Project: MyMoney App
Iteration 3

April 2018

Name	ID Number
Fabian Vergara	40006707
Jean-Loup	40006259
Vincent Boivin	27419694
Marc-Antoine Jette-Leger	27895038
Michael Xu	27206356
Kisife Giles	40001926

Table 1: Team: PA-PJ

Contents

1 Introduction

This document marks the end of the third and last phase of the three iterations in the COMP 354 group project. In the course of these three iterations, team PA-PJ worked on building the MyMoneyApp which is simple application to provide financial advice to young professionals.

1.1 Purpose

The objective in this phase of the project is to test the application against the requirements (for validity), robustness, and reliability. To this end, a test plan has been made. The plan presents the test categories used, the components, units or subsystems to be tested and the approach used. However, due to time constrains, not all components could be tested. The tested components are presented in section3, while those excluded are indicated in section 4.

2 Outline of Planned Tests

The following tests will be carried out on the MyMoneyApp. Due to time constrains, we will do only the black box testing and exclude white box testing. This means that the test carried out will test for inputs and outputs from the system, but not the code coverage.

2.0.1 Unit Testing

Unit testing tests individual units in the system.Here we test the core functionalists of individual classes for their behaviours. Black box test were carried out on the following methods of the following classes:

- RegistrationTest Class, for user registration.
- LogIn Class, for user login activity.
- Finance Class, for finance activities.
- Advice Class for providing advise to the user.
- BankingInfo Class for bank transactions.
- Display Class for accuracy of information displayed to the user.
- Encryption Class for encryption of user information.

The details of the tested units is presented in section 5.1 below.

2.0.2 Integration Testing

Involves testing how units work together to achieve tasks and responsibilities. Integration testing will be done on the following collaborating classes/units:

- LoginTest Class and BankingInfo Class for collaboration to login and upload the correct user information.
- Registration and Encryption classes.
- Advice class and Display class

2.0.3 Functional Testing

This test's purpose is to ensure the functionality of the software system against any type of input. It is a type of black-box testing, which is a type of testing that disregards the internal program structure and focuses on what the system does, rather than how it is made. To this end, what the system does is first identified then a test to check its functionality and reliability is designed based on the expected behaviour, then fault tolerance(robustness) in the case of invalid input. Functional testing has been run on these requirements:

- Registration and Login
- Data View
- Set Goals
- Data Input Automation
- Advice
- Spending History View
- Dynamic Screen Adjustment
- Savings Feature

2.0.4 User Interface Testing

User interface testing is a testing method that involves finding defects in the software system through the use of the user interface to in turn, ensure that it meets the System Requirements Specifications.
The application provides the following user interfaces which where tested and documented:

- The Main Panel
- Registration Panel
- Bank Panel
- Budgeting Panel
- Results Chart

3 Test Inclusions

This section specifies the core tests done on the application.

3.1 Unit Test

Unit testing is done with black box only, due to time constraints.

3.2 Integration Testing

The individually tested units are combined and tested to evaluate how they work together.

3.3 Functional Testing

Test cases are generated to match individual use cases.

3.4 User Interface Testing

Here the GUI is tested against user interaction.

3.5 Security and Access Control Testing

The application is tested against unauthorized access.

3.6 Load Testing

The application is tested against large input files.

4 Test Exclusions

The following test cases were deemed necessary for the MyMonetApp but will be excluded due to time constraints. White box/ code coverage test is excluded from the unit tests.

4.1 Data and Database Integrity Testing

In this project, simple text files are used in place of a database. Ideally, the application would use a database. In either case, this test would be necessary.

5 Testing Approach

5.1 Unit testing

AdviceTest Class : Test Case 1 : It tests to see if the input is smaller than 4 characters
Input : ""
Expected Output : Error
Output : Error
Bug : No

AdviceTest Class : Test Case 2 : It tests to see if the input is an advice object
Input : Advice object ("-230960.00 2016-02-18 Student Loans Debt")
Expected Output : ""
Output : ""
Bug : No

