Hodge Bank Test

This document describes the instructions for testing Hoge Bank web page user interface using Selenium automation tool with dependencies on TestNG, Maven, and Google Chrome Driver. Test Cases and instructions are included. Defects and Bugs are discussed at the end of each test case.

Test Background Description

Hoge Bank is a new digital bank in the region and they are going to release its first version to the public. However, before proceeding further, they need to test the features and make sure that it is usable. Bugs and defects should be described in this document. The web page is stable, however there are defects.

Contents

Test Environment and Tools

Selenium Eclipse IDE, IntelliJ IDEA, NetBeans, or any Integrated Development Environment (IDE) capable of building and running java web applications

TestNG

Maven

Google Chrome browser

Google Chrome Driver (https://chromedriver.chromium.org/downloads)
Important note: driver included, however google chrome driver must match user's runtime browser

Automation Script Technical Details

This script makes use of java's class inheritance method. TestNG is the framework used to validate requirements. Validation methods are in the Hoge_BankTest.java file and are namely assertEqual() and assertTrue(). The scripts total runtime is 51 seconds. The HB_RESOURCES.java file contains the WebDriver API used to manipulate the methods in each java class file. A Web Element is created in the java class files to capture elements inside the web page using namely the FindBy() method. The script uses an Interrupted Exception added to the Deposit and Withdrawal methods to create a 10 second timeout. Refactoring the code to better manage inheritance and reuse variables and methods can be done when bugs in the web page are fixed. New java classes can be created and have their methods called in the Hoge_BankTest.java class file for validation purposes. To run the scripts, download the package the HOGE_BANK.zip from https://github.com/carlosgsmith/HOGE_BANK.git then unzip and import HODGE_BANK folder into your IDE.

Requirements Matrix

Index	Requirement Description	Test Case Coverage
Requirement 1	User should be able to sign up an account	TC1
Requirement 1.1	User name cannot be blank	TC7
Requirement 1.2	User name cannot contain White Spaces	TC6
Requirement 1.3	Password cannot be less than 8 characters (<8)	TC4
Requirement 1.4	Password cannot be larger than 32 characters	TC5
Requirement 1.5	Password must contain numbers	TC8
Requirement 1.6	Password must contain upper letter	TC9
Requirement 2	User should be able to withdraw money	TC2
Requirement 2.1	Transaction Fee is 30% of intended withdrawal amount	TC2
Requirement 3	User should be able to deposit money	TC3
Requirement 3.1	Transaction Fee is 30% of the intended deposit amount	TC3
Requirement 4	Account balance should update every 10 seconds	TC2, TC3

Test Case 1 Sign_Up_TC1()

Description

This test case should validate the ability of a user to create an account on the Hoge Bank website. This process uses automation tools.

Automation Script Files: Hoge_Bank package, HB_SIGN_UP.java, HB_RESOURCES.java, HOGE_BANKTest.java
Automation Script Runtime: estimate 1 sec
Automation Script Location: https://github.com/carlosgsmith/HOGE_BANK.git
Automation Script Test Case: Sign_Up_TC1()

Prerequisite

Create multiple correct usernames and passwords
Usernames and Passwords can be managed in HB_RESOURCES file and HOGE_BANKTest.java

Test Steps

Step 1. Input user name

Step 2. Input password

Step 3. Click SignUp button

- ➔ **Req 1 User should be able to create an account**

Step 4. If 1st attempt fail repeat steps 1 to step 3

Step 5. Confirm user able to create an account

Step 6. Close Browser

Requirements:
Requirement 1 User should be able to sign up an account

Test Results

Verification and Validation processes confirm a user is able to sign up an account
Pass

Requirements Coverage:
Requirement 1

Additional Notes

User able to create an account, however issues exist and process is not user friendly

- ➔ **Issue 1** Account created on Login button, not SignUp button
- ➔ **Issue 2** Login Button removed from page after initial click
- ➔ **Issue 3** User clicks SignUp button, Login button action is triggered
- ➔ **Issue 4** User must make multiple attempts to login, not user friendly

Test Case 2 Deposit_Money_TC2()

Description

This test case should validate the ability of a user to deposit money from the Hoge Bank website. This process uses automation tools.

