

# 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

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 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 3 and Figure 4)

##### 1.1. “Amount” input text field

**Purpose** The user enters the amount to deposit in the “Amount” input text field.

**Test Data** Requirement: This amount should be a positive number (decimal or integer). No negative values or strings are allowed.

#### Boundary Case

- Pass a 0 or 0.00

#### Regular Cases

- Pass a positive integer value: 100
- Pass a positive decimal value: 200.78

#### Special Cases

- Pass a negative number: -50
- Pass a string: “Hello”
- Pass a negative decimal value: -30.55
- Pass a positive decimal value: 3.2222

**Expected Output (when “Done” button is clicked, and tests are implemented in order)**    Boundary Case

- Value 0 or 0.00: The “Current Balance” field and account remain unchanged.  
*Account: 0.00 Current Balance: 0.00*

Regular Cases

- Value 100: The “Current Balance” field and account are incremented by 100. *Account: 100.00 Current Balance: 100.00*
- Value 200.78: The “Current Balance” field and account are incremented by 200.78. *Account: 300.78 Current Balance: 300.78*

Special Cases

- Value -50: An error message is displayed and the “Display Balance” field and account remain unchanged.  
*Error: Value entered must be a positive number*
- Value “Hello”: An error message is displayed and the “Display Balance” field and account remain unchanged.  
*Error: Value entered must be a positive number*
- Value: 3.2222: The “Current Balance” field and account are incremented by 3.2222 but the amount in the “Current Balance” field is rounded to the nearest hundredth.  
*Account: 304.0022 Current Balance: 304.00*
- Value -30.55: An error message is displayed. The “Display Balance” field and account remain unchanged.  
*Error: Value entered must be a positive number*

## 1.2. “Type of Deposit” input text field

**Purpose**    Takes the type of deposit defined by the user.

**Test Data**    Requirement: Any strings, including the empty string. Numbers will be converted to strings

Boundary Case

- Pass nothing (leave the field empty)

Regular Cases

- Pass a string “Cash Entry”
- Pass a string containing a number “Money Transfer #3”

#### Special Cases

- Pass a negative number: -50
- Pass a string: “Hello”
- Pass a negative decimal value: -30.55
- Pass a positive decimal value: 3.2222

**Expected Output (when “Done” button is clicked, and tests are implemented in order)** Boundary Case

- The “TYPE OF DEPOSIT” field is left empty.

#### Regular Cases

- The “TYPE OF DEPOSIT” field corresponding to the transaction shows “Cash Entry”
- The “TYPE OF DEPOSIT” field shows corresponding to the transaction shows “Money Transfer”

### 1.3. “(optional) Transaction Description” input text field

**Purpose** Take any user-defined descriptions, such as purposes for transactions and other special notes.

**Test Data** Requirement: Any strings, including the empty string. Numbers will be converted to strings

#### Boundary Case

- Pass nothing (leave the field empty)

#### Regular Cases

- Pass a string “Money found in couch”
- Pass a string containing a number “Sold Xbox 360”

**Expected Output (when “Done” button is clicked and tests are implemented in order)**    Boundary Case

- The “DESCRIPTION” field is left empty.

Regular Cases

- The “DESCRIPTION” field corresponding to the transaction shows “Money found in couch”
- The “TYPE OF DEPOSIT” field shows corresponding to the transaction shows “Sold Xbox 360”

#### 1.4. “Done” button

**Purpose**    When clicked, the UI and the database are updated according to the values entered by the user. In some cases, error messages are displayed.

**Test Data**    See section 1.1, 1.2, and 1.3.

**Expected Output**    See section 1.1, 1.2, and 1.3.

#### 1.5. “Cancel” button

**Purpose**    When clicked, the transaction is canceled: the UI and database are not updated.

**Test Data**

- “Amount” field contains a valid amount: 50
- “Type of deposit” field contains information: “Cash”
- “(optional) Transaction Description” contains information: “Sold Xbox 360”

**Expected Output**

- Transaction is not recorded. That is:
- Account (database) does not contain a record corresponding to the input test data from section 1.5.2.