AdviceTest Class : Test Case 3 : It tests to see if the output is correct
Input : Advice object("-230960.00 2016-02-18 Student Loans Debt")
Expected Output : "You should be paying your debts rather than spending, as you have -230960.0 in debt and 0.0 in savingsHowever, you are accumulating money, as your bring in more than you use"
Output : "You should be paying your debts rather than spending, as you have -230960.0in debt and 0.0 in savings. However, you are accumulating money, as your bring in more than you use"
Bug : No

BankingInfoTest Class : Test Case 1 : It test that the transaction file is opened and not empty
Input : ""
Expected Output : true
Output : true
Bug : No

BankingInfoTest Class : Test Case 2 : It tests that the bank file is opened and not empty
Input : ""
Expected Output : true
Output : true
Bug : No

BankingInfoTest Class : Test Case 3 : It tests to see that the transactions are sorted by dates
Input : Transactions
Expected Output : true
Output : true

Bug : No

BankingInfoTest Class : Test Case 4 : It tests to see if the sorted files are created
Input : Created files
Expected Output : true
Output : true
Bug : No

BankingInfoTest Class : Test Case 5 : It tests to see if the files are saved correctly
Input : Created files
Expected Output : true
Output : true
Bug : No

DisplayTest Class : Test Case 1 : It tests that the panel for the BankPanel class is of size 960x720
Input : A BankPanel object
Expected Output : true
Output : true
Bug : No

DisplayTest Class : Test Case 2 : It tests that the panel for the FinancePanel class is of size 960x720
Input : A FinancePanel object
Expected Output : true
Output : true
Bug : No

DisplayTest Class : Test Case 3 : It tests that the panel for the log in panel is of size 960x720
Input : A MainPanel object
Expected Output : true
Output : true
Bug : No

DisplayTest Class : Test Case 4 : It test that the panel for the registration panel is of size 960x720
Input : A RegistrationPanel object
Expected Output : true
Output : true
Bug : No

EncryptionTest Class : Test Case 1 : It tests that a string is properly encrypted

Input : Unencrypted string
Expected Output : Encrypted string
Output : Encrypted string
Bug : No

EncryptionTest Class : Test Case 2 : It tests that a string is properly encrypted and decrypted
Input : Encrypted string
Expected Output : Decrypted string
Output : Decrypted string
Bug : No

FinanceTest Class : Test Case 1 : It tests that the value homeBudget is set to 1
Input : homeBudget
Expected Output : 1
Output : 0
Bug : Yes

FinanceTest Class : Test Case 2 : It tests that the value homeSpending is set to 0
Input : homeSpending
Expected Output : 0
Output : 0
Bug : No

FinanceTest Class : Test Case 3 : It tests that the value of text1 is set to ””
Input : text1
Expected Output : ””
Output : ””
Bug : No

LogInTest Class : Test Case 1 : It tests for the size of the list containing user information
Input : Size of the list
Expected Output : Size of the list
Output : Size of the list
Bug : No

LogInTest Class : Test Case 2 : It tests that the size of the list is greater than 0
Input : Size of the list
Expected Output : true
Output : true
Bug : No

LogInTest Class : Test Case 3 : It tests to see if the input is contained in the user information list
Input : test data
Expected Output : "Failure validating the user information!"
Output : "Failure validating the user information!"
Bug : No

5.2 Integration Testing

Test Attribute	Test Specifications
Test ID	INTG-1
Test Name	Login and BankInfo Integration
Test Suite ID	INTG
Test preconditions	An existing user account and bank transactions
Test Steps	Case 1: Log in to user's account
Test post conditions	—
Test Case	Expected Results
Case 1	Bank transactions uploaded should belong to corresponding user account
Case 2	Registration panel should still be displayed
Actual Results	—
Case 1	Corresponding bank transactions were loaded.
Test Results	—
Failed Tests	None
Passed Tests	Case 1

Table 2: Integration Test 1

Test Attribute	Test Specifications
Test ID	INTG-1
Test Name	Registration and Encryption
Test Suite ID	INTG
Test preconditions	Username and password should not match an existing account in the system.
Test Steps	Case 1: Create a new account and upload bank transactions.
Test post conditions	—
Test Case	Expected Results
Case 1	User information and bank transactions should be encrypted after registration.
Actual Results	—
Case 1	User login information and bank transactions were encrypted.
Test Results	—
Failed Tests	None
Passed Tests	Case 1

Table 3: Integration Test 2

5.3 Functional Testing

These tests aim to ensure the functionality and presence of the requirements that have been put forth in the SRS document.

