UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

IMPAKD

Project	COS 301 Main Project
Product	Test Plan and Report

Version 1.2

Unit Test Plan and Report

Prepared by:

Member 1- 12031748
Member 2- 13278012
Member 3- 13134885
Member 4- 13049128

September 26, 2016

IMPAKD LINK

For further references see gitHub. September 26, 2016

Contents

1 Introduction

1.1 Purpose

This document combines the unit test plan and report into a single coherent artefact. Explain the purpose of your system and of this document here. Why is test driven development essential to your system.

1.2 Scope

The scope of this document is structured as follows. The features that are considered for testing are listed in section .... Tests that have been identifed from the requirements are discussed in detail in section .... Furthermore, this document outlines the test environment and the risks involved in the testing approaches that will be followed. Assumptions and dependencies of this test plan will also be mentioned. Section 7.1, 7.1 and 9 outlines, discusses and concludes on the results of the tests, respectively.

1.3 Test Environment

1.4 Assumptions and Dependencies

Unit Test Plan

2 Test Items

3 Functional Features to be Tested

Feature ID	RDS Source	Summary	Test Case ID
1	Implementation.PIO.BackEnd.test.Test	Passed	01
2	Implementation.PIO.BackEnd.test.Test	Passed	02
3	Implementation.PIO.BackEnd.test.Test	Passed	03
4	Implementation.PIO.BackEnd.test.Test	Passed	04

4 Test Cases

4.1 Test Case 1: AddPoints

4.1.1 Condition 1: AddPointsByEntity

4.1.1.1 Objective: The purpose of this test is to do the following...

4.1.1.2 Input: The following inputs will be used to test this functionality:

4.1.1.3 Outcome: The following are the expected outcomes for a pass result for the functionality:

4.1.2 Condition 2:

4.2 Test Case 2: Add a property with negative ID

4.2.1 Condition 1: Back-End Server must be running

4.2.1.1 Objective: The purpose of this test is to check how the validation of add property works when invalid values are injected

4.2.1.2 Input: Negative values are used as input , -1

4.2.1.3 Outcome: The function will throw a ArithmeticException

4.3 Test Case 3: Delete a property with negative ID

4.3.1 Condition 1: Back-End Server must be running

4.3.1.1 Objective: The purpose of this test is to check how the validation of add property works when invalid values are injected

4.3.1.2 Input: Negative values are used as input , -1

4.3.1.3 Outcome: The function will throw a ArithmeticException

4.4 Test Case 4: Find a property that does not exist

4.4.1 Condition 1: Back-End Server must be running

4.4.1.1 Objective: The purpose of this test is to test how the function handles a request when an id of a none-existing property is sent

4.4.1.2 Input: Any propertyID Value that is not in the database , 33

4.4.1.3 Outcome: The function will throw a NullPointerException

5 Item Pass/Fail Criteria

Each item tested must meet a certain criteria in order to pass. These criteria are as follows:
Back-End

- the back end server must be running

AddPropertyValidation

- the property must not exist in the database
- the input in the parameters must be invalid

DeletepropertyValidation

- the property must not exist in the database
- the input in the parameters must be invalid

FindProperty

- the back end server must be running
- the property must not exist in the database
- the input in the parameters must be valid

6 Test Deliverables

Artefacts to be produced as part of unit testing include the following:

- Test Plan
- Test Report
- Link to Test Code

Unit Test Report

7 Detailed Test Results

7.1 Overview of Test Results

The test that were done were mostly on the back-end.This is where we validate our values before they are inserted into the database.all the test we did passed even those that would cause the systems fail. This was done by handling the system failure correctly,anticipating what would go wrong and handling that failure in a correct way through try and catch blocks.The tests expected the system to fail and the error to be handled ,so once it it handled in the expected way, the test pass. Junit spring framework was used for unit testing in the backend

7.2 Functional Requirements Test Results

The tests we have created for this module are contained within the file PropertyTest. and gitHub.

7.2.1 Test Case 1 (TC 4.1.1)

The following results were obtained from the tests conducted. The tests to produce the following results have passed/failed and the reasons are stated below.

7.2.1.1 Result: Pass/ Fail.

7.2.2 Test Case 2 (TC 4.2)

The invalid value was handled though a try and catch block by throwing an ArithmeticException

7.2.2.1 Result: Pass.

7.2.3 Test Case 3 (TC 4.3)

The invalid value was handled though a try and catch block by throwing an ArithmeticException

7.2.3.1 Result: Pass.

7.2.4 Test Case 4 (TC 4.4)

The invalid value was handled though a try and catch block

7.2.4.1 Result: Pass.

8 Other

9 Conclusions and Recommendations

Test were done on the back-end to test for security features implemented as the application allows users to input data.Parameters are passed though the url to the back-end. So regardless of the validation we have in the font-end, malicious users might pass in malicious code after the validation in the front-end through the url, so it is important for the application to handle both user errors and malicious code, the test have assured that the countermeasures implemented work correctly.