BugTracker

Version 1.3

[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 13/10/2016 | 1.0 | Begin Test Plan | Tessa |
| 25/10/2016 | 1.1 | Edit function testing | Xinwei Jiang |
| 26/10/2016 | 1.2 | Added functional test cases | Tessa |
| 26/10/2016 | 1.3 | Added Performance, Security, User Interface test cases | Tessa |

Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 Background 4

1.3 Scope 5

1.4 Project Identification 6

2. Requirements for Test 7

3. Test Strategy 9

3.1.1 Data and Database Integrity Testing 9

3.1.2 Functional Testing 9

3.1.3 User Interface Testing 17

3.1.4 Performance Testing 18

3.1.5 Security and Access Control Testing 19

3.2 Tools 20

4. Resources 21

4.1 Workers 21

4.2 Project Milestones 23

5. Deliverables 24

5.1 Test Logs 24

5.2 Defect Reports 24

Test Plan

# Introduction

## Purpose

This Test Plan document for Bug Tracker supports the following objectives:

* Black Box testing to ensure all functional requirements outlined in the SRS Version 1.3 are met. Tests will be run to intentionally produce errors along with tests to produce expected results.
* White Box testing will include unit tests that cover class variable values, statement execution, decision and conditional statements used within controller classes.
* Integration Testing to discover any faults in the interface design and make sure the program works as expected.
* Performance Testing to test the program conforms to a selection of non-functional requirements outlined in SRS Version 1.3.
* Database and data integrity Testing.

Required Resources:

* A set of erroneous data and a set of expected data.
* QT test framework.

Deliverables:

* A set of test outcomes that highlight where errors can occur together with fixes for the errors.
* An outline of improvements needed in regards to the user interface.

• [Identify existing project information and the software components that should be tested.

• List the recommended Requirements for Test (high level).

• Recommend and describe the testing strategies to be employed.

• Identify the required resources and provide an estimate of the test efforts.

• List the deliverable elements of the test project]

## Background

The target of these tests is a bug tracking software application made up of a bug report component, user profile component, search component and a MySQL database. The application interfaces are created in QT and C++ is the language used to produce the functionality.

The goal of the project is to provide an application where users can report and search for bugs found in their software. Software Developers gain reputation points for fixing bugs while reporters gain points for reporting bugs. The system generates statistical reports on bug numbers and other features of bugs.

The system uses the model view controller architectural design so includes boundary, controller and entity classes. Development of the project began in early September 2016, a working application is almost complete.

[Enter a brief description of the target-of-test (components, application, system, etc.) and its goals. Include information such as major functions and features, its architecture, and a brief history of the project. This section should only be about three to five paragraphs.]

## Scope

Stages of testing include:

* Unit testing of the major functions that produce output or receive input via a user interface. The interface will not be used, unit tests will test specific data within the unit test function. The user interface will be used as part of black box testing. Minor utility functions will not have their own unit tests.
* Integration test to determine if the interfaces work well together and no faults are detected as a user moves between each interface. This will be carried out in the form of a walk-through of the entire program.
* System tests will test whether the system meets its major functional requirements and use cases at this stage. These include:

1. A user can add a bug.
2. A user can view a bug with all its comments and history.
3. A user can log in.
4. A user can register which creates their profile.
5. A user can edit their profile.
6. A user can search for bugs using a filtering system of keywords, bug id.
7. A user can search other users using keywords, user name.
8. The system produces bug reports, of most importance is “how many bugs reported for an application”.
9. A Manager can assign a bug to a developer to fix.
10. A developer can gain reputation points by fixing a number of bugs.
11. A user can gain reputation points by reporting a number of good bugs.
12. A user can add comments to a bug.  
      
    System tests will also cover the following performance requirements:
13. The system shall take no longer than 12 seconds to upload

[Describe the stages of testing­⎯for example, Unit, Integration, or System⎯and the types of testing that will be addressed by this plan, such as Function or Performance.

Provide a brief list of the target-of-test’s features and functions that will or will not be tested.

List any assumptions made during the development of this document that may impact the design, development or implementation of testing.

List any risks or contingencies that may affect the design, development or implementation of testing.