5.3.1 Registration Functional tests

Test Attribute	Test Specifications
Test ID	FUNC-1
Test Name	Username should be unique
Test Suite ID	FUNC
Test preconditions	Username must not exist beforehand
Test Steps	Case 1: Click on registration
	Case 2: Input username that exists already
	Case 3: Input a password
	Case 4: Click register account button
Test post conditions	—
Test Case	Expected Results
Case 1	Invalid username warning should appear
Case 2	Registration panel should still be displayed
Actual Results	—
Case 1	Invalid username warning appears
Case 2	Registration panel still be displayed
Test Results	—
Failed Tests	None
Passed Tests	Case 1 case 2

Table 4: Registration functional test 1

Test Attribute	Test Specifications
Test ID	FUNC-2
Test Name	Username, Password are mandatory
Test Suite ID	FUNC
Test preconditions	The user cannot sign up without providing a username and a password
Test Steps	Case 1: Click on registration
	Case 2: Leave everything blank
	Case 3: click on Register Account
Test post conditions	—
Test Case	Expected Results
Case 1	Please enter username, password warning should appear
Case 2	Registration panel should still be displayed
Actual Results	—
Case 1	please enter username,password warning appears
Case 2	Registration panel still displayed
Test Results	—
Failed Tests	None
Passed Tests	Case 1 case 2

Table 5: Registration Functional Test 2

Test Attribute	Test Specifications
Test ID	FUNC-3
Test Name	Account is successfully created
Test Suite ID	FUNC
Test preconditions	—
Test Steps	Case 1: Click on registration
	Case 2: Enter an unused username
	Case 3: click on Register Account
Test post conditions	—
Test Case	Expected Results
Case 1	Registration page should close
Case 2	User is back to the login page
Actual Results	—
Case 1	Registration page closes
Case 2	User is back to the login page
Test Results	—
Failed Tests	None
Passed Tests	Case 1 case 2

Table 6: Registration Functional Test 3

5.3.2 Login Functional tests

Test Attribute	Test Specifications
Test ID	FUNC-4
Test Name	User logs in properly
Test Suite ID	FUNC
Test preconditions	User has an account registered in the system
Test Steps	Case 1**: Input username and password in the appropriate boxes
	Case 2**: Click on the login button
Test post conditions	—
Test Case**:	Expected Results
Case 1**:	Login page should disappear
Case 2:	User is at the main bank panel
Actual Results	—
Case 1	Login page disappears
Case 2	User is at the bank panel page
Test Results	—
Failed Tests	None
Passed Tests	Case 1, Case 2

Table 7: Login Functional Test 1

Content[1]

Test Attribute	Test Specifications
Test ID	FUNC-5
Test Name	Username/Password pair invalid/nonexistent
Test Suite ID	FUNC
Test preconditions	—
Test Steps	Case 1: Input username and password in the appropriate boxes
	Case 2: Click on the login button
Test post conditions	—
Test Case	Expected Results
Case 1**:	Invalid username/password should appear
Case 2**:	User is still at the login page
Actual Results	—
Case 1**:	invalid username/password appears
Case 2	User is at the login page
Test Results	—
Failed Tests	None
Passed Tests	Case 1, Case 2

Table 8: Login Functional Test 2

Content[2]

5.3.3 Data View

Test Attribute	Test Specifications
Test ID	FUNC-6
Test Name	Data view
Test Suite ID	FUNC
Test preconditions	User must have a valid pair of username/password
Test Steps	Case 1: Input username and password in the appropriate boxes
	Case 2: Click on the login button
Test post conditions	User is able to view his financial data
Test Case	Expected Results
Case 1	User logs in successfully
Case 2	User is at the main panel, with the data ready to be viewed
Actual Results	—
Case 1	User logs in successfully
Case 2	User is at the main panel, with the data ready to be viewed
Test Results	—
Failed Tests	None
Passed Tests	Case 1, Case 2

