Bug Tracking System

Software Requirements Specification

Version <1.0>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <dd/mmm/yy> | <x.x> | <details> | <name> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction

1.1 Purpose

1.2 Scope

1.3 Definitions, Acronyms and Abbreviations

1.4 References

1.5 Overview

2. Overall Description

3. Specific Requirements

3.1 Functionality

3.1.1 <Functional Requirement One>

3.2 Usability

3.2.1 <Usability Requirement One>

3.3 Reliability

3.3.1 <Reliability Requirement One>

3.4 Performance

3.4.1 <Performance Requirement One>

3.5 Supportability

3.5.1 <Supportability Requirement One>

3.6 Design Constraints

3.6.1 <Design Constraint One>

3.7 Online User Documentation and Help System Requirements

3.8 Purchased Components

3.9 Interfaces

3.9.1 User Interfaces

3.9.2 Hardware Interfaces

3.9.3 Software Interfaces

3.9.4 Communications Interfaces

3.10 Licensing Requirements

3.11 Legal, Copyright and Other Notices

3.12 Applicable Standards

4. Supporting Information

Software Requirements Specification

# Introduction

## Purpose

The purpose of this SRS is to identify the specifications of the functional requirements for the Bug Tracking System (BTS). The document also describes the non-functional requirements, design constraints, product perspectives and other requirements that are essential for the entire system to be clearly understood.

This document is intended for the software development team of a product and the potential users. The system will allow a platform for these people to report, track and fix bugs that arise in the product they developed/used.

## Scope

The software system to be created will be a stand-alone database application. The system will be generic and able to support many different products.

This document is proposed for two different groups of users; the software development team and the users of a product supported on the system to be created.

The software development team consists of an administrator, triagers, developers, reviewers and reporters. The administrator has control over the entire system and manages all other users. The triagers main role is to assign a developer to fix a bug that has been submitted by a reporter. The reviewer must check that the developer has fixed the bug correctly.

Section 3 of this document includes the functional requirements of the system. For the prospective customers to clearly understand these functionalities it is developed into four subsystems, which are:

* Users Managements System – manages profiles of all users
* Bug Management Subsystem- manages profiles of all bug reports in the Bug Tracker system
* Search Management Subsystem - manages search functions
* Report management system –

The Bug Tracker System also supports users browsing bug reports and generating bug report statistics for various users to inspect. Eg Number of bugs reports each day, number of bugs for a particular version of the software, etc. The Bug Tracker can also support different versions of a app/programs and even multiple programs.

The Bug Tracker system also uses a reputation system for each type of user. This encourages general users to report as many bugs as they find and rewards them if the bug happens to impact many other users (who can show their interest in a particular bug by subscribing to status updates on the report), encourages developers to work productively to solve critical bugs (without causing more problems)

## Definitions, Acronyms and Abbreviations

* Refer to Glossary.doc

## References

* **1.5 Overview**

The remainder of this document is divided into 2 main sections:

* Section 2: Overall Description will explain the requirements and general dynamics of the system.
* Section 3: Specific Requirements will include all the software requirements that must be achieved to satisfy the customer.

# Overall Description

## **Product Perspectives**

All software that is created will encounter issues and bugs, and it is important that these are dealt with quickly and efficiently so that users continue to use it and companies can continue to make money.

The need for this system is to assist the software development team of a product to efficiently fix bugs and for the end-user to report bugs. The system aims to create a generic platform to be suitable for different products.

## Product Functions

The main functions of the Bug tracking system are to

## **User Characteristics**

The users of the bug tracking system include administrators, reviewers, triagers, reporters and developers.

* Administrators are very knowledgeable on managing database applications.
* Triagers must have knowledge in project management, software development and testing.
* Developers are experienced software developers.
* Reviewers must be knowledgeable in software testing.
* Reporters are end users of a product supported on the bug tracking system who have basic computer skills to use the application.
* User with no account

## **Assumptions and dependencies**

The following assumptions and dependencies for the system are:

*[This section of the* ***SRS*** *should describe the general factors that affect the product and its requirements. This section does not state specific requirements. Instead, it provides a background for those requirements, which are defined in detail in Section 3, and makes them easier to understand. Include such items as:*

*• product perspective*

*• product functions*

*• user characteristics*

*• constraints*

*• assumptions and dependencies*

*• requirements subsets]*

# Specific Requirements

Each requirement (either functional or a non-functional one) of E-Library system is ranked

based on its level of importance.