List any constraints that may affect the design, development or implementation of testing]

## Project Identification

The table below identifies the documentation and availability used for developing the *test plan*:

[Note: Delete or add items as appropriate.]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Document  (and version / date)** | **Created or Available** | **Received or Reviewed** | **Author or Resource** | **Notes** |
| Requirements Specification | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Functional Specification | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Use-Case Reports | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Project Plan | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Design Specifications | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Prototype | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Data Model or Flow | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Project or Business Risk Assessment | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |

# Requirements for Test

**Functional:**

Critical and some Essential Requirements outlined in SRS 1.3 will be tested, these are as follows:

Use Case 1: The system will provide a user of any type with a gui to log in.

Use Case 2: The system should provide a guest to register an account in the system.

Use Case 3: The system should provide any user with a GUI to modify their own profile details, name, E-mail, profile picture and subscriptions.

Use Case 3: The system should provide any user with a GUI to modify their own password.

Use Case 6: The system should provide a user with a GUI form to search for users of any type.

Search criteria include: First Name, Last Name, Email, Role, Account

Status and Key word.

Use Case 7: The system should provide a user with a GUI to view any non-administrator user profile.

Use Case 8: The system should allow a user to filter users of any type by First Name,

Last Name and Account Status.

Use Case 9: The system should allow a Reporter, Developer, Triager, Reviewer to clock up reputation points to be displayed in their user profile and whenever their name is used in commenting or bug reporting or bug solving.

Use Case 10: The system should provide a System Admin with a GUI to edit an existing non-administrator user’s profile in the Bug Tracker System.

Use Case 12: The system should provide a user with a GUI form to enter new bugs. The form should include:

* Application the reporter is reporting a bug on
* Title
* Text area to enter description of bug
* Option to receive responses from their post to their email
* Choose how often they want responses, e.g. weekly, monthly

Optional information: Version, file uploads, repeatable walkthrough, platform, severity, priority.

Use Case 13: The system should provide any user with a facility to add comments.

Use Case 11: The system should provide a user of any type (Reporter, Developer, Triager, and Reviewer) with a GUI form to subscribe to the bug.

Use Case 18: The system will provide a user of any type with a gui to view a bug. This includes all active and non-active bugs including duplicate bugs.

Use Case 20: The system should provide a Triager with a GUI to Assign bugs to developers.

Use Case 19: The system should provide a Triager with a GUI to edit existing Bugs in the System. Details such as title, description, status, severity, priority, assigned developer can be changed.

Use Case 18: The system should provide a Triager with a GUI to view current bugs assigned to each developer.

Use Case 4: The system should provide a facility for the developer to change the status of the bug. A developer can only change a bug assigned to them.

Use Case 4: The system should provide users a GUI form to search for bugs of any type.

Search criteria include: Application, Date, Developer, Key Word, and Status.

Use Case 4: The system should allow a user to filter bugs of any type by Application,

Date, Developer and Status.

**Non-functional requirements to be tested:**

* The system should respond to any action of any user within 12 seconds.
* The system should display search results within 12 seconds.
* The system should generate a report within no more than 3 seconds.
* The system should not require users to have any specific knowledge or special technical skill to be able to use it. Bug Tracker system should be appropriate for a computer-literate user community with no additional training on the system so that users can use it after at most one day of exploring the system. By providing well-formed graphical user interfaces.
* The system should provide users with appropriate feedback to recover from errors within 3 seconds by providing informative error messages.

# Test Strategy

Functional and non-functional requirements will be tested by group members to provide a form of black box testing. Results of tests will be recorded. These tests ensure the main functional and non-functional requirements have been met and work as expected.

Unit tests should be written for methods included in controller classes as these provide the main functionality.

Database methods will be tested without the use of a user interface and before the database class is made available to the rest of the group members. Data and source code of tests, along with evidence of data being input and retrieved from the database will be recorded. These tests are carried out to ensure data is stored and retrieved without error and can then be used by the group creating user interfaces.

