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

Contents

1 Introduction

This document contains the overall test plan for the myMoney application. The myMoney application is a tool for young adults and students to keep track of their spending and help with their budgeting. This test plan describes the different tests performed to ensure a working program and explains the reasoning for the tests.

The purpose of this document is to provide assurance and show the reliability of the software. The test plan therefore includes:

- An overall view of the different tests performed
- A description of the tests
- References to the use cases and requirements satisfied by testing

Testing the program's different functionalities involves different types of tests. Unit, integration and user interface testing were performed for the purpose of this document. This was managed via black-box, white-box and boundary testing.

2 Test Plan

For the deposit and withdraw use cases, different unit tests were performed (including boundary tests) to ensure functionality. The different test cases were designed to test each part of each use case. This includes the format of what the user inputs, as well as ensuring that the database is updated or not, according to whether the transaction is legal or not.

In regards to our requirements document, the test plan references the deposit and withdrawal use cases specifically. The tests for each of these use cases are integral to the proper functionality of the application and are therefore an important part of the testing sequence. The balance display use case does not have any functional testing to ensure that it works correctly, but any modification to the user's balance should be reflected through the appropriate view. Similarly, the use cases for the "Show History" functionality as well as "Export History" do not rely on any user input except for a button click, so testing was only performed to make sure the output files are correct.

There were no tests performed to check if the application is portable to all operating systems described in the non-functional requirements, since the application is not being exported as a stand-alone app. It will function within the Eclipse environment for the time being.

2.1 System Level Test Cases

2.1.1 [TC-1] [Withdraw Money]

Table 2: TC-1.1 : Testing the Amount input

Test Case Number	TC-1.1
Description	The user enters the amount to withdraw in the Amount input text field.
Input	1. Click Withdraw Money button 2. Enter the amount as a positive number in Amount input text field environment. 3. Click Done button
Expected Output	1. The system records the number from input as Amount in a row of transaction history table. 2. New balance will be updated after deducting the input number from the current Account Balance.
Expected Post-Condition	Account balance will be presented as a negative value with two decimal places if it belows 0. Error message is displayed if the input is not valid, and Current Balance and database will not be updated.
Execution History	[04/14/2018—Shunyu Wang] Executed test successfully.
Trace to Use Case	UC-1

Table 3: TC-1.2 : Testing the type of withdraw input

Test Case Number	TC-1.2
Description	Takes the type of withdraw defined by the user.
Input	1. Click Withdraw Money button 2. Enter any positive number in Amount input text field. 3. Enter any type of withdraw defined by the user, such as Bill, Check, or empty string 4. Click Done button
Expected Output	1. The system records the number from input as Amount and the type of withdraw as Type of Withdraw in a row of transaction history table. 2. New balance will be updated after deducting the input number from the current Account Balance.
Expected Post-Condition	The same string will be taken from input as the output for Type of Withdraw in transaction history table. Empty string is allowed.
Execution History	[04/14/2018—Shunyu Wang] Executed test successfully.
Trace to Use Case	UC-1

Table 4: TC-1.3 : Testing the transaction description input

Test Case Number	TC-1.3
Description	Take any user-defined descriptions, such as purposes for transactions and other special notes.
Input	1. Click Withdraw Money button 2. Enter any positive number in Amount input text field 3. Enter any user-defined description in Transaction Description text field, such as Pay electricity bill of June, Pay off the debt owed to Jack , or empty string 4. Click Done button
Expected Output	1. The system records the number from input as Amount and the user-defined description as Transaction Description in a row of transaction history table. 2. New balance will be updated after deducting the input number from the current Account Balance.
Expected Post-Condition	The same string will be taken from input as the output for Transaction Description in transaction history table. Empty string is allowed.
Execution History	[04/14/2018—Shunyu Wang] Executed test successfully.
Trace to Use Case	UC-1

Table 5: TC-1.4 : Testing the Done UI Button

Test Case Number	TC-1.4
Description	When clicked, the UI and the database are updated according to the values entered by the user. In some cases, error messages are displayed.
Input	1. Click Withdraw Money button 2. Refer to TC-1.1, TC-1.2, and TC-1.3, input testing data appropriately 3. Click Done button
Expected Output	1. The system records the number from input as Amount, the type of withdraw as Type of Withdraw, and the user-defined description as Transaction Description in a row of transaction history table. 2. New balance will be updated after deducting the input number from the current Account Balance.
Expected Post-Condition	Account balance will be presented as a negative value with two decimal places if it falls belows 0. All testing data should be recorded together in database as a row transaction history, which can be examined by clicking Show History
Execution History	[04/14/2018—Shunyu Wang] Executed test successfully.
Trace to Use Case	UC-1

