COMP 354
Test Document for the project myMoney

Team PA-PK

April 2, 2018

Table 1: Team

Name	ID Number
Anne-Laure Ehresmann	27858906
Marc-Antoine Dube	40029307
Kadeem Caines	26343600
Abdel Rahman Jawhar	27192142
Keith Dion	40036340
Hrachya Hakobyan	40041555
Andrew-Smith	40034936
Dongyu Chen	27241909
Yauheni Karaniuk	40005680
Renny Xu	40005262
Wei Wang	40041116

Table 2: Revision history

Version	Date	Changes
1.0	15 March 2018	Completed test document

Contents

List of Figures

List of Tables

1 Introduction

The aim of this document is to ensure that a coherent and accurate testing strategy is used by each member of the testing team. It seeks to test the implementation of the system described in the Design Plan, testing its validity, robustness, and reliableness as a software, as well as ensuring that the requirements in the Requirements Specification are met. It seeks to do this in a rigorous and justified manner. This document contains an overarching test plan, which seeks to outline each test subsystem, its strategy with regards to testing the associated requirements, and its execution strategy. This document then contains, for each subsystem, a detailed explanation of the set of tests included, and a test case for each individual test. Put together, the test subsystems group into a entire system test.

2 Test Plan

The system test plan has been split into five subsystems:

- **Functional Testing:** This test subsystem seeks to certify the functionality of the software against the requirements set in the Requirements Specification. This category will use black-box testing as its strategy, ignoring the design document and instead purely basing itself off of what the software is expected to do. In its execution, a test will typically first identify how the software should perform, verifying the functionality and reliability of the software given valid user behaviour, and then robustness given invalid user behaviour.
- **Structural Testing:** This test subsystem seeks to verify the structure and code logic of the software. We ensure here that each part of the code functions as expected given both valid and invalid input, and test the behaviour of the system in unexpected states. This will let us confirm the valid flow of our code, and ensure logic faults are caught. For the execution of the test, we will use JUnit to create individual tests for each case. Each test will have an initial setup phase, a test phase, and a teardown phase, to ensure independence of state between each test. A test will also use Mockito, a mocking library, to ensure that the failure of some other, unrelated component of the code does not affect the performance of the tested component in each test.
- **Performance Testing:** This test subsystem seeks to measure the behaviour of the software in extreme states, when under particular workloads or dealing with extremely large datasets. It is useful for testing a number of our non-functional requirements, notably reliability, scalability, and, obviously, performance. In its execution, The tests measure performance statistics given a normal or 'control' environment, then compare it to the performance statistic given a particular dataset or workload.
- **Acceptance Testing:** This test subsystem seeks to meet the requirements of the use cases in the requirements set in the Requirements Specification. This is also a black-box

testing category, as in functional testing, but unlike the aforementioned, we are instead performing a validation of the system: is our system actually what the user needs? In its execution, each test will typically follow a use case and verify that the needs of the user are met and further usability in the domain of the use case is available.

- **Installation Testing:** This test subsystem seeks to verify that the installation process is both successful and easy in the platforms to be supported. This means ensuring that the choices taken by the user are respected (location of installation, installation just for one user or for whole computer...), verify that all dependent files and libraries are successfully linked and loaded, and valid configurations and connectivity to the database. The execution of this category is simply an activity wherein the installation process is attempted in a particular environment, testing all decisions and options available in the installation.

3 Functional Testing

Create User Account

Test Case	First name, last name, username and password are mandatory
Description	The user cannot sign up without providing a valid first name, last name, a username and a password
Input/Steps	1. Go to 'Sign Up' 2. Leave all the input fields blank 3. Click 'Sign up'
Output/Results	• Sign up fails, the account is not created • An error window displays all the errors

Test Case	The username must be unique
Description	The user cannot sign up with an already existing username
Input/Steps	1. Successfully sign 2. Log out 3. Go to 'Sign up' 4. Fill in all the input fields 5. Set the username field to be the username of the user created in the first st e 6. Click 'Sign Up'
Output/Results	• Sign up failed, the account is not created • An error window notifies that the username already exists