* Critical: highest importance level. Critical requirements are those that reflect core functionalities of E-Library system and must be firstly implemented
* Essential: second highest importance level. Essential requirements are those that reflect important functionalities of E-Library system and must be covered when Critical requirements have successfully implemented
* Desirable: medium importance level. Desirable requirements are those that reflect necessary but not critical functionalities of E-Library system and should be implemented when all Critical and Essential requirements have been fulfilled

* Optional: lowest importance level. Optional requirements are those that reflect enhanced functionalities of E-Library system and should be considered only when all Critical, Essential and Desirable requirements are completed

# Functionality Requirements

### 3.1.1 User Management Subsystem

This section includes the functionalities that the Bug Tracking system provides to manage their profiles.

### 3.1.1.1 Administrator Side

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type:** Functional | **Use Case:** |
| **Description:** The system should provide a reporter with a GUI to import user profiles from an external database. | | |
| **Rationale:** Administrator wants to import users from external database. | | |
| **Source:** Administrator | | |
| **Fit Criterion:** User profiles are successfully imported | | |
| **Dependencies:** None | | |
| **Rank of Importance:** Critical | | |
| **Supporting Materials:** None | | |
| **History:** | | |

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type:** Functional | **Use Case:** |
| **Description:** The system should provide a reporter with a GUI to view requests to change the profile of a non-administrative user. | | |
| **Rationale:** Administrator wants view requests to change profiles of a user. | | |
| **Source:** Administrator | | |
| **Fit Criterion:** Requests can be successfully viewed. | | |
| **Dependencies:** The profile to be changed must exist. Refer to F\_ | | |
| **Rank of Importance:** Critical | | |
| **Supporting Materials:** None | | |
| **History:** | | |

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type:** Functional | **Use Case:** |
| **Description:** The system should provide a reporter with a GUI to add a new user(non-administrative). A profile consists of: **Username, password,** | | |
| **Rationale:** Administrator wants to add a new user to the system. | | |
| **Source:** Administrator | | |
| **Fit Criterion:** A new user account may be successfully added | | |
| **Dependencies:** None | | |
| **Rank of Importance:** Critical | | |
| **Supporting Materials:** None | | |
| **History:** | | |

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type:** Functional | **Use Case:** |
| **Description:** The system should provide a reporter with a GUI to edit a new user(non-administrative). A profile consists of: **Username, password,** | | |
| **Rationale:** Administrator wants to add a new user to the system. | | |
| **Source:** Administrator | | |
| **Fit Criterion:** A new user account may be successfully edited | | |
| **Dependencies:** The profile to be edited must exist. Refer to F\_ | | |
| **Rank of Importance:** Critical | | |
| **Supporting Materials:** None | | |
| **History:** | | |

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type:** Functional | **Use Case:** |
| **Description:** The system should provide a reporter with a GUI to delete one or more existing non-administrative user accounts. | | |
| **Rationale:** Administrator wants to delete a user from the system. | | |
| **Source:** Administrator | | |
| **Fit Criterion:** A new user account may be successfully deleted | | |
| **Dependencies:** The profile to be deleted must exist. Refer to F\_ | | |
| **Rank of Importance:** Critical | | |
| **Supporting Materials:** None | | |
| **History:** | | |

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type:** Functional | **Use Case:** |
| **Description:** The system should provide a administrator with a GUI to edit their own profile a profile consists of: | | |
| **Rationale:** Administrator wants to edit their own profile. | | |
| **Source:** Administrator | | |
| **Fit Criterion:** The profile can be successfully edited. | | |
| **Dependencies:** The profile to be changed must exist. Refer to F\_ | | |
| **Rank of Importance:** Critical | | |
| **Supporting Materials:** None | | |
| **History:** | | |

3.1.1.2 User Side

This section includes all the functions that a non-administrative user (Triager, Developer, Reviewer, Reporter) may use to interact with the bug tracking system.