Table 9: Transactions View Functional Test

5.3.4 Set Goals

Test Attribute	Test Specifications
Test ID	FUNC-7
Test Name	Data view
Test Suite ID	FUNC
Test preconditions	User has logged into a pre-existing account and is now on the main bank panel
Test Steps	Case 1: Click on budget
	Case 2: Input the amount for each category
	Case 3: Click on add to budget
Test post conditions	User should see his budget increased and can have access to a chart
Test Case	Expected Results
Case 1	User gets to the budget page
Case 2	User inputs the required information
Case 3	User has access to a chart of his spending
Actual Results	—
Case 1	User gets to the budget page
Case 2	User inputs the required information
Case 3	User has access to a chart of his spendings
Test Results	—
Failed Tests	None
Passed Tests	Case 1, Case 2,Case 3

Table 10: Set Goals Functional Test

5.3.5 Data Input Automation

Test Attribute	Test Specifications
Test ID	FUNC-8
Test Name	Data view
Test Suite ID	FUNC
Test preconditions	User has logged into a pre-existing account and is now on the main bank panel
Test Steps	Case 1: The user's financial spending is displayed
Test post conditions	User is able to see his spending
Test Case	Expected Results
Case 1	User is able to see his spending
Actual Results	—
Case 1	User is able to see his spending
Test Results	—
Failed Tests	None
Passed Tests	Case 1

Table 11: Data Input Automation Functional Test

5.3.6 Advice Generation

Test Attribute	Test Specifications
Test ID	FUNC-9
Test Name	Advice generation based on input
Test Suite ID	FUNC
Test preconditions	User has logged into a pre-existing account and is now on the main bank panel
Test Steps	Case 1: The user's advice on his spending is displayed
Test post conditions	User is able to see his advice
Test Case	Expected Results
Case 1	User is able to see his advice
Actual Results	—
Case 1	User is able to see his advice
Test Results	—
Failed Tests	None
Passed Tests	Case 1

Table 12: Advice Generation Functional Test

5.3.7 Spending History View

Test Attribute	Test Specifications
Test ID	FUNC-10
Test Name	Able to sort the spending based on time
Test Suite ID	FUNC
Test preconditions	User has logged into a pre-existing account and is now on the main bank panel
Test Steps	Case 1: Click on date
Test post conditions	The spending history has been sorted by date
Test Case	Expected Results
Case 1	Spending history is sorted by date
Actual Results	—
Case 1	Spending history is sorted by date
Test Results	—
Failed Tests	None
Passed Tests	Case 1

Table 13: Spending History View Functional Test

5.3.8 Dynamic Screen Adjustment Test

This test case tests the use case 5; the application should dynamically adjust the screen settings to suit the host device settings.

Test Attribute	Test Specifications
Test ID	FUNC-10
Test Name	Dynamic change of screen size
Test Suite ID	FUNC
Test preconditions	User has logged into a pre-existing account and is now on the main bank panel
Test Steps	Case 1: Click on date
Test post conditions	The screen size adjusts to fit the information
Test Case	Expected Results
Case 1	The screen size adjusts to fit the information
Actual Results	—
Case 1	The screen size adjusts to fit the information
Test Results	—
Failed Tests	None
Passed Tests	Case 1

Table 14: Dynamic Screen Adjustment Functional Test

5.3.9 Savings Advice Functional Test

Test Attribute	Test Specifications
Test ID	FUNC-10
Test Name	Saving information is given(separate from regular advice)
Test Suite ID	FUNC
Test preconditions	User has logged into a pre-existing account and is now on the main bank panel
Test Steps	Case 1: Saving advice is displayed in the middle column
Test post conditions	Saving advice is displayed in the middle column
Test Case	Expected Results
Case 1	Saving advice is displayed in the middle column
Actual Results	—
Case 1	Saving advice is displayed in the middle column
Test Results	—
Failed Tests	None
Passed Tests	Case 1

Table 15: Savings Advice Functional Test

5.4 User Interface Testing

Each user interface was tested for functionality against user interactions. The test for each interface is presented in tabular form below.

