				plan() and only log output on success.
changeplan04	Attempt to change plan with an invalid account number (e.g., 99999).	No error message appeared in the .out file, but it printed to the console.	Used print() instead of write_console, so the error never got captured in the .out file.	Added a write_console callback parameter in ChangePlan; replaced all print() calls with self.write_console().
changeplan08	Change plan, then immediately do a deposit as admin in the same session.	The deposit line was correct, but an extra “Enter account number:” prompt appeared.	The deposit code was echoing the account number again, leading to duplicated console lines.	Removed the extra write_console(f"Enter account number...") in the deposit branch.
disable01	Disable a valid active account as admin.	Output line was correct, but the user was never told “account disabled” in the .out file.	process_disable() used print() instead of write_console.	Added a write_console callback in Disable; replaced all print() calls with self.write_console(). Also added a return status so we only log the transaction if process_disable() succeeds.
disable04	Disable using an invalid account number (not in the system).	The transaction was still logged even though it failed.	No status check was done; code always returned success.	Updated Disable.process_disable() to return 0 on failure and 1 on success; in main.py, we log the transaction only if return value == 1.
disable06	Verify no transactions are	The code still allowed further	The account’s availability was	Inserted checks in each

	accepted after disabling an account.	transactions for that user in the same session.	set to “D,” but we never checked availability again for subsequent commands.	transaction to call availability_check() before processing.
logout01	Logout after a standard user session.	The message “Logout successful” printed to console but not the .out file.	Used print("Logout successful.") instead of the write_console() callback.	Added a write_console parameter in Logout; replaced print() with self.write_console().
logout03	Attempt multiple logouts in the same session.	No error was shown for a second logout attempt.	process_logout() did not track whether logout was already performed.	Added a self.processed flag in Logout; if processed is already True, we call write_console("Error: Logout has already been performed...") and return False.
logout05	Logout while not logged in.	No error was displayed; the code proceeded to create a “00...” line in .etf.	The code did not check if logged_in was False.	Added a check: if not self.logged_in, call write_console("Error: You are not logged in.") and return False.