Organization

The tests are saved in `/testing/tests/{transaction}/{test name}`. Where {transaction} is the name of the transaction (e.g. withdrawal, or paybill}, and {test name} is the name of the test (e.g. StandardWithdrawal1). Each test folder has 3 files: input.txt, output.txt, and transaction_log.txt. input.txt is the user input, output.txt is the expected terminal output, and transaction_log.txt is the expected log file to be generated.

Test names follow the general format of {privilege level}{transaction}{id}, so a withdrawal from a non admin account would have a privilege level of “standard”, a transaction of “withdrawal”, and a somewhat arbitrary ID of “1” in this example. As a result the test name would be “StandardWithdrawal1”. ID is chosen based on the test creation order, as all tests are important, and ranking in terms of importance would be subjective. Additionally, some test names may deviate in the case of there being no difference between tests whether it was run in standard or admin. An example of this is with logout. Because logout for standard and admin users is the same, one test will suffice for both, so a test name of “Logout1” is used. Another addendum is that on transactions that require elevated permissions, the privilege level is not included, as all tests would require the elevated privilege, apart from tests testing that the elevated permissions are required.

Only 1 copy of the bank accounts file will be used for all tests, kept in `/testing/tests`. More information on how this will be used is in the Running the Tests section.

Running the Tests

Tests will be run using bash scripts. The master bank accounts file will be copied into each test directory for each test, to make sure that if the file is modified for any reason, it will not affect the rest of the tests.

The scripts will recursively search through all directories, and when it finds a directory with input.txt, output.txt, and transaction_log.txt it will: create a copy of the master bank accounts file, launch an instance of the front-end, enter the user input in input.txt and compare the output to output.txt and the generated transaction log against transaction_log.txt.

The scripts will then generate a report based on the results of each test. More information on the format of the reporting is in the Reporting section

This approach should ensure that no tests will impact future tests, giving each test a consistent environment, and avoiding failures resulting from that. Additionally, it means that new tests will be tested automatically, without needing to modify the scripts.

Reporting

The report will be generated in the /testing/tests directory with the name scheme of “test report” {timestamp}.txt. Timestamp is the current time Unix epoch. An example file name would be “test report 1738587267.txt”. This ensures that old reports are not overwritten, and the exact time a report was generated is known, regardless of timezone.

The report will follow a format of

```
{path}  {pass}
```

in the case of a passing test. In the case of a failing test it will follow the format of

```
{path}  {fail}  {input}  {output}  {expected output}
```

Where {path} is the path to the test, excluding /testing/tests (e.g. withdrawal/StandardWithdrawal1). {pass/fail} is pass or fail, depending on whether the test passed or failed. {input} is only seen in the case of the console output not matching the expected output, and shows the line expected to have caused the erroneous output. {output} is the output given by the program, this is either console output, or a transaction log. {expected output} is the output in the test case.

The report would follow a format similar to this:

```
Path            pass/fail   input       outputexpected output
create/Create1  pass
create/Create2  fail        99999.99    account creation failed
account created
create/Create2  fail                    05_JOHN DOE
_00000_09999999_        00_ADMIN              _00000_00000000_
create/Create2  fail                          00_ADMIN
_00000_00000000_
create/Create3  pass
create/Create4  pass
```

Note: Proper formatting is not apparent in this document. Actual reports will not be constricted by line length, so each pass/fail will be contained to exactly 1 line and will have proper indentation.

Pass/Fail is determined based on whether the lines match exactly. Non-matching lines will be deemed a failure and logged as such. This includes empty lines, as a result any transactions that fail would likely generate several fails, as the lines in the log file would

be out of sync, and error on each of them. In the event a test passes, only a single line is written, stating that the test passed.

Assumptions

- Users can bypass any caps by logging out and logging back in – transactions are not timestamped
- Transaction Logs which account has money being transferred out using “EX” in the miscellaneous portion of the transaction log
- Transactions logs which account has money being transferred in using “IN” in the miscellaneous portion of the transaction log
- “EX” transfers will be logged before “IN” transfers
- You cannot transfer negative amounts / transfer money out of another person’s account into one of your own accounts
- Admin logout is logged with an account name of “ADMIN”
- Duplicate account numbers are possible, as long as there is a different account name for each account
- Account creation also requests an account number, not just an account name