BUG TRACKING SYSTEM

Spring 2016 Session

Group 6

Jingwang Teh, jt957

Weicheng Yin, wy157

Luke O’Callaghan, loc105

John Chun-Yu Chan, cyc936

**Table of Contents**

[1.0 BUSINESS CASE 1](#_Toc460949048)

[1.1 BUSINESS CONTEXT 2](#_Toc460949049)

[1.2 USER AND TARGETED AUDIENCE 2](#_Toc460949050)

[1.3 BUSINESS JUSTIFICATION 2](#_Toc460949051)

[2.0 BUG TRACKING SYSTEM SOFTWARE REQUIREMENT SPECIFICATION (SRS) 3](#_Toc460949052)

[2.1 INTRODUCTION 4](#_Toc460949053)

[2.1.1 PURPOSE 4](#_Toc460949054)

[2.1.2 SCOPE 4](#_Toc460949055)

[2.1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS 5](#_Toc460949056)

[2.1.4 REFERENCES 5](#_Toc460949057)

[2.1.5 OVERVIEW 5](#_Toc460949058)

[2.2 OVERALL DESCRIPTION 6](#_Toc460949059)

[2.2.1 PRODUCT PERSPECTIVE 6](#_Toc460949060)

[2.2.2 PRODUCT FUNCTIONS 8](#_Toc460949061)

[2.2.3 USER CHARACTERISTICS 9](#_Toc460949062)

[2.2.4 CONSTRAINTS 9](#_Toc460949063)

[2.2.5 ASSUMPTIONS AND DEPENDENCIES 9](#_Toc460949064)

[2.3 SPECIFIC REQUIREMENTS 10](#_Toc460949065)

[2.3.1 FUNCTIONAL REQUIREMENTS 10](#_Toc460949066)

[2.3.2 NON-FUNCTIONAL REQUIREMENTS 28](#_Toc460949067)

[2.3.3 USE-CASE SPECIFICATIONS 37](#_Toc460949068)

[2.3.4 FUNCTIONAL REQUIREMENTS AND USE CASES MATRIX 55](#_Toc460949069)

[3.0 DOMAIN MODEL 57](#_Toc460949070)

[4.0 PROJECT MANAGEMENT 58](#_Toc460949071)

[PROJECT COST 59](#_Toc460949072)

[PROJECT SCHEDULE 60](#_Toc460949073)

[PROJECT RISK ASSESSMENT 62](#_Toc460949074)

[5.0 PROJECT PLAN FOR INCEPTION, ELABORATION, AND CONSTRUCTION PHASE 65](#_Toc460949075)

[INCEPTION PHASE 66](#_Toc460949076)

[ELABORATION PHASE 66](#_Toc460949077)

[CONSTRUCTION PHASE 67](#_Toc460949078)

[GROUP STRUCTURE, ROLE, AND ARTEFACTS 69](#_Toc460949079)

[GROUP STRUCTURE AND ROLE 70](#_Toc460949080)

[INITIAL PROJECT PLAN AND ARTEFACTS DELIVERED 71](#_Toc460949081)

[ITERATION PLAN 72](#_Toc460949082)

[PROJECT MEETING 74](#_Toc460949083)

[Week 3 Meeting #1 - 12 August 2016 75](#_Toc460949084)

[Summary 75](#_Toc460949085)

[Agenda 76](#_Toc460949086)

[Report 76](#_Toc460949087)

[Week 4 Meeting #2 - 19 August 2016 77](#_Toc460949088)

[Summary 77](#_Toc460949089)

[Agenda 77](#_Toc460949090)

[Report 77](#_Toc460949091)

[Week 5 Meeting #3 - 26 August 2016 78](#_Toc460949092)

[Summary 78](#_Toc460949093)

[Agenda 78](#_Toc460949094)

[Report 78](#_Toc460949095)

[Week 6 Meeting #4 - 2 September 2016 79](#_Toc460949096)

[Summary 79](#_Toc460949097)