### Data and Database Integrity Testing

The database class will use

[The databases and the database processes should be tested as a subsystem within the <**Project Name>**. These subsystems should be tested without the target-of-test’s User Interface as the interface to the data. Additional research into the DataBase Management System (DBMS) needs to be performed to identify the tools and techniques that may exist to support the testing identified below.]

|  |  |
| --- | --- |
| Test Objective: | [Ensure database access methods and processes function properly and without data corruption.] |
| Technique: | • [Invoke each database access method and process, seeding each with valid and invalid data or requests for data.  • Inspect the database to ensure the data has been populated as intended, all database events occurred properly, or review the returned data to ensure that the correct data was retrieved for the correct reasons] |
| Completion Criteria: | [All database access methods and processes function as designed and without any data corruption.] |
| Special Considerations: | • [Testing may require a DBMS development environment or drivers to enter or modify data directly in the databases.  • Processes should be invoked manually.  • Small or minimally sized databases (limited number of records) should be used to increase the visibility of any non-acceptable events.] |

### Functional Testing

### 

|  |  |
| --- | --- |
| Test Objective: | The system will provide a user of any type with a gui to log in. |
| Technique: | Black box testing   * Run test case 1 * Run test case 2 * Run test case 3 * Run test case 4 * Run test case 5   Compare the result with the expected output |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: | none |

|  |  |
| --- | --- |
| Test Objective: | The system should provide a guest to register an account in the system. |
|  | Black box testing   * Run test case 6 * Run test case 7 * Run test case 8 * Run test case 9 * Run test case 10 * Run test case 11 * Run test case 12   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: | none |

|  |  |
| --- | --- |
| Test Objective: | The system should provide any user with a GUI to modify their own profile details, name, E-mail, profile picture and subscriptions. |
| Technique: | Black box testing   * Run test case 13 * Run test case 14 * Run test case 15 * Run test case 16 * Run test case 17 * Run test case 18 * Run test case 19 * Run test case 20   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: |  |

|  |  |
| --- | --- |
| Test Objective: | The system should provide any user with a GUI to modify their own password. |
| Technique: | Black box testing   * Run test case 21 * Run test case 22 * Run test case 23   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: |  |

|  |  |
| --- | --- |
| Test Objective: | The system should provide a System Admin with a GUI to edit an existing non-administrator user’s profile in the Bug Tracker System. |
| Technique: | Black box testing   * Run test case 24 * Run test case 25 * Run test case 26 * Run test case 27 * Run test case 28 * Run test case 29 * Run test case 30   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: |  |

|  |  |
| --- | --- |
| Test Objective: | The system should provide any user with a GUI to view any non-administrator user profile. |
| Technique: | Black box testing   * Run test case 31 * Run test case 32   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: | none |

|  |  |
| --- | --- |
| Test Objective: | The system should provide any user with a GUI form to enter new bugs |
| Technique: | Black box testing   * Run test case 33 * Run test case 34 * Run test case 35 * Run test case 36 * Run test case 37   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: | none |

|  |  |
| --- | --- |
| Test Objective: | The system should provide any user with a GUI form to view a bug |
| Technique: | Black box testing   * Run all test cases between 38 … 42   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: | none |

### 

|  |  |
| --- | --- |
| Test Objective: | The system should provide any user with a GUI form to search and filter search criteria for users of any type |
| Technique: | Black box testing   * Run all test cases between 43 … 49   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: | none |

|  |  |
| --- | --- |
| Test Objective: | The system should allow a Reporter, Developer, Reviewer to clock up reputation points to be displayed in their user profile and whenever their name is used in commenting or bug reporting or bug solving. |
| Technique: | Black box testing   * Run all test cases between 50 … 51   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: | none |

|  |  |
| --- | --- |
| Test Objective: | The system should allow a Reporter, Developer, Reviewer to clock up reputation points to be displayed in their user profile and whenever their name is used in commenting or bug reporting or bug solving. |
| Technique: | Black box testing   * Run all test cases between 50-51   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: | none |

|  |  |
| --- | --- |
| Test Objective: | The system should provide a Triager with a GUI to Assign bugs to developers. |
| Technique: | Black box testing   * Run all test cases between 52 … 53   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: | none |