- “Current Balance field” is unchanged.

### **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.

## **2.1.2 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**

#### **2.1. “Amount” input text field**

**Purpose** The user enters the amount to deposit in the “Amount” input text field.

**Test Data** Requirement: This amount should be a positive number (decimal or integer). No negative values or strings are allowed.

Boundary Case

- Pass a 0 or 0.00

Regular Cases

- Pass a positive integer value: 100
- Pass a positive decimal value: 200.78

#### Special Cases

- Pass a negative number: -50
- Pass a string: “Hello”
- Pass a negative decimal value: -30.55
- Pass a positive decimal value: 3.2222

#### **Expected Output (when “Done” button is clicked, and tests are implemented in order)**    Boundary Case

- Pass 0 or 0.00: The “Current Balance” field and account remain unchanged.  
*Account: 0.00 Current Balance: 0.00*

#### Regular Cases

- Value 100: The “Current Balance” field and account are incremented by 100. *Account: -100.00 Current Balance: -100.00*
- Value 200.78: The “Current Balance” field and account are incremented by 200.78. *Account: -200.78 Current Balance: -200.78*

#### Special Cases

- Value -50: An error message is displayed and the “Display Balance” field and account remain unchanged.  
*Error: Value entered must be a positive number*
- Value “Hello”: An error message is displayed and the “Display Balance” field and account remain unchanged.  
*Error: Value entered must be a positive number*
- Value: 3.2222: The “Current Balance” field and account are incremented by 3.2222 but the amount in the “Current Balance” field is rounded to the nearest hundredth.  
*Account: -304.0022 Current Balance: -304.00*
- Value -30.55: An error message is displayed. The “Display Balance” field and account remain unchanged.  
*Error: Value entered must be a positive number*



## 2.2. “Type of Withdraw” input text field

**Purpose** Take any user-defined categories of withdraw, such as bill and check.

**Test Data** Requirement: Any strings, including the empty string. Numbers will be converted to strings. Assume all other inputs are valid.

Boundary Case

- Pass nothing (leave the field empty)

Regular Cases

- Pass a string “bill”
- Pass a string “check”

Special Cases

- Pass a negative number: -50
- Pass a string: “Hello”
- Pass a negative decimal value: -30.55
- Pass a positive decimal value: 3.2222

**Expected Output (when “Done” button is clicked, and tests are implemented in order)** Boundary Case

- The “TYPE OF WITHDRAW” field is left empty.

Regular Cases

- “bill” is under the “TYPE OF WITHDRAW”
- “check” is under the “TYPE OF WITHDRAW”

## 2.3. “(optional) Transaction Description” input text field

**Purpose** Take any user-defined descriptions, such as purposes for transactions and other special notes.

**Test Data** Requirement: Any strings, including the empty string. Numbers will be converted to strings. Assume all other inputs are valid

Boundary Case

- Pass nothing (leave the field empty)

Regular Cases

- Pass “Pay electricity bill of June”
- Pass “Pay off the debt owed to Jack”

**Expected Output (when “Done” button is clicked and tests are implemented in order)** Boundary Case

- The “DESCRIPTION” field is left empty.

Regular Cases

- “Pay electricity bill of June” is under the “DESCRIPTION”
- “Pay off the debt owed to Jack” is under the “DESCRIPTION”

## 2.4. “Done” button

**Purpose** When clicked, the UI and the database are updated according to the values entered by the user. In some cases, error messages are displayed.

**Test Data** See section 2.1, 2.2, and 2.3.

**Expected Output** See section 2.1, 2.2, and 2.3.

## 2.5. “Cancel” button

**Purpose** When clicked, the transaction is canceled: the UI and database are not updated.

**Test Data**

- “Amount” field contains a valid amount: 50
- “Type of deposit” field contains information: “Credit Card”
- “(optional) Transaction Description” contains information: “Pay monthly bill to Bell”

## Expected Output

- Transaction is not recorded. That is:
  - Account (database) does not contain a record corresponding to the input test data from the Test Data section of 2.5.
  - “Current Balance field” is unchanged.