[Agenda 79](#_Toc460949098)

[Report 79](#_Toc460949099)

[WORK DIARIES 80](#_Toc460949100)

[Work Diary by Jingwang Teh 81](#_Toc460949101)

[Work Diary by Weicheng Yin 86](#_Toc460949102)

[Work Diary by Luke O’Callaghan 88](#_Toc460949103)

[Work Diary by John Chun-Yu Chan 90](#_Toc460949104)

[EVIDENCE OF VERSION CONTROL 93](#_Toc460949105)

[MEMBERS 93](#_Toc460949106)

[COMMITS 93](#_Toc460949107)

[PULL REQUESTS 94](#_Toc460949108)

[APPENDICES 95](#_Toc460949109)

[LAB Q&A 96](#_Toc460949110)

[Week 2 Q&A with Tutor in Lab 96](#_Toc460949111)

[Week 3 Q&A with Tutor in Lab 98](#_Toc460949112)

[Week 4 Q&A with Tutor in Lab 100](#_Toc460949113)

[Week 6 Q&A with Tutor in Lab 104](#_Toc460949114)

[Week 7 Q&A with Tutor in Lab 106](#_Toc460949115)

[MEMBER CONTRIBUTION ASSESSMENT 107](#_Toc460949116)

# 1.0 BUSINESS CASE

## 

## 1.1 BUSINESS CONTEXT

The case of this project is to design a bug tracking system that has features similar to that of the bug tracking system "Bugzilla", developed by Mozilla. Our intention is to develop a web based system, where users may browse, report, and comment on bugs, as well as giving companies using this software a way to assign bugs to their developers, and a way for patches to be reviewed before they are merged to the main software product.

## 1.2 USER AND TARGETED AUDIENCE

The company that we are developing this software system for has the intention to use this system as a blank slate, which will then be sold to other companies that require a bug tracking system for their own software products. We shall work closely with the client to ascertain details for this project that aren't covered in the project outline, and develop models to assist in understanding the project development schedule.

Essentially there is a duality in the purpose this product serves; firstly, for the client whom we are developing the system for, but also for the clients of our client who will be using this bug tracking system for their own internal or public projects.

## 1.3 BUSINESS JUSTIFICATION

This system will then allow the company who is employing our services to make a profit on the software that we are contracted to develop for them. As it is, this software belongs to our customer, and after the project is completed we will only be responsible for maintenance of the system. Furthermore, the clients of our client will be able to utilize our bug tracking system to track the bugs within their software project and increase their productivity by having our system enhance the flow of bug management for their software projects much more efficiently.

Also, our bug tracking system will be independent of the organization’s system of the clients that our client decides to sell the software system to, as our system does not integrate into any existing system but to support it separately by submitting bug reports and going through the bug life cycle using the bug tracking system. Therefore, the integration of our system has minimal impact on their software project or system.

# 2.0 BUG TRACKING SYSTEM SOFTWARE REQUIREMENT SPECIFICATION (SRS)

## 

## 2.1 INTRODUCTION

### 2.1.1 PURPOSE

This document provides an overall description of the bug tracking system, the intended audience, and design constraints that comes with developing this product. It also describes the functionality of the product as well as its non-functional requirements, and other possible factors necessary to understand the domain of the software system that we are to develop.

### 2.1.2 SCOPE

The software system to be developed is a bug tracking system aimed to be produced and sold by the client to a third party organization.

This bug tracking system will serve a general category of users, as specified by the client, regardless of third-party organization that the system is being sold to, such as:

* Public User
* Bug Reporter, Triage, Developer, and Reviewer
* Administrator

The public user can participate with the system without the need to create an account, and is to be provided with the basic features that all users would have access to, such as: searching for bug reports, browsing categories of bug reports, and viewing system generated reports accessible by all. The public user is restricted to read-only privilege.