### 

|  |  |
| --- | --- |
| Test Objective: | The system should provide a Triager with a GUI to view current bugs assigned to each developer. |
| Technique: | Black box testing   * Run all test cases between 54   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: | none |

|  |  |
| --- | --- |
| Test Objective: | The system should provide a Triager with a GUI to view current bugs assigned to each developer. |
| Technique: | Black box testing   * Run all test cases between 54   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: | none |

|  |  |
| --- | --- |
| Test Objective: | The system should provide a facility for the developer to change the status of the bug. A developer can only change a bug assigned to them |
| Technique: | Black box testing   * Run all test cases between 55..56   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: | none |

|  |  |
| --- | --- |
| Test Objective: | The system should provide a Gui form to search for bugs of any type. |
| Technique: | Black box testing   * Run all test cases between 57..65   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: | none |

|  |  |
| --- | --- |
| Test Objective: | The system should allow a user to filter bugs of any type by Application, Date, Developer and Status |
| Technique: | Black box testing   * Run all test cases between 66..69   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: | none |

|  |  |
| --- | --- |
| Test Objective: | The system should provide a triager with a gui to edit an existing bug. |
| Technique: | Black box testing   * Run all test cases between 70..74   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: | none |

### User Interface Testing

### 

|  |  |
| --- | --- |
| Test Objective: | Walk through of the main interfaces of the program to discover any faults in the interface design and make sure the program works as one. |
| Technique: | Black box testing   * Run all test cases between 77..80   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: | none |

### Performance Testing

|  |  |
| --- | --- |
| Test Objective: | The system will take no longer than 12 seconds to load |
| Technique: | Black box testing   * Run test case 75   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: | A DSL internet connection |

### 

### 

|  |  |
| --- | --- |
| Test Objective: | The system will take no longer than 3 seconds to generate a report |
| Technique: | Black box testing   * Run test case 76   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: | A computer with at least 3.6GHz CPU and 4GB ram |

### Security and Access Control Testing

### 

|  |  |
| --- | --- |
| Test Objective: | The system will take no longer than 3 seconds to generate a report |
| Technique: | Black box testing   * Run test case 81...88   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: | none |

### 

### 

## Tools

The following tools will be employed for this project:

* QT tests for unit testing

# Resources

## Workers

This table shows the staffing assumptions for the project.

[NOTE: Delete or add items as appropriate.]

|  |  |  |
| --- | --- | --- |
| Human Resources | | |
| Worker | Minimum Resources Recommended  (number of full-time workers allocated) | Specific Responsibilities or Comments |
| Test Manager,  Test Project Manager | 1 | Provides management oversight.  Responsibilities:   * provide technical direction * acquire appropriate resources * provide management reporting |
| Test Designer | 1 | Identifies, prioritizes, and implements test cases.  Responsibilities:   * generate test plan * generate test model * evaluate effectiveness of test effort |
| Tester | 1 | Executes the tests.  Responsibilities:   * execute tests * log results * recover from errors * document change requests |
| Test System Administrator | 1 | Ensures test environment and assets are managed and maintained.  Responsibilities:   * administer test management system * install and manage worker access to test systems |
| Database Administratator, Database Manager | 1 | Ensures test data (database) environment and assets are managed and maintained.  Responsibilities:   * administer test data (database) |
| Designer | 1 | Identifies and defines the operations, attributes, and associations of the test classes.  Responsibilities:   * identifies and defines the test class(es) * identifies and defines the test packages |
| Implementer | 1 | Implements and unit tests the test classes and test packages.  Responsibilities:   * creates the test classes and packages implemented in the test model |

## Project Milestones

[Testing of <Project Name> should incorporate test activities for each of the test efforts identified in the previous sections. Separate project milestones should be identified to communicate project status accomplishments.]

|  |  |  |  |
| --- | --- | --- | --- |
| **Milestone Task** | **Effort** | **Start Date** | **End Date** |
| Plan Test | Complete Test Plan Doc | 12/10/2016 | 23/10/2016 |
| Design Test | Complete Test Cases | 14/10/2016 | 25/10/2016 |
| Implement Test | Complete Test Report | 25/10/2016 | 25/10/2016 |
| Execute Test | Record outcome in Test Report | 26/10/2016 | 26/10/2016 |
| Evaluate Test | Record recommendations | 26/10/2016 | 26/10/2016 |

# Deliverables

|  |  |  |
| --- | --- | --- |
| Documents | Group Member | Date By |
| Test Cases Functional | Selene & Tessa | 26/10/2016 |
| Test Cases Non-Functional | Selene & Tessa | 26/10/2016 |
| Test Report | Everyone – Mainly Selene | 26/10/2016 |
| Unit Tests | Toby and Nathaniel | 26/10/2016 |

## Test Logs

QT Tests – outcome recorded in the Test Report and QT Test logs file

## Defect Reports

Test Report is used to record and track outcomes of test cases.

# 