Automation Script Files: Hoge_Bank package, HB_DEPOSIT_MONEY.java, HB_RESOURCES.java, HOGE_BANKTest.java

Automation Script Runtime: estimate 11 secs
Automation Script Location: https://github.com/carlosgsmith/HOGE_BANK.git
Automation Script Test Case: Deposit_Money_TC2()

Prerequisite

User account must be created
Deposit amount can be updated/changed in the HB_DEPOSIT_MONEY and HOGE_BANKTest files

Test Steps

Step 1. Create an account then login

- ➔ Account created in TC1

Step 2. Get user current account balance

Step 3. Click deposit money link

Step 4. Wait for page to change

Step 5. Insert deposit amount into deposit input field

- ➔ Note Deposit Fee field is updated

Step 6. Click deposit button

Step 7. Wait for account balance to update

Step 8. Get new account balance

- ➔ **Requirement 3 User should be able to deposit money**
- ➔ **Requirement 3.1 Transaction Fee is 30% of the intended deposit amount**
- ➔ **Requirement 4 Account balance updated every 10 seconds**

Step 9. Confirm user able to deposit money

Step 10. Confirm user account balance is increased by deposit amount less transaction fee 30%

Step 11. Confirm user account is updated every 10 seconds

Requirements:

Requirement 3	User should be able to deposit money
Requirement 3.1	Transaction Fee is 30% of the intended deposit amount
Requirement 4	Account balance should update every 10 seconds

Test Results

Verification and Validation processes confirm a user is able to deposit money
Verification and Validation processes confirm transaction fee is 30% of the intended deposit amount
Verification and Validation processes confirm account balance updated every 10 seconds
Pass

Requirements Coverage:
Requirement 3
Requirement 3.1
Requirement 4

Additional Notes

Automation script confirm user is able to deposit money, however transaction is not updated in real time
User balance current balance not saved when user logs out then logs back in to account

- ➔ **Issue 1** Transaction not updated in real time
- ➔ **Issue 2** Account Balance not saved after deposit when user logs off then logs back in to account

Test Case 3 Withdraw_Money_TC3()

Description

This test case should validate the ability of a user to withdraw money from the Hoge Bank website. This process uses automation tools.

Automation Script Files: Hoge_Bank package, HB_WITHDRAW_MONEY.java, HB_RESOURCES.java, HOGE_BANKTest.java

Automation Script Runtime: estimate 11 secs
Automation Script Location: https://github.com/carlosgsmith/HOGE_BANK.git
Automation Script Test Case: Withdraw_Money_TC3()

Prerequisite

User account must be created
Withdrawal amount can be updated/changed in the HB_WITHDRAW_MONEY and HOGE_BANKTest files

Test Steps

Step 1. Create an account then login

- ➔ Account created in TC1

Step 2. Get user current account balance

Step 3. Click withdraw money link

Step 4. Wait for page to change

Step 5. Insert withdrawal amount into deposit input field

- ➔ **Requirement 2.1 Transaction Fee is 30% of intended withdrawal amount**
- ➔ Note Withdrawal Fee field is updated

Step 6. Click withdrawal button

Step 7. Wait for account balance to update

Step 8. Get new account balance

- ➔ **Requirement 2 User should be able to withdraw money**
- ➔ **Requirement 4 Account balance updated every 10 seconds**

Step 9. Confirm user able to withdraw money

Step 10. Confirm user account balance is decreased by withdrawal amount plus transaction fee 30%

Step 11. Confirm user account is updated every 10 seconds

Requirements:
Requirement 2 User should be able to withdraw money
Requirement 2.1 Transaction Fee is 30% of intended withdrawal amount
Requirement 4 Account balance should update every 10 seconds

Test Results

Verification and Validation processes confirm a user is able to withdraw money
Verification and Validation processes confirm transaction fee is 30% of intended withdrawal amount
Verification and Validation processes confirm account balance updated every 10 seconds
Pass

Requirements Coverage:
Requirement 2
Requirement 2.1
Requirement 4

Additional Notes

Automation script confirm user is able to withdraw money, however transaction is not updated in real time
User balance current balance not saved when user logs out then logs back in to account

- ➔ **Issue 1** Transaction not updated in real time
- ➔ **Issue 2** Account Balance not saved after withdrawal when user logs off then logs back in to account

Test Case 4 User_Password_Less_Than_Eight_TC4()

Description

This test case should validate that a user cannot create an account on the Hoge Bank website when password is less than 8 characters. This process uses automation tools.