Users like bug reporter, triager, developer, and reviewer are involved with the general flow of bug processing tasks within the bug tracking system, where all tasks associated with the general flow are to be supported by our system. The flow of bug processing task starts from bug reporting and bug assignment, to bug solving, bug review and assessment, and then finally bug report completion.

The purpose of the administrator is to manage the users and user groups of the system, and managing the factors for determining reputation to be given for bug reports and fixes.

In order to better manage the functions that the systems provide and to cater to the different types of users listed, the bug tracking system is divided into several correlated modules or subsystems:

* Account Management System manages user profiles of users registered with the bug tracking system
* User Management System manages user group and their read/write access, and is used by the user administrator
* Bug Reporting and Transfer System manages bug submission, bug description, bug status, and user comments by bug reporters, and allows Triage to assign the bug to the respective developers
* Bug Management and Review System manages work distribution and communication among developers in solving a bug, and allows the bug to be submitted to the bug reviewer for review
* Search System manages search query by users
* Report Generation System generates statistical reports based on number of bugs reported and solved
* Reputation System that manages reputations for bug reporters and developers

### 2.1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

|  |  |
| --- | --- |
| **DEFINITIONS** | |
| Bug | An error in the system that can be tracked by the bug tracking system. |
| Patch | A code fix for the errors in the system. |
| **ACRONYMS AND ABBREVIATIONS** | |
| SRS | SRS stands for Software Requirement Specification, which is a document that highlights the functions that the bug tracking system should perform in a given scenario. |
| RUP | RUP stands for Rational Unified Process, which consists of inception, elaboration, construction, and transition phase. However, transition phase is not included for this project, and currently, only the inception phase is necessary. |

### 2.1.4 REFERENCES

1. SRS Sample - SoftwareRequirementsSpecification.pdf

### 2.1.5 OVERVIEW

The rest of the SRS is divided into 2 main sections:

* Overall Description section - describes general factors affecting the development of the system and its requirements
* Specific Requirements section - consists of all software requirements that is required of the software system in order to satisfy the users’ needs.

## 2.2 OVERALL DESCRIPTION

### 2.2.1 PRODUCT PERSPECTIVE

Currently, there are organizations with various software products, which all have the potential to introduce bugs that could simply be an easy-to-fix nuisance or it could be something critical to the core operations of the system itself. However, not many organizations consider the need to implement a way to track and manage bugs related to their system,

Therefore, it has come to a point where the need for a bug tracking system for any organization focused on software products, whether it be a start-up or a medium-sized organization, is necessary in order to effectively assist them in managing bugs to maintain their software product, as it would be more beneficial and cost saving in the long run.

Our bug tracking systems aims to provide a solution that is cost efficient and also contain functionality of a bug tracking system without the dependency on software product type, or organizational size.

#### 2.2.1.1 SYSTEM INTERFACES

Our bug tracking system is to be developed as a web-based application that can be to run at localhost to provide bug related services.

The clients can log into the system from the PC which also act as a server and then interact with the system and provides basic services such as login, and search, as well as bug life cycle related services, like reporting bugs, commenting, triaging bugs, fixing bugs and reviewing bugs. Administrator functionalities are also accessible as long as the account is in administrator user group.

The Web Server also runs in the same PC, which is accessible by the client. The client can login to the web server, where the server can accept http requests, query the system database, then return and display the corresponding web page back, according to the client’s requirements.

The database in the same PC, is used to store the system’s user profiles and the bugs’ information.

#### 2.2.1.2 USER INTERFACES

The user interfaces provided to client will be a series of web pages. The homepage is straightforward and minimalistic, with a search bar for searching bugs in the middle of the web page and the navigation bar at the top. Users could login, navigate to the other web pages by clicking on the icon in the navigation bar.

The administrator can also login into the system via the web page like other users but have higher access to the system. The login page for the administrator will be different compared to the login page of a normal user. It will display a table which contains a list of users and their general profile information associated with it for the administrator to better manage the site’s user base.

The database we use is MySQL, and all accesses to the database is performed through GUIs provided by the system.