5.4.1 Main panel interface Test

Test Attribute	Test Specifications
Test ID	GUI-1
Test Name	Main Panel Test
Test Suite ID	GUI
Test preconditions	Panel must be displayed
Test Steps	Case 1*: Click to close panel
Case 2*	Click panel window to maximize
Case 3*	Click panel window to minimize
Case 4	Registration panel should be displayed
Case 5	Login activity should be initiated
Case 6	Return to main panel
Test post conditions	—
Test Case	Expected Results
Case 1*	Panel should close
Case 2*	Panel window can be maximized
Case 3*	Panel window can be minimized
Case 4	Registration panel should be displayed
Case 5	Login activity should be initiated
Case 6	Return to main panel
Actual Results	—
Case 1*	Panel window closed
Case 2*	Panel window maximized
Case 3*	Panel window minimized
Case 4	Registration panel displayed
Case 5	Login activity initiated
Case 6	Return to main panel
Test Results	—
Failed Tests	None
Passed Tests	Case 1 case 2 case 3 case 4 case 5 case 6

Table 16: GUI Test: Main Panel Interface

Content[3]

[3]* Signifies tests common to all panels

5.4.2 Registration Interface Test

Test Attribute	Test Specifications
Test ID	GUI-2
Tester	Coder 1
Test Name	Registration Panel Test
Test Suite ID	GUI
Test preconditions	Panel must be displayed
Test Steps	Case 1: Click to close panel
	Case 2: User name field
	Case 3: Confirm password field
	Case 4: Registration button
	Case 5: Click cancel button
Test post conditions	—
Test Case	Expected Results
Case 1	Panel should close
Case 2	User name field should be displayed
Case 3	Password field should be displayed
Case 4	Confirm password field should be displayed
Case 5	Registration activity
Actual Results	—
Case 1	Panel close as expected
Case 2	User name field displayed
Case 3	Password field displayed
Case 4	Confirm password field displayed
Case 5	Registration activity initiated
Test Results	—
Failed Tests	None
Passed Tests	Case 1 case 2 case 3 case 4 case 5 case 6
Remarks	A remark if necessary

Table 17: GUI Test: Registration Panel Interface

Content[4]

[4]* Cases in main panel interface tested.

5.4.3 Bank Panel Interface Test

Test Attribute	Test Specifications
Test ID	GUI-3
Tester	Coder 2
Test Name	Bank Panel Test
Test Suite ID	GUI
Test preconditions	Panel must be displayed
Test Steps	Case 1: Advice text area should displayed
	Case 2: Savings text area should displayed
	Case 3: Transactions text area should displayed
	Case 4: Click on Raw button
	Case 5: Click date button
	Case 6: Click exit button
	Case 7: Click budget button
Test post conditions	—
Test Case	Expected Results
Case 1	Advice should be displayed in advice text area
Case 2	Savings should be displayed on savings area should displayed
Case 3	Transactions should be displayed on text area should displayed
Case 4	Listener event should be initiated for raw data
Case 5	Listener event should be initiated and date displayed
Case 6	Application should exit
Case 7	User should be taken to budget panel
Actual Results	—
Case 1	Advice is displayed in advice text area
Case 2	Savings are displayed on savings area should displayed
Case 3	Transactions are displayed on text area should displayed
Case 4	raw date data is presented
Case 5	Transactions are displayed by date
Case 6	Application terminates
Case 7	User is taken to budget panel
Test Results	—
Failed Tests	None
Passed Tests	Case 1 case 2 case 3 case 4 case 5 case 6 case 7

Table 18: GUI Test: Bank Panel Interface

Content[5]

5.4.4 Budget Panel Interface Test

This interface should enable a user to set a budget.

Test Attribute	Test Specifications
Test ID	GUI-4
Tester	Coder 1
Test Name	Budget Panel Test
Test Suite ID	GUI
Test preconditions	Panel must be displayed
Test Steps	Case 1: Budget text area should displayed
	Case 2: Categories drop-down window should displayed
	Case 3: Budget entry fields should displayed
	Case 4: Click on "Set" button
	Case 5: Click "Cancel" button
	Case 6: Click exit button
	Case 7: Click the pie chart button
Test post conditions	—
Test Case	Expected Results
Case 1	Budget setting area should be displayed in advice text area
Case 2	Categories drop-down window should be displayed
Case 3	Budget entry fields should be displayed
Case 4	Listener event should be initiated for budget activity
Case 5	Listener event should be initiated and user returned to main panel
Case 6	Application should exit
Case 7	User should be taken to pie chart panel
Actual Results	—
Case 1	Budget area displayed as expected
Case 2	Budget categories presented as expected
Case 3	Budget entry fields displayed
Case 4	Budget activity started
Case 5	Transactions are displayed by date
Case 6	Application terminates
Case 7	User is taken to pie chart panel
Test Results	—
Failed Tests	None
Passed Tests	Case 1 case 2 case 3 case 4 case 5 case 6 case 7