Test Case	The password must be valid
Description	The user cannot sign up with a password not matching the required format, as specified in the business rules

Input/Steps	1. Go to 'Sign up' 2. Fill in all the input fields 3. Set the password to an alpha-numeric sequence of length less than 4 4. Set the repeat password field to match t he password field 5. Click 'Sign Up'
Output/Results	• Sign up failed, the account is not created • An error window notifies that the password is not valid

Test Case	The user account is successfully created
Description	The user must be able to successfully create an account provided that all input information is valid
Input/Steps	1. Go to 'Sign up' 2. Fill in all the input fields with valid data 3. Click 'Sign Up' 4. Moved to the login page: input the username and the passowrd 5. Click 'Log in'
Output/Results	• Sign up successful, the account is created • The user is logged-in to the newly created account

4 Structural Testing

4.1 Unit Test cases

com.github.comp354project.model.user.UserService.createUser(User)

Table 7: createUser(User)

Tester Name	Hrachya	
Test Date	1/31/18	
Class Name	com.github.comp354project.model.user.UserService	
Method Name	createUser(User)	
Purpose	This test suite tests the creation of a user	
Use Cases	01	
Test Scenarios		
createUser_withNullUser_shouldThrow		
Input Specification	user	
	null	
Expected Output	ValidationException is thrown	
Actual Output	ValidationException is thrown	
Bug Found	false	

Purpose	No null user can be saved in the database	
testCreateUser_withInvalidUser_shouldThrow		
Input Specification	user	errors
		4
Expected Output	ValidationException is thrown 4 exceptions are thrown because missing fields: username, password, firstname,	
Actual Output	ValidationException is thrown 4 exceptions are thrown because missing fields: username, password, firstname,	
Bug Found	false	
Purpose	No empty value user can be saved in the database	
testCreateUser_withValidUser_shouldReturnUser		
Input Specification	user	
	username: USERNAME password: PASSWORD firstName: FIRSTNAME lastName: LASTNAME	
Expected Output	User ID was autogenerated upon save The saved user is the same as the inputted user	
Actual Output	User ID was autogenerated upon save The saved user is the same as the inputted user	
Bug Found	false	
Purpose	A valid user should be inserted in the database	
testCreateUser_withExistingUsername_shouldThrow		
Input Specification	user	
	username: USERNAME password: PASSWORD firstName: FIRSTNAME lastName: LASTNAME	
Expected Output	ValidationException is thrown	
Actual Output	ValidationException is thrown	
Bug Found	false	
Purpose	A user cannot be created if the username is already taken	

com.github.comp354project.model.user.UserService.deleteBankAccount(Account)

Table 8: deleteBankAccount(Account)

Tester Name	Anne-Laure
Test Date	3/7/18
Class Name	com.github.comp354project.model.user.UserService
Method Name	deleteBankAccount(Account)
Purpose	This test suite tests the deletion of a bank account

Use Cases	04		
Test Scenarios			
testDeleteBankAccount_withNullAccount_ShouldThrow			
Input Specification	account		
	null		
Expected Output	ValidationException is thrown		
Actual Output	ValidationException is thrown		
Bug Found	false		
Purpose	No null account can be passed to the function		
testDeleteBankAccount_withoutBeingLoggedIn_ShouldThrow			
Input Specification	account	testUser	
	ID: 1 user: testUser bankName: TD type: Checking balance: 15823.12	ID: 1 firstName: Hrachya lastName: Hakobyan username: admin password: admin email: sample@email.com address: address phone: 111111	
Expected Output	AuthenticationException is thrown		
Actual Output	AuthenticationException is thrown		
Bug Found	false		
Purpose	A user that is not authenticated cannot delete his accounts		
testDeleteBankAccount_withoutProperAuthorisation_ShouldThrow			
Input Specification	testUser	user2	testAccount
	ID: 1 firstName: Hrachya lastName: Hakobyan username: admin password: admin email: sample@email.com address: address phone: 111111	username: username password: password firstName: firstname lastName: lastname ID: 999	ID: 1 user: testUser bankName: TD type: Checking balance: 15823.12
Expected Output	AuthorisationException is thrown		
Actual Output	AuthorisationException is thrown		
Bug Found	false		
Purpose	A user cannot modify the accounts of another user		
testDeleteBankAccount_WithProperAuthorisation_ShouldSucceed			
	testUser	testAccount	invocationCount