Automation Script Files: Hoge_Bank package, HB_USERNAME_PASSWORD_CHECKS.java, HB_RESOURCES.java, HOGE_BANKTest.java
Automation Script Runtime: estimate 1 sec
Automation Script Location: https://github.com/carlosgsmith/HOGE_BANK.git
Automation Script Test Case: User_Password_Less_Than_Eight_TC4()

Prerequisite

Usernames and Passwords can be managed in HB_RESOURCES file, HB_USERNAME_PASSWORD_CHECKS.java and HOGE_BANKTest.java

Test Steps

Step 1. Input user name

Step 2. Input password less than eight characters

Step 3. Click SignUp button

- ➔ **Requirement 1.3 Password cannot be less than 8 characters (<8)**

Step 4. If 1st attempt fail repeat steps 1 to step 3

Step 5. Confirm website returns message “Password cannot be less than 8 characters”

Step 6. Close Browser

Requirements:
Requirement 1.3   Password cannot be less than 8 characters (<8)

Test Results

Verification and Validation processes confirm password cannot be less than 8 characters (<8)
Pass

Requirements Coverage:
Requirement 1.3

Additional Notes

Sign-up process is not user friendly

Test Case 5 User_Password_Greater_Than_Thirty_Two_TC5()

Description

This test case should validate that a user cannot create an account on the Hoge Bank website when password is less than 8 characters. This process uses automation tools.

Automation Script Files: Hoge_Bank package, HB_USERNAME_PASSWORD_CHECKS.java, HB_RESOURCES.java, HOGE_BANKTest.java
Automation Script Runtime: estimate 1 sec
Automation Script Location: https://github.com/carlosgsmith/HOGE_BANK.git
Automation Script Test Case: User_Password_Greater_Than_Thirty_Two_TC5()

Prerequisite

Usernames and Passwords can be managed in HB_RESOURCES file, HB_USERNAME_PASSWORD_CHECKS.java and HOGE_BANKTest.java

Test Steps

Step 1. Input user name

Step 2. Input password

Step 3. Click SignUp button

- ➔ **Requirement 1.4 Password cannot be larger than 32 characters**

Step 4. If 1st attempt fail repeat steps 1 to step 3

Step 5. Confirm website returns message “Password cannot be longer than 32 characters”

Step 6. Close Browser

Requirements:
Requirement 1.4 Password cannot be larger than 32 characters

Test Results

Verification and Validation processes confirm password cannot be larger than 32 characters
Pass

Requirements Coverage:
Requirement 1.4

Additional Notes

Sign-Up process is not user friendly

Test Case 6 User_Name_Contain_White_Spaces_TC6()

Description

This test case should validate that a user cannot create an account on the Hoge Bank website when username has white space. This process uses automation tools.

Automation Script Files: Hoge_Bank package, HB_USERNAME_PASSWORD_CHECKS.java, HB_RESOURCES.java, HOGE_BANKTest.java
Automation Script Runtime: estimate 1 sec
Automation Script Location: https://github.com/carlosgsmith/HOGE_BANK.git
Automation Script Test Case: User_Name_Contain_White_Sopaces_TC6()

Prerequisite

Usernames and Passwords can be managed in HB_RESOURCES file, HB_USERNAME_PASSWORD_CHECKS.java and HOGE_BANKTest.java

Test Steps

Step 1. Input user name

Step 2. Input password

Step 3. Click SignUp button

- ➔ **Requirement 1.2 User name cannot contain White Spaces**

Step 4. If 1st attempt fail repeat steps 1 to step 3

Step 5. Confirm website returns message “User name cannot contain whitespaces”

Step 6. Close Browser

Requirements:
Requirement 1.2 User name cannot contain White Spaces

Test Results

Verification and Validation processes confirm password cannot be larger than 32 characters
Pass

Requirements Coverage:
Requirement 1.2

Additional Notes

Sign-Up process is not user friendly

Test Case 7 User_Name_Blank_TC7()

Description

This test case should validate that a user cannot create an account on the Hoge Bank website when user name is blank. This process uses automation tools.