## 2.2 Unit Test Cases

### 2.2.1 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.

Tester name:	Alissa				Test date:		18-04-07	
Class Name:	DepositMoneyController				Method:	UpdateModel		
Variable name:	DepositMoneyViewData				Lower Bound:	Empty for String		
	The amount to be deposited- float amount					0 for float		
	The type of transaction - String type				Upper Bound:	None		
	The description of transaction - String							
Test Case 1:	amount = 100, type = "Cash", transactionReason = "Salary", date.ValueOf("1999-12-21")							
Test Case 2:	amount = 3.78, type = "Cash", transactionReason = "Sell", date.ValueOf("1999-12-21")							
Test Case 3:	amount = 20.33467, type = "Cash", transactionReason = "Gift", date.ValueOf("1999-12-21")							
Test Case 4 (boundary):	amount = 0, type = "", transactionReason = "", date.ValueOf("1999-12-21")							
Test Case	1	2	3	4				
Expected Output:	amount = 100, type = "Cash", transaction Reason = "Salary", date = date.Value Of("1999-12-21"	amount = "3.78", type = "Cash", transaction Reason = "Sell", date = date.Value Of("1999-12-21"	amount = "20.33467", type = "Cash", transaction Reason = "Gift", date = date.Value Of("1999-12-21"	amount = "0", type = "", transaction Reason = "", date = date.Value Of("1999-12-21"				
Actual Output:	amount = 100, type = "Cash", transaction Reason = "Salary", date = date.Value Of("1999-12-21"	amount = "3.78", type = "Cash", transaction Reason = "Sell", date = date.Value Of("1999-12-21"	amount = "20.33467", type = "Cash", transaction Reason = "Gift", date = date.Value Of("1999-12-21"	amount = "0", type = "", transaction Reason = "", date = date.Value Of("1999-12-21"				
Bug found?	No	No	No	No				

Figure 1: Unit test for Deposit Money use case

## 2.2.2 Unit Test for Withdraw Money Use Case

### UpdateModel(WithdrawMoneyViewData 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.

Tester Name	Shunyu			Test Date	2018-04-07		
Class Name	WithdrawMoneyController			Method	UpdateModel		
Variable name	WithdrawMoneyViewData			Lower Bound	Empty for String	Upper Bound	None
	o The amount to be withdrawn – float amount				0 for float		
	o The type of transaction – String type						
	o The Description of Transaction – String transactionReason						
	o The date of transaction – Date date						
Test Case 1	amount = 100, type = "Bill", transactionReason = "monthly rent", date = Date.ValueOf("1999-12-21")						
Test Case 2	amount = 3.78, type = "Cash", transactionReason = "grocery", date = Date.ValueOf("1999-12-21")						
Test Case 3	amount = 20.33467, type = "Debit", transactionReason = "food", date = Date.ValueOf("1999-12-21")						
Test Case 4 (boundary)	amount = 0, type = "", transactionReason = "", date = Date.ValueOf("1999-12-21")						
Test Case	1	2	3	4			
Expected Output	amount = 100, type = "Bill", transactionReason = "monthly rent", date = Date.ValueOf("1999-12-21")	amount = 3.78, type = "Cash", transactionReason = "grocery", date = Date.ValueOf("1999-12-21")	amount = 20.33467, type = "Debit", transactionReason = "food", date = Date.ValueOf("1999-12-21")	amount = 0, type = "", transactionReason = "", date = Date.ValueOf("1999-12-21")			
Actual Output	amount = 100, type = "Bill", transactionReason = "monthly rent", date = Date.ValueOf("1999-12-21")	amount = 3.78, type = "Cash", transactionReason = "grocery", date = Date.ValueOf("1999-12-21")	amount = 20.33467, type = "Debit", transactionReason = "food", date = Date.ValueOf("1999-12-21")	amount = 0, type = "", transactionReason = "", date = Date.ValueOf("1999-12-21")			
Bug Found?	No	No	No	No			