Input Specification	ID: 1 firstName: Hrachya lastName: Hakobyan username: admin password: admin email: sample@email.com address: address phone: 111111	ID: 1 user: testUser bankName: TD type: Checking balance: 15823.12	1
Expected Output	Execution of the deletion of the account once		
Actual Output	Execution of the deletion of the account once		
Bug Found	false		
Purpose	An authenticated user should succeed in deleting his own bank accounts		

com.github.comp354project.model.user.UserService.deleteUser(User)

Table 9: deleteUser(User)

Tester Name	Abed Jawhar
Test Date	3/13/18
Class Name	com.github.comp354project.model.user.UserService
Method Name	deleteUser(User)
Purpose	This test suite tests the creation of a user
Use Cases	02
Test Scenarios	
testDeleteUser_withNullUser_shouldThrow	
Input Specification	user null
Expected Output	ValidationException is thrown
Actual Output	ValidationException is thrown
Bug Found	false
Purpose	A null user can't be deleted
testDeleteUser_withNonexistantUser_shouldThrow	
Input Specification	testUser ID: 1 firstName: Hrachya lastName: Hakobyan username: admin password: admin email: sample@email.com address: address phone: 111111
Expected Output	ValidationException is thrown

Actual Output	ValidationException is thrown			
Bug Found	false			
Purpose	A user that does not exist can't be deleted			
testDeleteUser_withExistingtUser_shouldSucceed				
Input Specification	testUser	returnSize		
	ID: 1 firstName: Hrachya lastName: Hakobyan username: admin password: admin email: sample@email.com address: address phone: 111111	0		
Expected Output	The number of users with ID 1 is 0			
Actual Output	The number of users with ID 1 is 0			
Bug Found	false			
Purpose	A valid user should be deleted			
testDeleteUser_withExistingtUser_shouldDeleteAssociatedAccounts				
Input Specification	testUser	testAccount	returnSize	deleteAccountInvoc
	ID: 1 firstName: Hrachya lastName: Hakobyan username: admin password: admin email: sample@email.com address: address phone: 111111	ID: 1 user: testUser bankName: TD type: Checking balance: 15823.12	0	1
Expected Output	The number of users with ID 1 is 0 Delete account should be invocated 1 time			
Actual Output	The number of users with ID 1 is 0 Delete account should be invocated 1 time			
Bug Found	false			
Purpose	A valid user should be deleted and his accounts also			

com.github.comp354project.model.user.UserService.updateUser(User)

Table 10: updateUser(User)

Tester Name	Abed Jawhar
Test Date	3/13/18
Class Name	com.github.comp354project.model.user.UserService
Method Name	updateUser(User)

Purpose	This test suite tests the update of a user			
Use Cases	02			
Test Scenarios				
testUpdateUser_withNullUser_shouldThrow				
Input Specification	user			
	null			
Expected Output	ValidationException is thrown			
Actual Output	ValidationException is thrown			
Bug Found	false			
Purpose	A null user can't be updated			
testUpdateUser_withNonexistenttUser_shouldThrow				
Input Specification	testUser			
	ID: 1 firstName: Hrachya lastName: Hakobyan username: admin password: admin email: sample@email.com address: address phone: 111111			
Expected Output	ValidationException is thrown			
Actual Output	ValidationException is thrown			
Bug Found	false			
Purpose	A user that does not exist can't be updated			
testUpdateUser_withValidUser_shouldSucceed				
Input Specification	testUser	firstName	lastName	password
	ID: 1 firstName: Hrachya lastName: Hakobyan username: admin password: admin email: sample@email.com address: address phone: 111111	Abed	jawhar	admin2
Expected Output	The firstName is updated to 'Abed' The lastName is updated to 'jawhar' The password is updated to 'admin2'			
Actual Output	The firstName is updated to 'Abed' The lastName is updated to 'jawhar' The password is updated to 'admin2'			
Bug Found	false			
Purpose	A valid user should be updated			

com.github.comp354project.model.auth.SessionManager.login(username, password)