#### 2.2.1.3 HARDWARE INTERFACES

All components of the bug tracking system should be executable using a personal computer with web functionality as the interface.

#### 2.2.1.4 SOFTWARE INTERFACES

##### 2.2.1.4.1 USER INTERFACE

The user interacts with the bug tracking system through the use of a web browser.

The system should support Google Chrome 52.0.2743.116 or above.

#### 2.2.1.5 COMMUNICATION INTERFACES

Both the web server and the database server should be located in the same PC.

#### 2.2.1.6 OPERATIONS

The bug reporting functions should be easy for all users to use, as long as they have basic knowledge on using the Web Browser, and no extra software or tools should be required.

The web server should be easy to manage, and maintenance should not be required from the client as only the technicians who developed the system are required to maintain it.

The database should be able to import the data provided by the client.

### 2.2.2 PRODUCT FUNCTIONS

The two main function of the bug tracking system is to allow users to submit bug reports for tracking, and provide support for the management of bug life cycle from the start to the end.

For public users, the bug tracking system provides basic functionality accessible by the public through the following basic functions:

* Browse bug reports. Each bug report is separated in its own category, which can be browsed individually.
* Search bug reports. Bug reports can be searched with keywords, such as bug report name and category.
* View statistical reports. Statistical report highlights the performance of bug reporters and developers in terms of bug reports being solved through the system.

For bug reporters, the system helps them to report bugs and contribute to bug reports through the following functions:

* Submit bug report.
* Commenting on bug report.
* Earn reputation by submitting good bug reports.

For triagers, developers, and reviewers, the system helps them to participate in managing bug reports from the creation of the bug report to its closing, through the following function:

* Bug report assignment. Triagers can assign bug reports to developers in order to focus on the bug that needs to be solved.
* Developer comments. Developers can contribute to bug reports through the use of comments in bug reports, in regards to the progress of the bug being fixed.
* Bug report reassignment or closure. This allows reviewer to determine the status of the bug, whether to reassign it as it was not solved, or close it as it has been solved.

Lastly, the various user groups in the system is managed by the administrator through the following functions:

* Create user accounts
* Change user roles. This allows the user administrator to assign the user to various roles, such as triager, developer, and reviewer.
* Deactivate user account.

All functionalities are included based on the business needs of a bug tracking system, to allow a more convenient way of tracking bugs through bug reports and managing bug life cycle.

### 2.2.3 USER CHARACTERISTICS

* **Bug Reporters** are people who uses their system and have enough understanding of the bug at hand to produce a bug report.
* **Triagers** know the developer teams that are available and have the technical expertise to determine the characteristics of a bug report, which allows better management of bug reports, and assigning them to the correct developer to solve the bug report at hand.
* **Developers** are knowledgeable in the structure of their own system and are capable of producing a fix according to the bug report given.
* **Reviewers** are aware of both the triager and developer in the company, are knowledgeable of the system components, and capable of isolating system tests according to bug reports.
* **Administrators** have higher responsibility and also are knowledgeable about the company’s human resources and are capable of managing their system profiles by assigning the appropriate roles to the staff.

### 2.2.4 CONSTRAINTS

The system should strictly be implemented according to the constraints as specified below:

* Account Authentication: All bug reporting and bug management should be done through a registered account.
* Access Control: Each type of user should only have access to the appropriate interface associated with the tasks that they are allowed to do, for example, only the administrators can change the role of a user.
* Programming Language: The system should only be developed using an Object-oriented language, such as C++ or Java.
* Database and Web Server Location: The system database and web server should be implemented and stored locally in the same computer.
* The system must be developed before week 13 of Spring 2016 session.

### 2.2.5 ASSUMPTIONS AND DEPENDENCIES

All potential users of the bug tracking system have an email address, and have access to a computer with web capabilities to access the website.

## 2.3 SPECIFIC REQUIREMENTS

### 2.3.1 FUNCTIONAL REQUIREMENTS

All requirements specified (both functional and non-functional) of the system is ranked based on its degree of importance.