Table 19: GUI Test: Budget Panel Interface

Content[6]

5.5 Security and Access Control Testing

Test Attribute	Test Specifications
Test ID	SEC-1
Test Name	Security Test
Test Suite ID	Sec
Test preconditions	—
Test Steps	Case 1: Click on register
Test Steps	Case 2: Input username/password
Test post conditions	—
Test Case	Expected Results
Case 1	All data is encrypted
Actual Results	—
Case 1	Data is encrypted datafile \login_info
Test Results	—
Failed Tests	None
Passed Tests	Case 1

Table 20: Security and access Test Content[7]

5.6 Load Testing

Loading large files fails in that the Bank Bank Panel GUI can not accommodate all of the information.
MyMoneyApp algorithm handles different loads of different magnitudes but fails to display everything correctly. This test puts in evidence a cosmetic flaw of the system that needs to be addressed.

Remark: A scrollable GUI could solve this problem.

6 Description of Input Files

MyMoneyApp receives data stream from bank institutions in the form of text files with extension .txt. There is one text file for every bank data stream that will be provided from MyMoneyApp users' banks. However, for this iteration and to run most of the tests, the development team only uses one input file.
This input file is extremely important as most of MyMoneyApp functionality depends on such. Without this file, no useful information is displayed to a given user, and the user is informed that a data file is needed to be provided in order to enjoy MyMoneyApp features.

Transactions_data.txt corresponds to the input file name and its respective extension. This input file also benefits from MyMoneyApp encryption. In there, the format is as follow:

amount date company expenseType
-1000000 2018-04-08 MyMoneyAppTeam Business

The input file uses white spaces as delimiter and new line characters as different transaction input (line). MyMoneyApp is designed to ignore white spaces at the beginning of each transaction input within the file input and white spaces at the end before a new line character is found.

Other input files are generated as the user registers and logins for the first time. However, those input files are independent of the user and are used as private persistent data storage used within the app. Those other input files are text files (.txt extension) as well and are named as follow: **user_not_found.txt** and **login_info.txt**

user_not_found.txt is independent of the user and is only used as a string input to show an error to the user as he enters an incorrect login information. The user is prompted to create an account or enter his correct credentials. This input file is used to change the message MyMoneyApp will show to the user on wrong logging without needing intervention of the development team.

login_info.txt is independent of the user as well. This persistent data file is used to store login information of the user. login_info.txt benefits from MyMoneyApp encryption to ensure that user login data is privately secured from unscrupulous hands.

7 Description of Output Files

MyMoneyApp generates text files from the **transactions_data.txt** input file. The output files generated by the application reflect its internal data manipulation algorithm.
The MyMoneyApp algorithm analyzes **transactions_data.txt** line by line and models each transaction into a transaction java object. Each transaction object is then stored in a hashtable, where its key is its date, and each key corresponds to a list of transactions that are added as many transactions might have the same date. Then the algorithm is able to organize transactions by date (where it happens to be a key on a hashtable of transactions lists). Finally, the algorithm exports each list with its respective key as a file, where the file name is its key (date) and the content of the list is the content of its respective file. This is achieve to use those files a cache and manipulate them easily as the user is able to change values.

The user is able to test this feature as those files are publicly available. However, internally, MyMoneyApp development team ensures functionality by running unit tests of this algorithm as follows: Insert a new transaction file with one transaction, let the algorithm do its processing, then check if there exists a file with the respective transaction date and its file content matches the new transaction file line input (which is a transaction by itself as well).
Users are able to test this by providing their transaction file and seeing if its corresponding transaction file by date is generated. Both files are stored in the same root folder. The user is not able to perform any steps in between, as the algorithm is fully automated and is not expecting any interaction with the user by any means.

8 References

Roger S Pressman, Software Engineering: A Practitioner's Approach, 7th edition, McGraw-Hill

List of Tables