Table 11: login(username, password)

Tester Name	Hrachya				
Test Date	2/7/18				
Class Name	com.github.comp354project.model.auth.SessionManager				
Method Name	login(username, password)				
Purpose	This test suite tests the login of a user				
Use Cases	02				
Test Scenarios					
testLogin_withInvalidCredentials_shouldThrow					
Input Specification	username	password			
Expected Output	ValidationException is thrown				
Actual Output	ValidationException is thrown				
Bug Found	false				
Purpose	A user with invalid credentials should not be able to login				
testLogin_withValidCredentials_shouldReturnUser					
Input Specification	testUser	username	password	loggedIn	authenticateInvoc
	ID: 1				
	firstName: Hrachya				
	lastName: Hakobyan				
	username: admin				
	password: admin			true	1
	email: sample@email.com				
	address: address				
	phone: 111111				
Expected Output	The method authenticate is invoked 1 time The user is logged in The logged in user is equal to 'testUser' object				
Actual Output	The method authenticate is invoked 1 time The user is logged in The logged in user is equal to 'testUser' object				
Bug Found	false				
Purpose	A user with valid credentials should be able to login				
testLogin_withLoggedInUser_shouldThrow					
Input Specification	username	password			
Expected Output	UserLoggedInException is thrown				
Actual Output	UserLoggedInException is thrown				
Bug Found	false				

Purpose	A user that is already logged in should not be able to login again

com.github.comp354project.model.auth.SessionManager.logout()

Table 12: logout()

Tester Name	Hrachya			
Test Date	2/7/18			
Class Name	com.github.comp354project.model.auth.SessionManager			
Method Name	logout()			
Purpose	This test suite tests the function to logout			
Use Cases	02			
Test Scenarios				
testLogin_withInvalidCredentials_shouldThrow				
Input Specification	username	password	isLoggedIn	currentUser
			false	null
Expected Output	After logout, the login status should be false After logout, the current user should be null			
Actual Output	After logout, the login status should be false After logout, the current user should be null			
Bug Found	false			
Purpose	A user should be completely logged out of the application			

com.github.comp354project.model.auth.AuthenticationService.authenticate(username, password)

Table 13: authenticate(username, password)

Tester Name	Hrachya	
Test Date	2/3/18	
Class Name	com.github.comp354project.model.auth.AuthenticationService	
Method Name	authenticate(username, password)	
Purpose	This test suite tests the authentication of the user	
Use Cases	01	
Test Scenarios		
testAuthenticate_withInvalidUsernameOrPassword_shouldThrow		
Input Specification	username	password
	null	null
Expected Output	ValidationException is thrown The number of Validation Errors is equal to 2	
Actual Output	ValidationException is thrown The number of ValidationErrors is equal to 2	

Bug Found	false		
Purpose	A user with invalid credentials should not be able to authenticate		
testAuthenticate_withNonexistentUsername_shouldThrow			
Input Specification	username	password	
	usernam e	password	
Expected Output	ValidationException is thrown		
Actual Output	ValidationException is thrown		
Bug Found	false		
Purpose	A user witha nonexistent username should not be able to authenticate		
testAuthenticate_withIncorrectPassword_shouldThrow			
Input Specification	testUser	username	password
	ID: 1 firstName: Hrachya lastName: Hakobyan username: admin password: admin email: sample@email.com address: address phone: 11 1111	admin	INCORRECT_PASSWORD
Expected Output	ValidationException is thrown		
Actual Output	UserLoggedInException is thrown		
Bug Found	false		
Purpose	Authenticat ion with a valid username but an incorrect password should fail		
testAuthenticate_withCorrectCredentials_shouldReturnUser			
Input Specification	testUser	username	password
	ID: 1 firstName: Hrachya lastName: Hakobyan username: admin password: admin email: sample@email.com address: address phone: 11 1111	admin	admin
Expected Output	The authentication is successful and the authenticated u ser is returned The authenticated user is equal to the 'testUser' object		
Actual Output	The authentication is successful and the authenticated use r is returned The authenticated user is equal to the 'testUser' object		
Bug Found	false		
Purpose	Authentication with a valid username but an incorrect password should fail		

