TESTING

# **Test Plan**

## **Purpose**

This test plan documentation for NoBugs bug-tracking system has been created with the following objectives in mind:

* To ensure that the system can comply with the requirements stated within the SRS. The system contains five subsystems that should be tested for consistency; account, bug reporting, comment, bug search, and statistics.

Testing of these subsystems will ensure that they are functional, will not produce errors, and give confidence that the subsystems are stable with all user input.

## **Background**

The software system being developed is a web-based bug-tracking system, designed to track bugs that users encounter on certain software products which allows developer to address bugs more rapidly compared to a developer not using a bug tracking system. The test cases designed for this system necessitate the testing of the subsystems mentioned in the purpose section of this document. Each will be tested for functionality, stability and correctness. In addition to that, the classes will also be tested.

The system endeavours to achieve a solid web-based bug tracking system that supports a range of users. This includes a public user, registered user (referred to as a reporter), triager, reviewer, developer, and administrator. Every user can search the system for bugs, but only registered users may report bugs and comment. The specialised roles have further functionalities such as assigning bugs to developers, uploading patches, reviewing patches, and managing user accounts.

To attain these requirements, the system has been divided into five different subsystems:

* Account management subsystem deals with the user accounts and permissions
* Comment subsystem deals with managing the user comments
* Searching subsystem allows the user to search the database for requested bug information
* Bug reporting subsystem allows registered users to create a bug report and submit it to the system
* Statistics subsystem calculates relevant information about the number of bugs created and solved

## **Scope of Testing**

The test cases designed to test the subsystems which are being developed. These test cases include functional, user interface, and performance testing which will be included in all the subsystems of the system to make certain the finished system will perform correctly.

## **Testing Documentation**

Documentation required for the test case generation, as well as its availability, is listed in the table below.

|  |  |  |
| --- | --- | --- |
| **Document Name** | **Available?** | **Reviewed?** |
| Software Requirement Specification | Yes | Yes |
| Use case descriptions | Yes | Yes |
| Prototype | Yes | Yes |

# **Test Requirements**

The following sections are designed to test the previously mentioned subsystems.

## **Accounts Subsystem**

* + 1. *Public User*

|  |  |
| --- | --- |
| **Use Case** | **Test Case ID** |
| Register account | TEST1 |
| Log in | TEST2 |

* + 1. *Reporter*

|  |  |
| --- | --- |
| **Use Case** | **Test Case ID** |
| Submit bug report | TEST3 |
| Submit comment on bug report | TEST4 |

* + 1. *Triager*

|  |  |
| --- | --- |
| **Use Case** | **Test Case ID** |
| Assign developer to bug | TEST5 |
| Merge Bugs | TEST6 |

* + 1. *Reviewer*

|  |  |
| --- | --- |
| **Use Case** | **Test Case ID** |
| Review bug patch | TEST8 |

* + 1. *Administrator*

|  |  |
| --- | --- |
| **Use Case** | **Test Case ID** |
| Activate User | TEST9 |
| Deactivate user | TEST10 |
| Create new user | TEST11 |
| Change user account type | TEST12 |
| Update user details | TEST13 |

## **Comments Subsystem**

|  |  |
| --- | --- |
| **Use Case** | **Test Case ID** |
| User submit comment | TEST4 |

## **Searching Subsystem**

|  |  |
| --- | --- |
| **Use Case** | **Test Case ID** |
| User search for bug with keywords | TEST14 |

## **2.4 Reporting Subsystem**

|  |  |
| --- | --- |
| **Use Case** | **Test Case ID** |
| User submit bug report | TEST3 |
| Triager assign bug report to developer | TEST5 |
| Reviewer review bug report | TEST8 |

## **2.5 Statistics**

|  |  |
| --- | --- |
| **Use Case** | **Test Case ID** |
| Generate statistics | TEST15 |

# **Test Cases**

## **Black Box Testing**

|  |
| --- |
| **Test** **Case** **ID**: TEST1 |
| **Purpose**: To test whether a public user can register |
| **Action**:   1. Public user clicks on the “Sign up” button 2. Enter their user name, email, password and confirm the password 3. Click the “Submit” button to finish the registration |
| **Output**:   1. Admin is presented with details of user on the profile page. |
| **Pass/Fail**: Pass |
| **Resultant** **Changes**: User’s information will be stored in the database, and the password will be hashed using MD5 function before storing. |
| **Error** **Messages**: If there’s duplicated email information, there will be an error message to ask the user to use another email address to register. |

|  |
| --- |
| **Test** **Case** **ID**: TEST2 |
| **Purpose**: To test whether a registered user account can login |
| **Action**:   1. Public user clicks on the “Sign in” button 2. Enter their email and password to log in 3. Click the “Login” button log in |
| **Output**:   1. The navigation bar will display profile and log out button. |
| **Pass/Fail**: Pass |
| **Resultant** **Changes**: NULL |
| **Error** **Messages**: If user inputs the wrong log in details, there will be an error message to ask the user to log in again using different log in details. |

|  |
| --- |
| **Test** **Case** **ID**: TEST3 |
| **Purpose**: To test whether a registered user can submit a bug report |
| **Action**:   1. Go to profile page 2. Click on “Report bug” in the dashboard section 3. Fill out details about the bug 4. Click the “submit” button to finalize the details 5. Return to the profile page |
| **Output**:   1. New bug is displayed at the top of the main page |
| **Pass/Fail**: Pass |
| **Resultant** **Changes**: Bug is added to the database, and is now searchable from the search bar. |
| **Error** **Messages**: If a field is left blank while trying to submit a bug, |