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type:** Functional | **Use Case:** |
| **Description:** The system should provide a user with a GUI to log into the Bug Tracking system. | | |
| **Rationale:** A user wants to log into the system. | | |
| **Source:** Administrator, Triager, Developer, Reviewer, Reporter | | |
| **Fit Criterion:** A user may log in successfully | | |
| **Dependencies:** None | | |
| **Rank of Importance:** Critical | | |
| **Supporting Materials:** None | | |
| **History:** | | |

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type:** Functional | **Use Case:** |
| **Description:** The system should provide a reporter with a GUI to log out of the Bug Tracking system. | | |
| **Rationale:** A user wants to log off the system. | | |
| **Source:** Administrator, Triager, Developer, Reviewer, Reporter | | |
| **Fit Criterion:** A user may log out successfully. | | |
| **Dependencies:** None | | |
| **Rank of Importance:** Critical | | |
| **Supporting Materials:** None | | |
| **History:** | | |

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type:** Functional | **Use Case:** |
| **Description:** The system should provide a user of any type with a GUI to view their own profile. A profile consists of: Username, User type, | | |
| **Rationale:** A user wants to view their own profile. | | |
| **Source:** Administrator, Triager, Developer, Reviewer, Reporter | | |
| **Fit Criterion:** A user profile should be viewed successfully. | | |
| **Dependencies:** The profile to be viewed must exist. Refer to F\_ | | |
| **Rank of Importance:** Critical | | |
| **Supporting Materials:** None | | |
| **History:** | | |

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type:** Functional | **Use Case:** |
| **Description:** The system should provide a user of any type with a GUI to change their own password. | | |
| **Rationale:** A user wants to change their password. | | |
| **Source:** Administrator, Triager, Developer, Reviewer, Reporter | | |
| **Fit Criterion:** A user password should be changed successfully | | |
| **Dependencies:** The profile with a password to be changed should exist. Refer to F\_ | | |
| **Rank of Importance:** Critical | | |
| **Supporting Materials:** None | | |
| **History:** | | |

3.2.1 Bug Management Subsystem

3.2.1.1 Submitting a Bug report

3.2.1.2 Assigning a Bug

3.2.1.3 Fixing a Bug

3.2.1.4 Reviewing a Bug

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type: Functional** | **Use Case #:** |
| **Description:** The system should provide a reporter a GUI to view the status of a bug. | | |
| **Rationale:** A user wants to view a bug’s status. | | |
| **Source:** Reporter | | |
| **Fit Criterion:** A Bug’s status is displayed to the reporter. | | |
| **Dependencies:** None | | |
| **Rank of Importance:** Critical | | |
| **Supporting Materials:** None | | |
| **History:** | | |

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type: Functional** | **Use Case #:** |
| **Description** The system should provide a reporter with a GUI form to submit a bug report. A Bug report consists of: Bug Title, Bug Description, Product | | |
| **Rationale:** A reporter wants to submit a bug report. | | |
| **Source:** Reporter | | |
| **Fit Criterion:** A bug is submitted successfully. | | |
| **Dependencies:** None | | |
| **Rank of Importance:** Critical | | |
| **Supporting Materials:** None | | |
| **History:** | | |

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type: Functional** | **Use Case #:** |
| **Description** The system should provide a reporter with a GUI to view all bugs that they have reported with the bug’s current status. | | |
| **Rationale:** A reporter wants to keep track of the bugs they have reported. | | |
| **Source:** Reporter | | |
| **Fit Criterion:** The list of bugs the reporter has submitted should be displayed. | | |
| **Dependencies:** The bug reports must exist. | | |
| **Rank of Importance:** Essential | | |
| **Supporting Materials:** None | | |
| **History:** | | |

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type: Functional** | **Use Case #:** |
| **Description** The system should automatically send a notification message to reporter when a bug status has been updated. | | |
| **Rationale:** A reporter wants to be notified if the bug status has changed. | | |
| **Source:** Reporter | | |
| **Fit Criterion:** The notification message is sent to the reporter. | | |
| **Dependencies:** The reporter has reported a bug. | | |
| **Rank of Importance:** Essential | | |
| **Supporting Materials:** None | | |
| **History:** | | |

