|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Failure report | | | | |
| No. test | Test for | Defect | Modification | Comment |
| Login test | | | | |
| T1.0 | Valid login input should be accepted by program in ATM mode | In the front end greeting, the console prints ("Welcome to the front end: Sucessfully login.”) | ("Welcome to the front end: \nSuccessfully login.”) | Add “\n” after font end greeting. (separate line)  Typo fix: “Successfully”. |
| T1.0 | Valid login input should be accepted by program in ATM mode | In select mode step, the console prints (“Select mode to enter: Sucessfully entered atm mode.”) | (“Select mode to enter: \nSuccessfully entered ATM mode.”) | Add “\n” after select mode message. (separate line)  Typo fix: “Successfully”.  Change to Upper Case: “ATM” |
| T1.0 | Valid login input should be accepted by program in ATM mode | In the atm greeting, the console prints (“Helo, welcome to the ATM mode, please type the transaction you want to make: [EOF]”) | (“Hello, welcome to the ATM mode, please type the transaction you want to make:”) | Delete Print EOF in atm.py file. Typo fix: “Hello” |
| T1.2 | Test if and only if user input unprivileged transaction is accepted in ATM mode | In the atm greeting, the console prints (“Hello, welcome to the ATM mode, please type the transaction you want to make: Enter your account please:”) | (“Hello, welcome to the ATM mode, please type the transaction you want to make: \nEnter your account please:”) | Add “\n” after atm greeting. (separate line) |
| T1.5 | Test if and only if user input unprivileged transaction is accepted in ATM mode | After invalid command input in atm mode (“The operation is not supported in ATM mode” “==”\*34) | After invalid command input in atm mode (“The operation is not supported in ATM mode”) | Delete print(“=”\*34) after error message |
| T1.6 | Test if and only if user input unprivileged transaction is accepted in ATM mode | After invalid command input in agent mode (“The operation is not supported in agent mode” “==”\*34) | After invalid command input in agent mode (“The operation is not supported in agent mode”) | Delete print(“=”\*34) after error message |
| T3.0 | Assure mistyping login command is not accepted and shows error message | Error: EOF when reading a line.  Never login logout. There is an error of reading input line, after all the input fails. | Try:  frontEnd code  Except:  quit | Add catch exception to front end. If an error occurs, kills the program immediately. |
| T3.2(new added test case) | Test if logout command is accepted in choose mode step. | Cannot logout in select mode step:  If choice ==”atm”:  code  Elif choice == “agent”:  Code  Else:  error | If choice ==”atm”:  code  Elif choice == “agent”:  Code  Elif choice == “logout”:  status = True  f = open(transactionSummary, "w") f.writelines("EOS")  f.close()  Else:  Error | Add elif == “logout” in choose mode step, add EOS to output file directory after logout. |
| Logout test | | | | |
| T1.3 | Assure logout command is not accepted after login | Unable to input command after logout. Program ends when logout. | If choice ==”atm”:  code  Elif choice == “agent”:  Code  Elif choice == “logout”:  Code  Else:  Error  print("Successfully logout.")  login(accountListFile, transactionSummary) | Enable login after logout,  add logout message. |
| Createaccount test | | | | |
| T1.0 | check that createaccount is accepted when creating a new account with the different account number from all  other current numbers | user\_input = ["login", "agent", "createaccount", "1000601", "TheCreatedAccount01", "logout"] | user\_input = ["login", "agent", "createacct", "1000601", "TheCreatedAccount01", "logout"] | Change input from “createaccount” to “createacct” to match the command of interface |