|  |
| --- |
| **Test** **Case** **ID**: TEST4 |
| **Purpose**: To test whether a registered user can submit a comment on the bug report. |
| **Action**:   1. Click on the bug from the main page or from searching the bug 2. Scroll to the bottom of the page and type a comment 3. Click the “submit” button under the comment box 4. Page redirects to the main page. |
| **Output**:   1. Comment will be displayed on the bug page the next time it is loaded. |
| **Pass/Fail**: Pass |
| **Resultant** **Changes**: Comment is added to the database, and is now viewable to other users browsing the bug. |
| **Error** **Messages**: NULL |

|  |
| --- |
| **Test** **Case** **ID**: TEST5 |
| **Purpose**: To test whether the triager can assign a bug to a developer |
| **Action**:   1. Go to profile page 2. On dashboard click “Assign bug to developer” 3. Click “Assign” on the relevant bug 4. Select the developer from the list 5. Click “Change” |
| **Output**:   1. Bug will be removed from the unassigned list in the triager’s dashboard |
| **Pass/Fail**: Pass |
| **Resultant** **Changes**: Bug assigned field is updated from NULL to the developer’s name. |
| **Error** **Messages**: NULL |

|  |
| --- |
| **Test** **Case** **ID**: TEST6 |
| **Purpose**: To test whether the triager can merge duplicate bugs |
| **Action**:   1. Go to profile page 2. Click on “Merge duplicate bugs” in the dashboard 3. For each duplicate bug you want to merge, click the “merge” button in the table. |
| **Output**:   1. Bug is removed from the listed when it is merged. |
| **Pass/Fail**: Pass |
| **Resultant** **Changes**: Duplicate bug is deleted from the database. |
| **Error** **Messages**: NULL |

|  |
| --- |
| **Test** **Case** **ID**: TEST8 |
| **Purpose**: To test whether the reviewer can review a bug patch |
| **Action**: |
| **Output**: |
| **Pass/Fail**: Pass |
| **Resultant** **Changes**: |
| **Error** **Messages**: NULL |

|  |
| --- |
| **Test** **Case** **ID**: TEST9 |
| **Purpose**: To test whether the administrator can activate a user |
| **Action**:   1. Go to profile page 2. Click the “user list” in the dashboard 3. Click the “modify” button next to the desired user 4. Change status to “active” 5. Click “change” |
| **Output**:   1. Returns administrator to profile page |
| **Pass/Fail**: Pass |
| **Resultant** **Changes**: The associated user account in the database will be changed to have a bool value indicating it’s activated. |
| **Error** **Messages**: NULL |

|  |
| --- |
| **Test** **Case** **ID**: TEST10 |
| **Purpose**: To test whether the administrator can deactivate a user |
| **Action**:   1. Go to profile page 2. Click the “user list” in the dashboard 3. Click the “modify” button next to the desired user 4. Change status to “deactivated” 5. Click “change” |
| **Output**:   1. Returns administrator to profile page |
| **Pass/Fail**: Pass |
| **Resultant** **Changes**: The associated user account in the database will be changed to have a bool value indicating it’s deactivated. |
| **Error** **Messages**: NULL |

|  |
| --- |
| **Test** **Case** **ID**: TEST11 |
| **Purpose**: To test whether the administrator can create a new user |
| **Action**:   1. Go to profile page 2. Click “add user” in the dashboard 3. Enter new user account details 4. Click “add” |
| **Output**:   1. Return to profile page |
| **Pass/Fail**: Pass |
| **Resultant** **Changes**: User’s information will be stored in the database, and the password will be hashed using MD5 function before storing. |
| **Error** **Messages**: If there’s duplicated email information, there will be an error message to ask the administrator to use another email address to register. |

|  |
| --- |
| **Test** **Case** **ID**: TEST12 |
| **Purpose**: To test whether the administrator can change the account type of a user |
| **Action**:   1. Go to profile page 2. Click the “user list” 3. Click the “modify” button next to the desired user 4. Change role to desired role 5. Click “change” |
| **Output**:   1. Returns administrator to profile page |
| **Pass/Fail**: Pass |
| **Resultant** **Changes**: The users account type is updated by changing the account type in the database. |
| **Error** **Messages**: NULL |

|  |
| --- |
| **Test** **Case** **ID**: TEST13 |
| **Purpose**: To test whether the registered user can update their user details. |
| **Action**:   1. Go to profile page 2. Click “edit” 3. Update details accordingly 4. Click “save changes” |
| **Output**:   1. Updated user details displayed |
| **Pass/Fail**: Pass |
| **Resultant** **Changes**: User’s details are updated in the database with the new values. |
| **Error** **Messages**: If there’s duplicated email information, there will be an error message to ask the user to use another email address. |

|  |
| --- |
| **Test** **Case** **ID**: TEST14 |
| **Purpose**: To test whether a registered user can search for a bug using keywords |
| **Action**:   1. In the search bar on the main page, enter keywords relating to the required bug 2. Press enter 3. The page will load with bug related to the keyword search. |
| **Output**:   1. A list of bugs related to the keyword will appear on the page. |
| **Pass/Fail**: Pass |
| **Resultant** **Changes**: The bug view count will increase by one. |
| **Error** **Messages**: If there are no bugs that match the description, no bugs will be displayed on screen. |

|  |
| --- |
| **Test** **Case** **ID**: TEST15 |
| **Purpose**: To test whether the system can generate statistics relevant for users to view |
| **Action**:   1. Load the main page |
| **Output**:   1. The statistics are loaded at the bottom of the web page. |
| **Pass/Fail**: Pass |
| **Resultant** **Changes**: NULL |
| **Error** **Messages**: NULL |

# **Summary of Testing**

The results of testing each subsystem indicate that the system “NoBugs” is operational. There were no system errors encountered while testing the product, thus meaning that the product complies to what was laid out in the software requirement specification.