1. **Critical** : Highest level of importance. Requirements with this rank are those that reflect the core functionalities of the system and must be implemented in early stages.

2. **Essential** : Medium level of significance. Essential requirements reflect important functionalities and should be added after critical requirements have been implemented.

3. **Optional** : Lowest level of importance. These are stretched goals that the development team has set out. These requirements are to be fulfilled when critical and essential levels of functionalities have been satisfied.

#### 2.3.1.1 Account Management System

This section includes the functionalities that our system provides to the public user and registered user for managing their user account and profile.

|  |  |  |
| --- | --- | --- |
| **Registration Function** | | |
| ***Requirement #:* #F\_2.3.1.1\_1** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_1 |
| ***Description:*** Allow the public user to sign-up to the system. The user need to sign-up with the following information:   * Username * E-mail address * Password | | |
| ***Rationale:*** Allow new user to sign-up to the system to access other functions | | |
| ***Source:*** Client | | |
| ***Fit Criterion:*** User become registered user to access certain functions | | |
| ***Dependencies:*** User must have a valid email address for registration | | |
| ***Rank of Importance:*** Critical | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

|  |  |  |
| --- | --- | --- |
| **Login** | | |
| ***Requirement #:* #F\_2.3.1.1\_2** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_1 |
| ***Description:*** Allow user to login into the system to access other functions based on their user type. They need the following information to log in   * User name * Password | | |
| ***Rationale:*** User wants to login to the system to access other functions | | |
| ***Source:*** Client | | |
| ***Fit Criterion:*** A user is successfully logged in into the system | | |
| ***Dependencies:*** User must have a valid user account | | |
| ***Rank of Importance:*** Critical | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

|  |  |  |
| --- | --- | --- |
| **Logout** | | |
| ***Requirement #:* #F\_2.3.1.1\_3** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_1 |
| ***Description:*** Allow the user to logout from the system | | |
| ***Rationale:*** User wants to logout from the system | | |
| ***Source:*** Client | | |
| ***Fit Criterion:*** A user is successfully logged out from the system | | |
| ***Dependencies:*** User must be logged in | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

|  |  |  |
| --- | --- | --- |
| **Edit details** | | |
| ***Requirement #:* #F\_2.3.1.1\_4** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_1 |
| ***Description:*** Allow user to edit their own personal details on profile. The details include:   * E-mail * Password | | |
| ***Rationale:*** User wants to edit their information | | |
| ***Source:*** Client | | |
| ***Fit Criterion:*** A account is successfully edited | | |
| ***Dependencies:*** The account must exist | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

#### 2.3.1.2 User Management System

This section includes the functionalities that the system provides to the administrator.

|  |  |  |
| --- | --- | --- |
| **Admin login** | | |
| ***Requirement #:* #F\_2.3.1.2\_1** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_1 |
| ***Description:*** Allow administrator to login to the system. For security reason, the admin login page will be different from the user login page | | |
| ***Rationale:*** Administrator wants to login to the system | | |
| ***Source:*** Administrator | | |
| ***Fit Criterion:*** An administrator is successfully logged in to the system | | |
| ***Dependencies:*** Administrator must have a admin type account | | |
| ***Rank of Importance:*** Critical | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

|  |  |  |
| --- | --- | --- |
| **Deactivate or Activate user account** | | |
| ***Requirement #:* #F\_2.3.1.2\_2** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_5 |
| ***Description:*** Allow administrator to deactivate or activate a user account. Once the account has been deactivated, it cannot be logged in until the admin reactivated it. | | |
| ***Rationale:*** Administrator wants to deactivate or activate a user account | | |
| ***Source:*** Administrator | | |
| ***Fit Criterion:*** An administrator is successfully deactivated or activated a user account | | |
| ***Dependencies:*** The target user account must exist | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