| T1.0 | check that createaccount is accepted when creating a new account with the different account number from all  other current numbers | expected\_terminal\_output = ['Welcome to the front end: ', 'Successfully login.', 'Select mode to enter: ', 'Successfully entered agent mode.', 'Hello, welcome to the agent mode, please type the transaction you want to make: ', 'Enter the new account number you want to create please: ', 'Enter the account name for the new account: ', '<-- Your transaction has been successfully made! -->', '==' \* 34, 'Hello, welcome to the agent mode, please type the transaction you want to make: ', 'Successfully logout.', 'Welcome to the front end: '] | expected\_terminal\_output = ['Welcome to the front end: ', 'Successfully login.', 'Select mode to enter: ', 'Successfully entered agent mode.', 'Hello, welcome to the agent mode, please type the transaction you want to make: ', 'Enter the new account number you want to create please: '+'Enter the account name for the new account: '+'<-- Your transaction has been successfully made! -->', '==' \* 34, 'Hello, welcome to the agent mode, please type the transaction you want to make: ', 'Successfully logout.', 'Welcome to the front end: '] | Use ‘+’ replace ‘,’ between “Enter the new account number you want to create please:”, “Enter the account name for the new account:”, and “ <-- Your transaction has been successfully made! -->” in Expected Output to match the real output |
| T1.0 | check that createaccount is accepted when creating a new account with the different account number from all  other current numbers | user\_input = ["login", "agent", "createaccount", "1000601", "TheCreatedAccount01", "logout"] | user\_input = ["login", "agent", "createaccount", "1000666", "TheCreatedAccount01", "logout"] | Change the input account number from “1000601” to “1000666” to match the new account number in output file |
| T1.1 | check that createaccount is not accepted when creating a new account with the same account number from the current account number | user\_input = ["login", "agent", "createacct", "1000600", "TheCreatedAccount01", "logout"] | user\_input = ["login", "agent", "createacct", "1000600", "logout"] | Remove the unnecessary input |
| T1.2 | check that createaccount is not accepted when creating a new account with the account number which is more than  seven decimal digits | user\_input = ["login", "agent", "createacct", "100060100000", "TheCreatedAccount01", "logout"] | user\_input = ["login", "agent", "createacct", "100060100000", "logout"] | Remove the unnecessary input |
| T1.3 | check that createaccount is not accepted when creating a new account with the account number which  is less than seven decimal digits | user\_input = ["login", "agent", "createacct", "1000", "TheCreatedAccount01", "logout"] | user\_input = ["login", "agent", "createacct", "1000", "logout"] | Remove the unnecessary input |
| T1.4 | check that createaccount is not accepted when creating a new account with the account number which beginning with  0 | user\_input = ["login", "agent", "createacct", "0000601", "TheCreatedAccount01", "logout"] | user\_input = ["login", "agent", "createacct", "0000601", "logout"] | Remove the unnecessary input |
| T1.5 | check that createaccount is not accepted when creating a new account with the account number which includes the letters | user\_input = ["login", "agent", "createacct", "1aaaaa1", "TheCreatedAccount01", "logout"] | user\_input = ["login", "agent", "createacct", "1aaaaa1", "logout"] | Remove the unnecessary input |
| Deleteaccount test | | | | |
| T1.1 | check that deleteacct is not accepted when deleting an account with nonexistent account number | user\_input = ["login", "agent", "deleteacct", "9000600", "TestAccount01", "logout"] | user\_input = ["login", "agent", "deleteacct", "9000600", "logout"] | Remove the unnecessary input |
| T1.2 | check that deleteacct is not accepted when deleting an account with invalid account name | user\_input = ["login", "agent", "deleteacct", "1000600", "TestAccount99", "logout"] | user\_input = ["login", "agent", "deleteacct", "1000600", " TestAccount01 ", "logout"] | Change input from “TestAccount99” to “ TestAccount01 ” because the front end does not connect to the Master Accounts File which means it can not check if the account name is already exist or not |
| T1.3 | check that deleteacct is not accepted when the account number which is more than seven decimal digits | user\_input = ["login", "agent", "deleteacct", "100060100000", "TheCreatedAccount01", "logout"] | user\_input = ["login", "agent", "deleteacct", "100060100000", "logout"] | Remove the unnecessary input |
| T1.4 | check that deleteacct is not accepted when the account number which  is less than  seven decimal digits | user\_input = ["login", "agent", "deleteacct", "1000", "TheCreatedAccount01", "logout"] | user\_input = ["login", "agent", "deleteacct", "1000", "logout"] | Remove the unnecessary input |
