Iteration 3 - Test Document

Team PA-PI-a

8 April 2018

Table 1: Team

Name	ID Number
Melanie Taing	40009850
Laurie Gagnon	22943433
Wayne Yiel Leung	26586988
Jordan Rutty	27300107
Michael Foo	40000225
Pierre-Andre Leger	40004010
Colin Greczkowski	40001600

Contents

1 Introduction

The purpose of this document is to gather all information necessary for testing of the My-Money application. This document describes the testing approach and overall framework that will be used to test the MyMoney application.

The following pages will identify the requirements that will be tested, the testing strategy used, the test cases and their results, and the description of input files.

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

All test cases for testing at the system level.

Purpose
State the purpose of the test. Indicate which requirement and which aspect of that requirement is being tested.

Input Specification
State the context for the test in terms of system state. State the input test data. You may need to mention operations invoked as well as data for the operation. You can cross-reference to actual file data specified in an appendix.

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
State which requirements (at the level of use case and scenario) are tested by this test case.

Table 2: Template Test Case

Test Case Number	UT-1
Test Case Description	This test case is used to ensure the generated puzzle board has the same dimensions as the input width and height
Input	1. None - Default 8-8 board size 2. Width/height from "input.txt" file.
Expected Output	1. "OK" - Test executed successfully.
Expected Post-Conditions	The system responds to the presence or absence of the input vector and outputs a success message upon test successful completion in "output.txt", along with a time-stamp containing the test's execution time and date.
Execution History	1. 05/04/2018 — Tester's name — Executed test successfully.

2.2 Subsystem Level Test Cases

All test cases for testing at the subsystem level.
One subsection per subsystem

2.2.1 Subsystem X

2.3 Unit Test cases

All test cases for testing at the unit level.
One subsection per unit

2.3.1 Unit Test Case 1

Table 3: UT-1

Test Case Number	UT-1
Test Case Description	This test case is used to ensure that transactions are properly saved or updated to their repository
Input	1. A Transaction object populated with generic data 2. A second Transaction object with the ID of the first one. 3. A test transaction database.
Expected Output	1. Transaction details are printed to console.
Expected Post-Conditions	A transaction database is created and a transaction is inserted. The balance of this transaction is then updated to a new value.
Execution History	1. 04/03/2018 — Colin Greczkowski — Executed test successfully.

2.3.2 Unit Test Case 2

Table 4: UT-2

Test Case Number	UT-2
Test Case Description	This test verifies that the deleteItem method works as intended, and deletes a Transaction record for a given ID
Input	1. A generic account ID 2. A Transaction object populated with generic data, associated to the generic account. 3. A test transaction database.
Expected Output	1. "Delete Transaction 1"
Expected Post-Conditions	The test transaction database should be empty.
Execution History	1. 04/07/2018 — Colin Greczkowski — Executed test successfully.

2.3.3 Unit Test Case 3

Table 5: UT-3

Test Case Number	UT-3
Test Case Description	This test case is used to make sure all Transactions associated to an account are properly purged from the repository.
Input	1. A generic account ID 2. Two Transaction objects populated with generic data, associated to the generic account. 3. A test transaction database.
Expected Output	1. "Delete Transaction 1" 2. "Delete Transaction 2"
Expected Post-Conditions	The test transaction database does not contain the two transactions that had the generic account ID.
Execution History	1. 04/03/2018 — Colin Greczkowski — Executed test successfully.

2.3.4 Unit Test Case 4

Table 6: UT-4

Test Case Number	UT-4
Test Case Description	This tests the RepositoryContainer's ability to save a variety of types of objects (Transactions, Accounts, Budgets).
Input	1. Test Transaction, Budget and Account Databases 2. A test Transaction 3. A test Account 4. A test Budget
Expected Output	1. The test transaction's details are printed to console.
Expected Post-Conditions	The account,transaction and budget items are saved to their respective test databases. Balances are updated correctly.
Execution History	1. 04/07/2018 — Colin Greczkowski — Executed test successfully.

2.3.5 Unit Test Case 5

Table 7: UT-5

Test Case Number	UT-5
Test Case Description	TO BE COMPLETED: BudgetContainer test
Input	1. A Budget object populated with generic data 2. A test Account database with transactions 3. A test Budget database
Expected Output	1. The test budget's details are printed to console.
Expected Post-Conditions	A budget database is created and a budget is inserted. The recorded budget amount is updated according to transactions made across all accounts for that budget.
Execution History	1. 04/08/2018 — Melanie Taing — Not executed

3 Test Results

List the tests, indicating which passed and which did not pass. List requirements indicating the percentage of tests that passed for that requirement.

4 References

A Description of Input Files

Describe/include test data from input files.