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type: Functional** | **Use Case #:** |
| **Description** The system should provide the user with a GUI to add a comment to a bug report. | | |
| **Rationale:** A reporter wants to comment on a bug report. | | |
| **Source:** Reporter | | |
| **Fit Criterion:** The comment is successfully added. | | |
| **Dependencies:** The bug report must exist. | | |
| **Rank of Importance:** Essential | | |
| **Supporting Materials:** None | | |
| **History:** | | |

3.1.3 Search Management System

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type: Functional** | **Use Case #:** |
| **Description** The system should provide a reporter with a GUI search form  Search criteria should include: Bug Id, user name, product, keyword, user type, | | |
| **Rationale:** A reporter wants to search . | | |
| **Source:** Administrator, Traiger, Developer, Reviewer, Reporter, Browse | | |
| **Fit Criterion:** A GUI search form is displayed. | | |
| **Dependencies:** None | | |
| **Rank of Importance:** Critical | | |
| **Supporting Materials:** None | | |
| **History:** | | |

3.1.4 Report Management System

**Non-Functional Requirements**

## Performance, Usability, Reliability

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type: Performance** | **Use Case #:** |
| **Description** The system should be able to work 24 hours per day, seven days per week except during the maintenance time. | | |
| **Rationale:** A reporter wants to comment on a bug report. | | |
| **Source:** Administrator, Triager, Developer, Reviewer, Reporter | | |
| **Fit Criterion:** Any user who has an authorised account can access the system at any time. | | |
| **Dependencies:** None. | | |
| **Rank of Importance:** Essential | | |
| **Supporting Materials:** None | | |
| **History:** | | |

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type: Performance** | **Use Case #:** |
| **Description** The system should respond to any action of a user within 30 seconds. | | |
| **Rationale:** A user wants a response to their action quickly. | | |
| **Source:** Administrator, Triager, Developer, Reviewer, Reporter | | |
| **Fit Criterion:** Any response to a user action should be performed within 30 seconds. | | |
| **Dependencies:** None. | | |
| **Rank of Importance:** Essential | | |
| **Supporting Materials:** None | | |
| **History:** | | |

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type: Performance** | **Use Case #:** |
| **Description** The system should display search results within 30 seconds. | | |
| **Rationale:** A user wants a receive results quickly. | | |
| **Source:** Administrator, Triager, Developer, Reviewer, Reporter | | |
| **Fit Criterion:** Search results should be displayed within 30 seconds. | | |
| **Dependencies:** None. | | |
| **Rank of Importance:** Essential | | |
| **Supporting Materials:** None | | |
| **History:** | | |

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type: Performance** | **Use Case #:** |
| **Description** The system should generate a report in no longer than 30 seconds. | | |
| **Rationale:** A user wants to receive results from the report quickly. | | |
| **Source:** Administrator, Triager, Developer, Reviewer, Reporter | | |
| **Fit Criterion:** Search results should be displayed within 30 seconds. | | |
| **Dependencies:** None. | | |
| **Rank of Importance:** Essential | | |
| **Supporting Materials:** None | | |
| **History:** | | |

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type: Usability** | **Use Case #:** |
| **Description** The system should be easy enough to use by users who are computer-literate but do not have special knowledge or skills. A user should not require more than one day of exploring the system until it is fully understood. | | |
| **Rationale:** A user should be able to use the system without prior training. | | |
| **Source:** Administrator, Triager, Developer, Reviewer, Reporter | | |
| **Fit Criterion:** A user can use the system after no longer than one day of exploration. | | |
| **Dependencies:** None. | | |
| **Rank of Importance:** Essential | | |
| **Supporting Materials:** None | | |
| **History:** | | |

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type: Usability** | **Use Case #:** |
| **Description** The system should include step-by-step instructions for the user. | | |
| **Rationale:** A user should be able to use the system without prior training. | | |
| **Source:** Administrator, Triager, Developer, Reviewer, Reporter | | |
| **Fit Criterion:** A user can use the system after no longer than one day of exploration. | | |
| **Dependencies:** None. | | |
| **Rank of Importance:** Desirable | | |
| **Supporting Materials:** None | | |
| **History:** | | |

|  |  |  |
| --- | --- | --- |
| **Requirement #:** | **Requirement Type: Reliability** | **Use Case #:** |
| **Description** The system should be able to support up to 100 simultaneous users at any time. | | |
| **Rationale:** To support simultaneous usage of the system. | | |
| **Source:** Administrator, Triager, Developer, Reviewer, Reporter | | |
| **Fit Criterion:** The system should successfully allow 100 simultaneous users at any time. | | |
| **Dependencies:** None. | | |
| **Rank of Importance:** Essential | | |
| **Supporting Materials:** None | | |
| **History:** | | |