Figure 2: Unit test for Withdraw Money use case

## 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.ensc.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

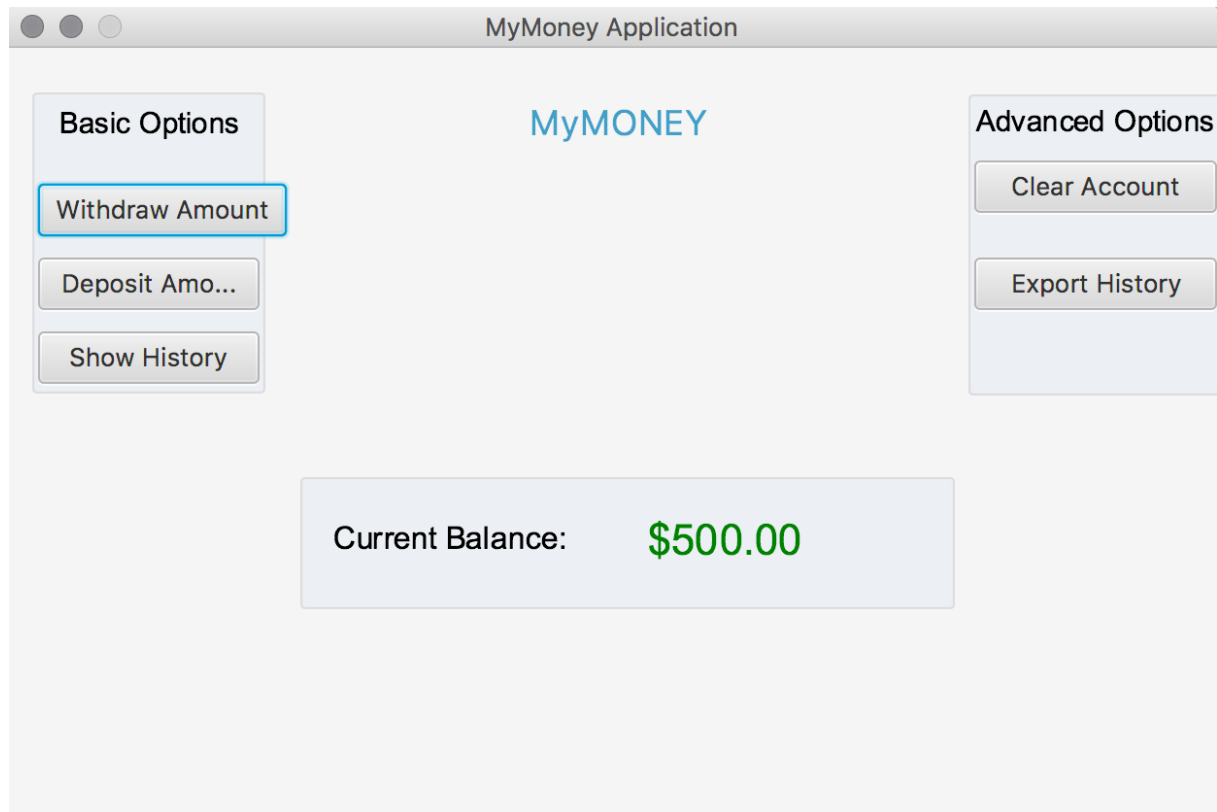
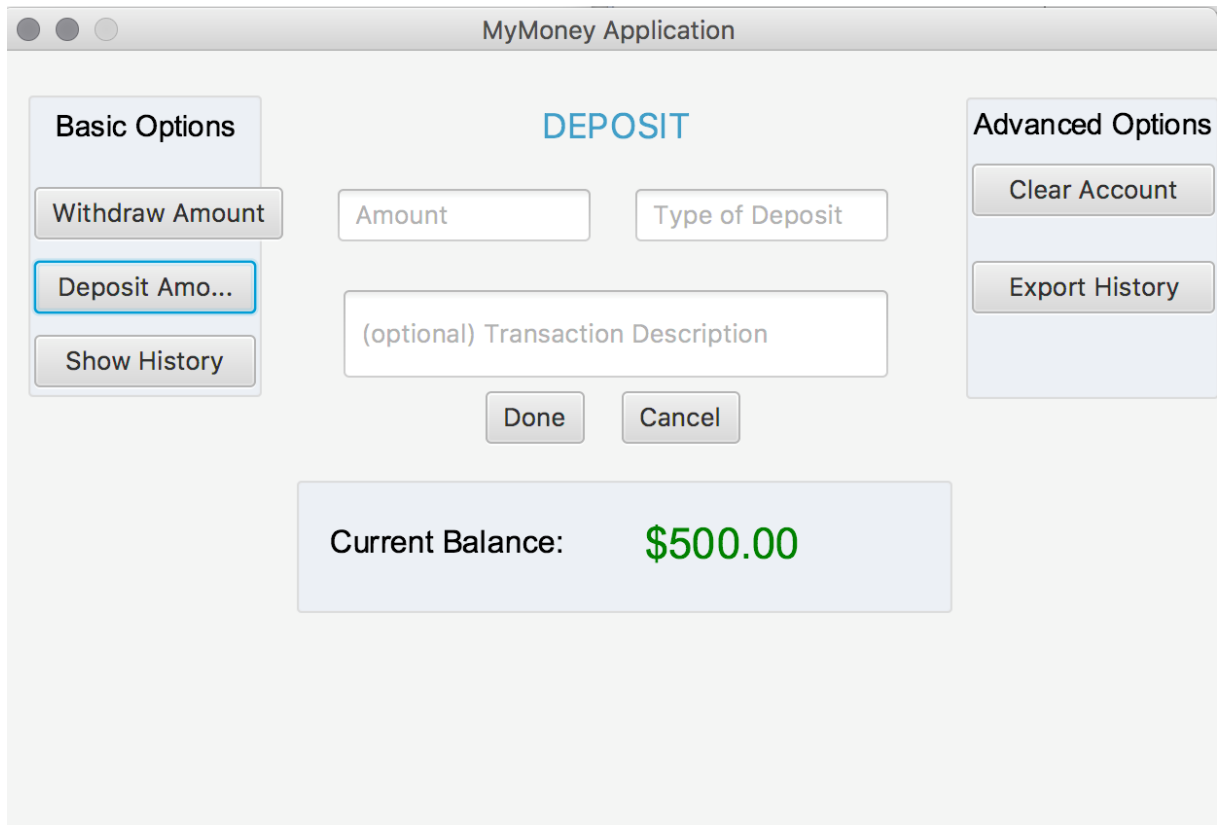


Figure 3: The user opens myMoney App.



The image shows a software window titled "MyMoney Application" containing a "DEPOSIT" dialog box. The dialog is organized into three main sections: "Basic Options" on the left, a central input area, and "Advanced Options" on the right. The "Basic Options" section contains three buttons: "Withdraw Amount", "Deposit Amo..." (which is highlighted with a blue border), and "Show History". The central area features a title "DEPOSIT" in blue, followed by two input fields labeled "Amount" and "Type of Deposit", and a larger text area labeled "(optional) Transaction Description". Below these are "Done" and "Cancel" buttons. The "Advanced Options" section on the right contains two buttons: "Clear Account" and "Export History". At the bottom of the dialog, a light blue box displays the "Current Balance: \$500.00" in green text.

Section	Field/Option	Value/Label
Basic Options	Withdraw Amount	Button
	Deposit Amo...	Button (highlighted)
	Show History	Button
Central Input	Amount	Input Field
	Type of Deposit	Input Field
Advanced Options	Clear Account	Button
	Export History	Button
Current Balance:		\$500.00

Figure 4: The fields mentioned in the Deposit Test Case section are shown above.