com.github.comp354project.model.account.AccountService.addAccount (request, user)

Table 14: addAccount(request, user)

Tester Name	Hrachya	
Test Date	2/7/18	
Class Name	com.github.comp354project.model.account.AccountService	
Method Name	addAccount(request, user)	
Purpose	This test suite tests the functionality of adding a new bank account	
Use Cases	03	
Test Scenarios		
testAddAccount_withInvalidParameters_shouldThrow		
Input Specification	request	accountOwner
	null	null
Expected Output	ValidationException is thrown The number of Validation Errors is equal to 2	
Actual Output	ValidationException is thrown The number of ValidationErrors is equal to 2	
Bug Found	false	
Purpose	Adding an account with invalid request or user should fail	
testAddAccount_withNonexistentRemoteAccount_shouldThrow		
Input Specification	request	accountOwner
	ID: 1	ID: 1 firstName: Hrachya lastName: Hakobyan username: admin password: admin email: sample@email.com address: address phone: 111111
Expected Output	AccountDoestNotExistException is thrown	
Actual Output	AccountDoestNotExistException is thrown	
Bug Found	false	
Purpose	A request f or adding a nonexistent account should fail	
testAddAccount_withInvalidUser_shouldThrow		
	request	accountOwner

Input Specification	ID: 1	ID: 1 firstName: Hrachya lastName: Hakobyan username: admin password: admin email: sample@email.com address: address phone: 111111	
Expected Output	ValidationException is thrown		
Actual Output	ValidationException is thrown		
Bug Found	false		
Purpose	Adding an account with an invalid owner should throw		
testAddAccount_withExistingAccount_shouldThrow			
Input Specification	request	accountOwner	
	ID: 1	ID: 1 firstName: Hrachya lastName: Hakobyan username: admin password: admin email: sample@email.com address: address phone: 111111	
Expected Output	AccountExistsException is thrown		
Actual Output	AccountExistsException is thrown		
Bug Found	false		
Purpose	Adding an already existing account should throw		
testAddAccount_withValidAccount_shouldReturnValidAccount			
Input Specification	request	accountOwner	expectedAccount
	ID: 1	ID: 1 firstName: Hrachya lastName: Hakobyan username: admin password: admin email: sample@email.com address: address phone: 111111	ID: 1 user: accountOwner bankName: TD type: Checking balance: 15823.12
Expected Output	The account is fetched and persisted in the database T he persisted account is equal to the 'expectedAccount' object The returned account is equal to the 'expectedAccount' object		
Actual Output	The account is fetched and persisted in the database The persisted account is equal to the 'expectedAccount' object The returned account is equal to the 'expectedAccount' object		

Bug Found	false
Purpose	Adding a va lid account with a valid owner must succeed

com.github.comp354project.model.account.AccountService.deleteAccount(account)

Table 15: deleteAccount(account)