Table 6: TC-1.5 : Testing the Cancel UI Button

Test Case Number	TC-1.5
Description	When clicked, the transaction is canceled: the Account Balance and database are not updated.
Input	1. Click Withdraw Money button 2. Input any testing data into text fields 3. Click Cancel button
Expected Output	1. Data are cleared and all input text fields are collapsed.
Expected Post-Condition	The system state doesnt change, meaning Current Balance is not changed and no change in database.
Execution History	[04/14/2018—Shunyu Wang] Executed test successfully.
Trace to Use Case	UC-1

2.1.2 [TC-2] [Deposit Money]

Table 7: TC-2.1 : Testing the amount input

Test Case Number	TC-2.1
Description	The user enters the amount to deposit in the Amount input text field.
Input	1. Click Deposit Money button 2. Enter the amount as a positive number in Amount input text field. 3. Click Done button
Expected Output	1. The system records the number from input as Amount in a row of transaction history table. 2. New balance will be updated after adding the input number to the current Account Balance.
Expected Post-Condition	Account balance will be presented as a positive value with two decimal places. Error message is displayed if the input is not valid, and Current Balance and database will not be updated.
Execution History	[04/14/2018—Shunyu Wang] Executed test successfully.
Trace to Use Case	UC-2

Table 8: TC-2.2 : Testing the type of deposit

Test Case Number	TC-2.2
Description	Takes the type of deposit defined by the user.
Input	1. Click Deposit Money button 2. Enter the amount as a positive number in Amount input text field. 3. Enter any type of deposit defined by the user, such as Direct Deposit, Cash, or empty string 4. Click Done button
Expected Output	1. The system records the number from input as Amount and the type of deposit as Type of Deposit in a row of transaction history table. 2. New balance will be updated after adding the input number to the current Account Balance.
Expected Post-Condition	The same string will be taken from input as the output for Type of Deposit in transaction history table. Empty string is allowed.
Execution History	[04/14/2018—Shunyu Wang] Executed test successfully.
Trace to Use Case	UC-2

Table 9: TC-2.3 : Testing the transaction description input

Test Case Number	TC-2.3
Description	Take any user-defined descriptions, such as purposes for transactions and other special notes.
Input	1. Click Deposit Money button 2. Enter the amount as a positive number in Amount input text field. 3. Enter any user-defined description in Transaction Description text field, such as Bi-weekly pay, Receive the bonus for working overtime, or empty string. 4. Click Done button
Expected Output	1. The system records the number from input as Amount and the user-defined description as Transaction Description in a row of transaction history table 2. New balance will be updated after adding the input number to the current Account Balance
Expected Post-Condition	The same string will be taken from input as the output for Transaction Description in transaction history table. Empty string is allowed.
Execution History	[04/14/2018—Shunyu Wang] Executed test successfully.
Trace to Use Case	UC-2

Table 10: TC-2.4 : Testing the Done UI Button

Test Case Number	TC-2.4
Description	When clicked, the UI and the database are updated according to the values entered by the user. In some cases, error messages are displayed.
Input	1. Click Deposit Money button 2. Refer to TC-2.1, TC-2.2, and TC-2.3, input testing data appropriately. string. 3. Click Done button
Expected Output	1. The system records the number from input as Amount, the type of deposit as Type of Deposit, and the user-defined description as Transaction Description in a row of transaction history table. 2. New balance will be updated after adding the input number to current Account Balance.
Expected Post-Condition	Account balance will be presented as a negative value with two decimal places if it falls belows 0. All testing data should be recorded together in database as a row transaction history, which can be examined by clicking Show History
Execution History	[04/14/2018—Shunyu Wang] Executed test successfully.
Trace to Use Case	UC-2

Table 11: TC-2.5 : Testing the Cancel UI Button

Test Case Number	TC-2.5
Description	When clicked, the transaction is canceled: the UI and database are not updated.
Input	1. Click Deposit Money button 2. Input any testing data into text fields 3. Click Cancel button
Expected Output	1. All data are cleared and all input text fields are collapsed.
Expected Post-Condition	The system state doesnt change, meaning Current Balance is not changed and no change in database
Execution History	[04/14/2018—Shunyu Wang] Executed test successfully.
Trace to Use Case	UC-2

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

3.1.1 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.

3.1.2 "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.

3.1.3 "(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.

3.1.4 "Done" button

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

3.1.5 “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

3.2.1 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.

3.2.2 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.

3.2.3 (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.

3.2.4 "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.

3.2.5 "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

A.1.1 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

A.1.2 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

A.2.1 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

A.2.2 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

B.1.1 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

B.1.2 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

B.2.1 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

B.2.2 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.

C Figures

Figure 1: The user opens myMoney App.

Figure 2: The fields mentioned in the Deposit Test Case section are shown above.