Test Document

Team PB-PI

April 8, 2018

Table 1: Team

Name	ID Number
Alissa Bellerose	27377320
Sabrina D'Mello	27739486
Melanie Damilig	40032420
Tobi Decary-Larocque	27407645
Zain Farookhi	26390684
Giulia Gaudio	27191766
Jason Kalec	40009464
Damian Kazior	40016168
Johnny Mak	40002140
Philip Michael	40004861
Ramez Nicolas Nahas	26718108
Steven Tucci	40006014
Shunyu Wang	40043915

1 Introduction

The introduction of the document provides an overview of the entire document, briefly introducing what are its goals, and what information is to be found in it.

2 Test Plan

Describe what forms of testing you plan to do (unit, subsystem, integration), describe briefly the schedule and resources for testing, and how you designed your test cases.

Indicate which qualities (from requirements) were tested and which qualities were not tested.

2.1 System Level Test Cases

Deposit Test Case

Purpose
If the amount provided by the user is valid (a positive number), the deposit money functionality should add the amount indicated by the user to the account (database) and increment the Current Balance field by the amount indicated. If the user provided an invalid amount, such as a string or a negative number, the system should display an error message and not update the account (database) and Current Balance. The purpose of the Deposit Test Case is the following: verify that the deposit money functionality works as described above.

Input Specification (see Figure 1 and Figure 2)

Expected Output
State the expected system response and output. You can cross-reference to actual file data specified in an appendix.

Traces to Use Cases
The Deposit Money use case and the Display Balance use case are tested with this test case. The requirements being tested:

- The user should be able to deposit money into the selected account.
- The system should display error messages when the user tries to perform an invalid operation.
- The user interface should be updated appropriately.
- The account information should be updated appropriately.

Withdraw Test Case

Purpose
If the amount provided by the user is valid (a number), the withdraw money functionality should deduct the amount indicated by the user from the account and decreases the Current Balance field by the amount indicated. If the user provided an invalid amount, such as a string or a negative number, the system should display an error message and not update the account and Current Balance will not be updated. Type of Withdraw and Transaction Description fields are optional and could take any string, including the empty string as input. The purpose of the Withdraw Test Case is the following: verify that the withdraw money functionality works as described above.

Input Specification

Expected Output
State the expected system response and output. You can cross-reference to actual file data specified in an appendix.

Traces to Use Cases
The Withdraw Money use case and the Display Balance use case are tested with this test case. The requirements being tested:

- The user should be able to withdraw money from the selected account.
- The system should display error messages when the user tries to perform an invalid operation.
- The user interface should be updated appropriately.
- The account information should be updated appropriately.

2.2 Subsystem Level Test Cases

2.3 Unit Test Cases

Unit Test for Deposit Money Use Case

UpdateModel(DepositMoneyViewData data)
Test will be conducted on the UpdateModel method in the DepositMoneyController class. The parameter DepositMoneyViewData is a class and contains the following fields:

- The amount to be deposited - float amount
- The type of transaction - String type
- The description of the transaction - String transactionReason

- The date of transaction - Date date

When the UpdateModel is called, taking DepositMoneyViewData as a parameter, it creates a new entry (row) in the database table deposit_money (new entries are always added at the top of the database table). Therefore, the last transaction is always at the top of the table.

Unit Test for Deposit Money Use Case

UpdateModel(DepositMoneyViewData data)
Tests will be conducted on UpdateModel method in WithdrawMoneyController Class. Unlike UpdateView, we could only conduct a few unit tests on it since it is a black box and coders cant not sense whether the model data is updated on the database. So, we have to manually input a series of test data, and verifies whether the real results we read on database are aligned with expected updates. The parameter WithdrawMoneyViewData is a class that contains four field as follows:

- The amount to be withdrawn float amount
- The type of transaction String type
- The Description of Transaction String transactionReason
- The date of transaction Date date

When we call UpdateModel taking WithdrawMoneyViewData, it will create a new table row as the first table row, then we read from the first table row and test whether it is the same as the inputs.

3 Test Results

In this section we will be outlining the test results for both of our main test cases while providing in depth information regarding regular, special, and boundary cases.

3.1 Deposit Money Test

Expected Output (when Done button is clicked, and tests are implemented in order)

Regular Cases

- Value 100: Works as expected.
  Account: 100.00 Current Balance: 100.00

- Value 200.78: Works as expected.
  Account: 300.78 Current Balance: 300.78

Special Cases

- Value -50: Does not work as expected. Subtracts 50 from the account and from the Current Balance field.
  No error message is displayed.
- Value Hello: Works as expected. An error message appears.
  Error: Value entered must be a positive number.
- Value: 3.2222: Works as expected.
  Account: 304.0022 Current Balance: 304.00
- Value -30.55: Subtracts 30.55 from the account and from the Current Balance field.
  No error message is displayed.