Tester Name	Anne-Laure
Test Date	3/5/18
Class Name	com.github.comp354project.model.account.AccountService
Method Name	deleteAccount(account)
Purpose	This test suite tests the functionality of removing a user's bank account
Use Cases	04
Test Scenarios	
testDeleteAccount_withNullAccount_shouldThrow	
Input Specification	account
	null
Expected Output	ValidationException is thrown
Actual Output	ValidationException is thrown
Bug Found	false
Purpose	Delete a nu ll account should fail
testDeleteAccount_withAccountWithNullID_shouldThrow	
Input Specification	account
	ID: null user: null bankName: type: balance: 0
Expected Output	ValidationException is thrown
Actual Output	ValidationException is thrown
Bug Found	false
Purpose	Deleting an account with null ID should fail
testDeleteAccount_withNonExistentAccount_shouldThrow	
Input Specification	account
	ID: 1 user: accountOwner bankName: TD type: Checking balance: 15823.12
Expected Output	ValidationException is thrown
Actual Output	ValidationException is thrown
Bug Found	false

Purpose	Deleting an nonexistent account should fail
testDeleteAccount_withValidAccount_shouldSucceed	
Input Specification	account ID: 1 user: accountOwner bankName: TD type: Checking balance: 15823.12
Expected Output	The account is deleted from the database
Actual Output	The account is deleted from the database
Bug Found	false
Purpose	Deleting an existing account should succeed
testDeleteAccount_withValidAccount_shouldDeleteAllAssociatedTransactionsAndAccount	
Input Specification	account ID: 1 user:accountOwner bankName: TD type: Checking balance: 15823.12 transactions: [object Object]
Expected Output	The account is deleted from the database All the associated transactions are deleted from the database
Actual Output	The account is deleted from the database All the associated transactions are deleted from the database
Bug Found	false
Purpose	Deleting an existing account should delete all associated transactions

com.github.comp354project.model.account.TransactionService.update Transaction-Category(transactionID, category)

Table 16: updateTransactionCategory(transactionID, category)

Tester Name	Hrachya
Test Date	3/4/18
Class Name	com.github.comp354project.model.account.TransactionService
Method Name	updateTransactionCategory(transactionID, category)
Purpose	This test suite tests the functionality of updating the category of a transaction
Use Cases	08
Test Scenarios	
testUpdateCategory_withNullTransactionID_shouldThrow	

Input Specification	transactionID	category
	null	Leisure
Expected Output	ValidationException is thrown	
Actual Output	ValidationException is thrown	
Bug Found	false	
Purpose	Updating a null transaction ID should fail	
testUpdateCategory_withNonexistentTransaction_shouldThrow		
Input Specification	transactionID	category
	111111	Leisure
Expected Output	ValidationException is thrown	
Actual Output	ValidationException is thrown	
Bug Found	false	
Purpose	Updating a nonexistent transaction should fail	
testUpdateCategory_withNullCategory_shouldSucceed		
Input Specification	transactionID	category
	10	nu ll
Expected Output	The ’category’ of the transaction with the specified ID is set to null	
Actual Output	The ’category’ of the transaction with the specified ID is set to null	
Bug Found	false	
Purpose	Updating th e category of a valid transaction to null must succeed	
testUpdateCategory_withEmptyCategory_shouldSucceed		
Input Specification	transactionID	category
	10	
Expected Output	The ’category’ of the transaction with the specified ID is set to ”	
Actual Output	The ’category’ of the transaction with the specified ID is set to ”	
Bug Found	false	
Purpose	Updating th e category of a valid transaction to an empty string must succeed	
testUpdateCategory_withValidCategory_shouldSucceed		
Input Specification	transactionID	category
	10	Leisure
Expected Output	The ’category’ of the transaction with the specified ID is set to ’Leisure’	
Actual Output	The ’category’ of the transaction with the specified ID is set to ’Leisure’	
Bug Found	false	
Purpose	Updating the category of a valid transaction must succeed	
testUpdateCategory_withInvalidCategory_shouldThrow		
Input Specification	transactionID	category
	10	AAAAAAAAAAAAAAAAA
Expected Output	ValidationException is thrown	
Actual Output	ValidationException is thrown	
Bug Found	false	

Purpose	Updating th e category of a valid transaction to a an invalid value as determined by the bus iness rules must fail

5 Performance Testing

6 Acceptance Testing

7 Installation Testing

8 References

A Description of Input Files

Describe/include test data from input files.

B Description of Output Files

Describe/include test expected output that are output files.