| T1.5 | check that deleteacct is not accepted when the account number which beginning with  0 | user\_input = ["login", "agent", "deleteacct", "0000601", "TheCreatedAccount01", "logout"] | user\_input = ["login", "agent", "deleteacct", "0000601", "logout"] | Remove the unnecessary input |
| T1.6 | check that createaccount is not accepted when the account number which includes the letters | user\_input = ["login", "agent", "deleteacct", "1aaaaa1", "TheCreatedAccount01", "logout"] | user\_input = ["login", "agent", "deleteacct", "1aaaaa1", "logout"] | Remove the unnecessary input |
| T1.7 | check that other transactions such as deposit is not accepted to an account when the account is deleted | user\_input = ["login", "agent", "deleteacct", "1000600", "TestAccount01", "Deposit", "logout"] | user\_input = ["login", "agent", "deleteacct", "1000600", "TestAccount01", "deposit", "1000600", "logout"] | Change input from “Deposit” to “deposit” to match the command of interface and add the necessary input of account number “1000600” for the failed case with other transactions to a deleted account |
| Deposit test | | | | |
| Test1.3 | Invalid account cannot be deposited | In function functionDeposit  ToAccount = input(" -->") | ToAccount = input("--> ") | Add a space after >, makes the format matching |
| Test2.2 | Invalid amount cannot be deposited | In function verfiyAmount  Print(“Invalid amount, please try agian!”) | Print((“Invalid amount, please try again!”) | Type fixed to match the expected output |
| Test2.8(new added) | Each deposit should within $2,000 | In function  VerifyDepositAmount print("The limit of each depoist is $2,000, please retype: ", end= “”) | print("The limit of each depoist is $2,000, please retype: ") | Change it to default setting to match the format. |
| Test2.9(new added) | Daily deposit limit is $5,000 | In function verfiyDepositAmount print("you have used ${}, please try again."  .format(self.depositRecord.get(account))) | print(f"you have used ${self.depositRecord  .get(account):.2f}, please try again.") | Use the new format method in python, makes thing looks better and matching the output. |
| Withdraw test | | | | |
| Test1.0 | Valid withdraw request should be accepted | The user input : "login", "atm", "withdraw", "1000327", "10000000", "logout" | "login", "atm", "withdraw", "1000327", "100000", "logout" | 10000000 exceed the limit, cannot be used as successful withdraw case |
| Test1.0 | Valid withdraw request should be accepted | The expected output file:  DWR 0000000 100000 1000327 \*\*\* | WDR 0000000 100000 1000327 \*\*\* | Type fixed |
| Test1.1 | Valid withdraw request should be accepted | "login", "agent", "withdraw", "1000327", "10000000", "logout" | "login", "agent", "withdraw", "1000327", "100000", "logout" | The limit of each withdraw should be $1000.00. So the origin one didn’t work |
| Test1.1 | Valid withdraw request should be accepted | The expected output file:  DWR 0000000 100000 1000327 \*\*\* | WDR 0000000 100000 1000327 \*\*\* | Typo fixed |
| Test2.8 (new added) | The limit of each withdraw is $1,000 | In function verifyWithdrawAmount  Print(“The limit of each withdraw is $1,000, please retype: ”, end=“”) | Print(“The limit of each withdraw is $1,000, please retype: ”) | Chang it to default format to match the output |
| Test2.9(new added) | The daily limit of withdraw is $5,000 | In function verifyWithdrawAmount  Print("you have withdrawn ${}, please try again."  .format(self.withdrawRecord.get(account))) | f"you have withdrawn ${self.withdrawRecord.  get(account):.2f}, please try again." | Use the new format method in python, makes thing looks better and matching the output. |
| Transfer test | | | | |
| Test3.8(new added) | The limit of each transfer is $10,000 | In function verifyTransferAmount  Print("The limit of each transfer is $10,000, please retype: ", end=“”) | Print("The limit of each transfer is $10,000, please retype: ") | Chang it to default format to match the output |
| Test3.9(new added) | The daily limit of transfer is $10,000 | In function verifyTransferAmount print("you have transferred ${}, please try again."  .format(self.transfer  Record.get(fromAccount))) | print(f"you have transferred ${self.transferRecord  .get(fromAccount):.2f}, please try again.") | Use the new format method in python, makes thing looks better and matching the output. |