## Usability

*[This section should include all of those requirements that affect usability. For example,*

*• specify the required training time for a normal users and a power user to become productive at particular operations*

*• specify measurable task times for typical tasks or base the new system’s usability requirements on other systems that the users know and like*

*• specify requirement to conform to common usability standards, such as IBM’s CUA standards Microsoft’s GUI standards]*

### <Usability Requirement One>

*[The requirement description goes here.]*

## Reliability

*[Requirements for reliability of the system should be specified here. Some suggestions follow:*

*• Availability—specify the percentage of time available ( xx.xx%), hours of use, maintenance access, degraded mode operations, etc.*

*• Mean Time Between Failures (MTBF) — this is usually specified in hours, but it could also be specified in terms of days, months or years.*

*• Mean Time To Repair (MTTR)—how long is the system allowed to be out of operation after it has failed?*

*• Accuracy—specify precision (resolution) and accuracy (by some known standard) that is required in the system’s output.*

*• Maximum Bugs or Defect Rate—usually expressed in terms of bugs per thousand of lines of code (bugs/KLOC) or bugs per function-point( bugs/function-point).*

*• Bugs or Defect Rate—categorized in terms of minor, significant, and critical bugs: the requirement(s) must define what is meant by a “critical” bug; for example, complete loss of data or a complete inability to use certain parts of the system’s functionality.]*

### <Reliability Requirement One>

*[The requirement description.]*

## Supportability

*[This section indicates any requirements that will enhance the supportability or maintainability of the system being built, including coding standards, naming conventions, class libraries, maintenance access, maintenance utilities.]*

### <Supportability Requirement One>

*[The requirement description goes here.]*

## Design Constraints

*[This section should indicate any design constraints on the system being built. Design constraints represent design decisions that have been mandated and must be adhered to. Examples include software languages, software process requirements, prescribed use of developmental tools, architectural and design constraints, purchased components, class libraries, etc.]*

### <Design Constraint One>

*[The requirement description goes here.]*

## Purchased Components

*[This section describes any purchased components to be used with the system, any applicable licensing or usage restrictions, and any associated compatibility and interoperability or interface standards.]*

## Interfaces

*[This section defines the interfaces that must be supported by the application. It should contain adequate specificity, protocols, ports and logical addresses, etc. so that the software can be developed and verified against the interface requirements.]*

### User Interfaces

*[Describe the user interfaces that are to be implemented by the software.]*

### Hardware Interfaces

*[This section defines any hardware interfaces that are to be supported by the software, including logical structure, physical addresses, expected behavior, etc. ]*

### Software Interfaces

*[This section describes software interfaces to other components of the software system. These may be purchased components, components reused from another application or components being developed for subsystems outside of the scope of this* ***SRS*** *but with which this software application must interact.]*

### Communications Interfaces

*[Describe any communications interfaces to other systems or devices such as local area networks, remote serial devices, etc.]*

## Licensing Requirements

*[Defines any licensing enforcement requirements or other usage restriction requirements that are to be exhibited by the software.]*

## Legal, Copyright, and Other Notices

*[This section describes any necessary legal disclaimers, warranties, copyright notices, patent notice, wordmark, trademark, or logo compliance issues for the software.]*

## Applicable Standards

*[This section describes by reference any applicable standard and the specific sections of any such standards which apply to the system being described. For example, this could include legal, quality and regulatory standards, industry standards for usability, interoperability, internationalization, operating system compliance, etc.]*

# Supporting Information

*[The supporting information makes the* ***SRS*** *easier to use. It includes:*

*• Table of contents*

*• Index*

*• Appendices*

*These may include use-case storyboards or user-interface prototypes. When appendices are included, the* ***SRS*** *should explicitly state whether or not the appendices are to be considered part of the requirements.]*