Automation Script Files: Hoge_Bank package, HB_USERNAME_PASSWORD_CHECKS.java, HB_RESOURCES.java, HOGE_BANKTest.java
Automation Script Runtime: estimate 1 sec
Automation Script Location: https://github.com/carlosgsmith/HOGE_BANK.git
Automation Script Test Case: User_Name_Blank_TC7()

Prerequisite

Usernames and Passwords can be managed in HB_RESOURCES file, HB_USERNAME_PASSWORD_CHECKS.java and HOGE_BANKTest.java

Test Steps

Step 1. Input user name

Step 2. Input password

Step 3. Click SignUp button

- ➔ **Requirement 1.1 User name cannot be blank**

Step 4. If 1st attempt fail repeat steps 1 to step 3

Step 5. Confirm website returns message "User name cannot be blank"

Step 6. Close Browser

Requirements:
Requirement 1.1 User name cannot be blank

Test Results

Verification and Validation processes confirm user name cannot be blank
Pass

Requirements Coverage:
Requirement 1.1

Additional Notes

Sign-Up process is not user friendly

Test Case 8 User_Password_Without_Numbers_TC8()

Description

This test case should validate that a user cannot create an account on the Hoge Bank website when password does not contain numbers. This process uses automation tools.

Automation Script Files: Hoge_Bank package, HB_USERNAME_PASSWORD_CHECKS.java, HB_RESOURCES.java, HOGE_BANKTest.java
Automation Script Runtime: estimate 1 sec
Automation Script Location: https://github.com/carlosgsmith/HOGE_BANK.git
Automation Script Test Case: User_Password_Greater_Than_Thirty_Two_TC5()

Prerequisite

Usernames and Passwords can be managed in HB_RESOURCES file, HB_USERNAME_PASSWORD_CHECKS.java and HOGE_BANKTest.java

Test Steps

Step 1. Input user name

Step 2. Input password

Step 3. Click SignUp button

- ➔ **Requirement 1.5 Password must contain numbers**

Step 4. If 1st attempt fail repeat steps 1 to step 3

Step 5. Confirm website returns message “Password must contain numbers”

Step 6. Close Browser

Requirements:
Requirement 1.5 Password must contain numbers

Test Results

Verification and Validation processes confirm password must contain numbers
Pass

Requirements Coverage:
Requirement 1.5

Additional Notes

Sign-Up process is not user friendly

Test Case 9 User_Password_Without_Capital_Letter_TC9()

Description

This test case should validate that a user cannot create an account on the Hoge Bank website when password does not contain capital letters. This process uses automation tools.

Automation Script Files: Hoge_Bank package, HB_USERNAME_PASSWORD_CHECKS.java, HB_RESOURCES.java, HOGE_BANKTest.java
Automation Script Runtime: estimate 1 sec
Automation Script Location: https://github.com/carlosgsmith/HOGE_BANK.git
Automation Script Test Case: User_Password_Without_Capital_Letter_TC9()

Prerequisite

Usernames and Passwords can be managed in HB_RESOURCES file, HB_USERNAME_PASSWORD_CHECKS.java and HOGE_BANKTest.java

Test Steps

Step 1. Input user name

Step 2. Input password

Step 3. Click SignUp button

- ➔ **Requirement 1.6 Password must contain upper letter**

Step 4. If 1st attempt fail repeat steps 1 to step 3

Step 5. Confirm website returns message "Password must contain uppercase letters"

Step 6. Close Browser

Requirements:
Requirement 1.6 Password must contain upper letter

Test Results

Verification and Validation processes confirm password must contain uppercase letters
Pass

Requirements Coverage:
Requirement 1.6

Additional Notes

Sign-Up process is not user friendly