**Andres Cordoba, Kevin Acevedo, Neal Rideout,**

**Oskar Danielsson, Sanjay Sohal, Tyler Serio**

**Bank Application**

**Test Design**

***April 15, 2015***

**Table of Contents**

[Purpose of this document](#h.l1m30fv0dc1i)

[Application Overview](#h.6jgh67yn5p5l)

[Testing Scope](#h.oaurlgbjp5md)

[In scope](#h.3mwwh38lnwp2)

[Out of Scope](#h.dufkwrscpf4l)

[Items not tested](#h.v9qtcjcdjfrc)

[Types of testing performed](#h.hcyjcq8mt0zq)

[Smoke Testing](#h.1zgo9gk3p7l9)

[JUnit Testing](#h.r4angs30tume)

[Test Design Identifier](#h.rqdcaao1ch72)

[Features to be Tested](#h.qgozclkiny9p)

[Approach Refinements](#h.g006zqpmllrp)

[Test Identification](#h.ctwoqfmovtdc)

[Feature Pass/Fail Criteria](#h.kp1spmul70ij)

[Exit Criteria](#h.bop1aswnoqkc)

[Conclusion](#h.tqb7meg2fg0)

[References](#h.qccoo91mlf0s)

# 

# 

# Purpose of this document

The purpose of this document is to test the Bank Application created by our group. This testing is to make sure that all the classes are working properly. Also, this will ensure us that each method’s performance reflects the requirement. The document will show how the tests are developed and performed.

# Application Overview

The bank system is designed to resemble an actual bank’s backend system, and has many modules that can be found in the real world such as account transfers, withdrawing and depositing, and opening accounts.

# Testing Scope

Since we are not delivering a full-fledged bank application, we are only performing the tests as requested in the assignment details.

## In scope

The following modules have been subject of functional testing:

* Creation of accounts
* Creation and additions of cards
* Accessing personal funds
* Handling business responsibilities

## Out of Scope

No testing of performance was executed.

## Items not tested

None

# Types of testing performed

## Smoke Testing

After every module was finished, its developer performed basic tests to make sure the major functionality was intact, and that the compilation succeeded.

## JUnit Testing

To verify the integrity of the system, JUnit tests were created with the intent to catch any leak of information that may occur during application execution. The results are compiled in this report.,

# Test Design Identifier

The tests will be done through JUnit through classes that we have created and we did individual tests for each class.

# Features to be Tested

The features we tested include:

* Creating an account
* Performing transactions
* Creating cards
* Handling responsibilities between branch manager and branch president
* Handling loan requests

# Approach Refinements

We refined our testing by using methods described in JUnit testing. For every test we performed in the project we created a separate class that would check to make sure the proper values were being output at the end of the test. We analyzed each test result to check for hidden null pointer exceptions.

One example of a situation where we found a hidden Null Pointer Exception whereSpecify the results of any analysis that provides a rationale for test case selection. For example, one might specify conditions that permit a determination of error tolerance (e.g., those conditions that distinguish valid inputs from invalid inputs).

Summarize the common attributes of any test cases. This may include input constraints that must be true for every input in the set of associated test cases, any shared environmental needs, any shared special procedural requirements, and any shared case dependencies.

# Test Identification

List the identifier and a brief description of each test case associated with this design. A particular test case may be identified in more than one test design specification. List the identifier and a brief description of each procedure associated with this test design specification.

|  |  |
| --- | --- |
| **Identifier** | **Description** |
| TC001 - Loan request | Verifies that a loan request returns an approved or denied amount with description. |
| TC002 - Create Card | Verifies that the CreateCard class creates a Debit Card. |
| TC003 - Create Card | Verifies that the CreateCard class creates a Credit Card. |
| TC004 - Create Account | Verifies that an account can be created for Savings |
| TC005 - Create Account | Verifies that an account can be created for Chequing |
| TC006 - Fund Deposit | Verifies that funds can be deposited into an account. |
| TC007 - Fund WIthdrawal | Verifies that funds can be withdrawn from an account. |
| TC008 - Verify transaction possible | Verifies that transactions has been made possible. |
| TC009 - Verify allowed cards | Verifies that there are only two type of account allowed. |
| TC010 - Verify CreateCard | Verifies that the CreateCard method only creates either a Credit or a Debit Card. |
| TC011- Verify Manager Approval | Verifies that the manager has the ability to approve a loan request |
| TC012-Verify President Approval | Verifies that the president has the ability to approve a loan request |
| TC013-Client request verification | Verifies that the amount and description matches the request of the client |

# Feature Pass/Fail Criteria

|  |  |  |
| --- | --- | --- |
| Test ID | Pass | Fail |
| TC001 | A requested loan amount or message is returned. | An exception is thrown if the next handler is missing. |
| TC002 | The message "A Debit Card has been created" is displayed. | An exception is thrown based on the value. A message “Invalid type of card provided” is displayed. |
| TC003 | The message "A Credit Card has been created" is displayed. | An exception is thrown based on the value. A message “Invalid type of card provided” is displayed. |
| TC004 | A message saying a Savings Account has been created should display. | An exception is thrown based on the value. A message “Invalid type of account provided” is displayed. |
| TC005 | An exception is thrown based on the value. | A message “Invalid type of account provided” is displayed. |
| TC006 | An exception is thrown based on the value. | A message indicating that the amount was less than or equal to zero will be displayed. |
| TC007 | A confirmation message and the amount that has been withdrawn will be returned | A message indicating that there is insufficient funds will be displayed. |
| TC008 | A message showing the accounts involved and the funds that have been exchanged will be returned | A message indicating that the funds could not be transferred and a reason why is returned |
| TC009 | An exception is thrown based on the value. | A message confirming the creation of the type of card is returned |
| TC010 | An exception is thrown based on the value. | A card has been created and a confirmation message returned. |
| TC011 | An exception is thrown based on the value. | A confirmation message and updates the loan |
| TC012 | An exception is thrown based on the value. | A confirmation message and updates the loan |
| TC013 | An exception is thrown based on the value. | A confirmation message appears. |

# Exit Criteria

1. There should be test cases to test all modules - **Yes**
2. All test cases should run without failure - **Yes**
3. All code defects should be rectified - **Yes**

# Conclusion

After meeting all exit criteria, our project is given the green light for submission. Gathering of all documentation and sources should be done before sending off.

# 

# 

# **References**

Dhall, S. QTP Tutorials & Interview Questions. (2015).

Examples of Test Design Specifications Template. Retrieved from Retrieved from <http://qtp.blogspot.ca/2009/06/test-design-specification-template.html>

Software Testing Help (2015, Apr 16).

A Simple 12 Steps Guide to Write an Effective Test Summary Report

Retrieved from <http://www.softwaretestinghelp.com/test-summary-report-template-download-sample/>

Hower, R. SoftwareQATest.com. (2015, Feb 8). :

Software QA and Testing Frequently-Asked-Questions Part 2

Retrieved from

<http://www.softwareqatest.com/qatfaq2.html#FAQ2_4>