|  |  |  |
| --- | --- | --- |
| **Create user account** | | |
| ***Requirement #:* #F\_2.3.1.2\_3** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_5 |
| ***Description:*** Allow administrator to create a user account | | |
| ***Rationale:*** Administrator wants to create a user account | | |
| ***Source:*** Administrator | | |
| ***Fit Criterion:*** An administrator successfully creates a user account | | |
| ***Dependencies:*** The target user account must exist | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

|  |  |  |
| --- | --- | --- |
| **Change user role** | | |
| ***Requirement #:* #F\_2.3.1.2\_4** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_5 |
| ***Description:*** Allow administrator to change the user role. There are four types of user the admin can change   * Developer * Bug reporter * Triager * Reviewer | | |
| ***Rationale:*** Administrator wants to change the user role | | |
| ***Source:*** Administrator | | |
| ***Fit Criterion:*** An administrator is successfully change the user role | | |
| ***Dependencies:*** The target user account must exist | | |
| ***Rank of Importance:*** Critical | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

|  |  |  |
| --- | --- | --- |
| **View user list** | | |
| ***Requirement #:* #F\_2.3.1.2\_5** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_5 |
| ***Description:*** Allow administrator to view all the registered user in a list. The user list will provide a filter for quick search | | |
| ***Rationale:*** Administrator wants to view the user | | |
| ***Source:*** Administrator | | |
| ***Fit Criterion:*** An administrator is successfully view the user of the system | | |
| ***Dependencies:*** The admin must login | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

|  |  |  |
| --- | --- | --- |
| **Track assigned work** | | |
| ***Requirement #:* #F\_2.3.1.2\_6** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_5 |
| ***Description:*** Allow administrator to track the status of the work that assigned to each developer | | |
| ***Rationale:*** Administrator wants to track the work | | |
| ***Source:*** Administrator | | |
| ***Fit Criterion:*** An administrator is successfully view track list | | |
| ***Dependencies:*** There need to be a work that are assigned to a developer | | |
| ***Rank of Importance:*** Critical | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

|  |  |  |
| --- | --- | --- |
| **Change factor** | | |
| ***Requirement #:* #F\_2.3.1.2\_7** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_5 |
| ***Description:*** Allow administrator to change the factor for determining the reputation | | |
| ***Rationale:*** Administrator wants to change the factor | | |
| ***Source:*** Administrator | | |
| ***Fit Criterion:*** An administrator is successfully change the factor | | |
| ***Dependencies:*** The administrator need to be logged in | | |
| ***Rank of Importance:*** Critical | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

#### 2.3.1.3 Bug Reporting and Transfer System

This section includes the functionalities that the system provided to the Bug reporter and the Triager. Also a comment function is provided to all user type except the public user

|  |  |  |
| --- | --- | --- |
| **Report bug** | | |
| ***Requirement #:* #F\_2.3.1.3\_1** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_1 |
| ***Description:*** Allow bug reporter to report a bug from the code. They report the bug in a form provide by the system. The following information needed to be included   * Title * Bugs contain in the code | | |
| ***Rationale:*** Bug reporter wants to report a bug | | |
| ***Source:*** Bug reporter | | |
| ***Fit Criterion:*** Bug reporter successfully report a bug | | |
| ***Dependencies:*** The bug reporter need to login to the system | | |
| ***Rank of Importance:*** Critical | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

|  |  |  |
| --- | --- | --- |
| **Comment on bug** | | |
| ***Requirement #:* #F\_2.3.1.3\_2** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_1 |
| ***Description:*** Allow all logged in user to comment on a bug for discussion | | |
| ***Rationale:*** Logged in user wants to comment on a bug | | |
| ***Source:*** Logged in user | | |
| ***Fit Criterion:*** A comment is successfully posted | | |
| ***Dependencies:*** The user need to login to the system before they comment on a bug | | |
| ***Rank of Importance:*** Critical | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