Type of Deposit input text field

Boundary Cases

- Pass nothing (leave the field empty). Works as expected. The deposit concludes with default reason.

Regular Cases

- Pass a string Cash Entry. Works as expected and provides deposit reason.
- Pass a string containing a number Money Transfer #3. Works as expected and provides deposit reason.

(optional) Transaction Description input text field

Boundary Cases

- Pass nothing (leave the field empty). Works as expected with no description.

Regular Cases

- Pass a string Money found in couch. Works as expected and provides description.
- Pass a string containing a number Sold Xbox 360. Works as expected and provides description.

Done button

All tests above satisfy this case as they require the use of the Done button. Please refer to sections 1 and 2.

Cancel button

Regular Cases

- Transaction is not recorded. That is:
  - Account (database) does not contain a record corresponding to the input. Works as expected.
  - Current Balance field is unchanged. Works as expected.

3.2 Withdraw Money Test

Expected Output (when Done button is clicked, and tests are implemented in order)

Boundary Cases

- Passing a 0 or a 0.00: Works as expected, no actual change is done to the balance.

Regular Cases

- Value of 100: Works as expected, withdrawal is performed.
- Value of 200.78: Works as expected. Withdrawal is performed.

Special Cases

- Value of -50: Does not work as expected. It adds 50 to the account and to the Current Balance field. No error message is shown.
- Value Hello: Works as expected. An error message is shown.
- Value 3.2222: Works as expected by being rounded to the nearest hundredth when performing the withdrawal.

Expected Output (when Show History is clicked).

Boundary Cases

- is under the TYPE OF WITHDRAW: Works as expected. It is left blank.

Regular Cases

- bill is under the TYPE OF WITHDRAW: Works as expected. Bill is shown as the type of the withdrawal.
- check is under the TYPE OF WITHDRAW: Works as expected. Check is shown as the type of the withdrawal.

(optional) Transaction Description input text field.

Boundary Cases

- is under the DESCRIPTION: Works as expected. It is left blank.

Regular Cases

- Pay electricity bill of June is under the DESCRIPTION: Works as expected. Shows this as the description for the withdrawal.
- Pay off the debt owed to Jack is under the DESCRIPTION: Works as expected. Shows this as the description for the withdrawal.

Done button

Expected output is relative to the outlined uses in the above cases. They all require the use of the Done button, therefore this button is tested through those same tests.

Cancel button

Regular Cases

- Transaction is not recorded. That is:
  - Account (database) does not contain a record corresponding to input data: Works as expected.
  - Current Balance field is unchanged: Works as expected. No changes occur.

4 References

We obtained a test document sample that we used as a reference: Montrealopoly, Master Test Plan, from: https://users.encs.concordia.ca/ paquet/wiki/images/3/35/Phase3final.pdf

A Description of Input Files

A.1 Mymoneyappdb.db

File Description

Database file used for SQLite. Contains all the database information locally, such as the different tables and their contents. It contains the following four Tables:

- Withdraw_Money
- Deposit_money
- Display_Balance
- sqlite_sequence

Input Description

The application will read from this file every time it is run to get all the Database information and display them on the GUI.

A.2 Transaction_History.csv

File Description

Excel file used to keep a history of all transactions executed within the application, as well as all the information relating to those transactions. It contains the following six columns:

- Date
- Transaction Type
- Description
- Amount
- Type of Withdrawal
- Type of Deposit

Output Description

The application will read from this file every time it tries to access the history of all transactions to be able to display them on the GUI.

B Description of Output Files

B.1 Mymoneyappdb.db

File Description

Database file used for SQLite. Contains all the database information locally, such as the different tables and their contents. It contains the following four Tables:

- Withdraw_Money
- Deposit_money
- Display_Balance
- sqlite_sequence

Output Description

Everytime a new withdraw or deposit action happens, the application will write to this file to add new entries under their respective tables.

B.2 Transaction_History.csv

File Description

Everytime a new withdraw or deposit action happens, the application will write to this file to add new entries under their respective tables.

- Date
- Transaction Type
- Description
- Amount
- Type of Withdrawal
- Type of Deposit

Output Description

Everytime a new withdraw or deposit action happens, the application will write to this file to add new rows containing all the information of the transaction.