|  |  |  |
| --- | --- | --- |
| **Subscribe on bug report** | | |
| ***Requirement #:* #F\_2.3.1.3\_3** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_1 |
| ***Description:*** Allow all logged in user to subscribe to bug report to keep in touch on it | | |
| ***Rationale:*** Logged in user wants to subscribe on a bug | | |
| ***Source:*** Logged in user | | |
| ***Fit Criterion:*** A reported bug is successfully subscribed | | |
| ***Dependencies:*** The user need to login to the system before they subscribe on a bug | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** N/A | | |

|  |  |  |
| --- | --- | --- |
| **Manage bug** | | |
| ***Requirement #:* #F\_2.3.1.3\_4** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_2 |
| ***Description:*** Allow triager to manage the quality aspect of a bug. Its including the following aspect:   * To check if the bug is invalid * To check if the bug is duplicated | | |
| ***Rationale:*** triager wants to manage the quality aspect a bug | | |
| ***Source:*** triager | | |
| ***Fit Criterion:*** The triager have managed the bug | | |
| ***Dependencies:*** The reported bug must exist | | |
| ***Rank of Importance:*** Critical | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

|  |  |  |
| --- | --- | --- |
| **Check Bug if Invalid** | | |
| ***Requirement #:* #F\_2.3.1.3\_5** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_2 |
| ***Description:*** Allow triager to check whether the bug is valid or not | | |
| ***Rationale:*** Triager wants to check whether the bug is valid or not | | |
| ***Source:*** Triager | | |
| ***Fit Criterion:*** The triager have validated the bug | | |
| ***Dependencies:*** The triager need to implement the requirement **#F\_#2.3.1.3\_3** | | |
| ***Rank of Importance:*** Critical | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

|  |  |  |
| --- | --- | --- |
| **Check Bug if Duplicated** | | |
| ***Requirement #:* #F\_2.3.1.3\_6** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_2 |
| ***Description:*** Allow triager to check whether the bug is duplicated or not | | |
| ***Rationale:*** triager wants to check whether the bug is duplicated or not | | |
| ***Source:*** triager | | |
| ***Fit Criterion:*** The triager have checked whether the bug is duplicated or not | | |
| ***Dependencies:*** The triager need to implement the requirement **#F\_#2.3.1.3\_3** | | |
| ***Rank of Importance:*** Critical | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

|  |  |  |
| --- | --- | --- |
| **Merge bug** | | |
| ***Requirement #:* #F\_2.3.1.3\_7** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_2 |
| ***Description:*** Allow triager to merge the duplicated bug | | |
| ***Rationale:*** Triager wants to merge the duplicated bug | | |
| ***Source:*** Triager | | |
| ***Fit Criterion:*** The triager have merge the duplicated bug | | |
| ***Dependencies:*** The triager need to implement the requirement **#F\_#2.3.1.3\_3** | | |
| ***Rank of Importance:*** Critical | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

|  |  |  |
| --- | --- | --- |
| **View reported bug** | | |
| ***Requirement #:* #F\_2.3.1.3\_8** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_2 |
| ***Description:*** Allow triager to review the reported bug, he/she can add technical information on it so that the developer can be more easy to work on it | | |
| ***Rationale:*** Triager wants to review the reported bug | | |
| ***Source:*** Triager | | |
| ***Fit Criterion:*** The triager have review the reported bug | | |
| ***Dependencies:*** There is a bug that are reported by the reporter | | |
| ***Rank of Importance:*** Critical | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

|  |  |  |
| --- | --- | --- |
| **Receive unsolved bug** | | |
| ***Requirement #:* #F\_2.3.1.3\_9** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_2 |
| ***Description:*** Allow triager to receive the unsolved bug by the reviewer | | |
| ***Rationale:*** Triager wants to receive the unsolved bug | | |
| ***Source:*** Triager | | |
| ***Fit Criterion:*** The triager have received the unsolved bug | | |
| ***Dependencies:*** There is a bug that are transferred by the reviewer | | |
| ***Rank of Importance:*** Critical | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

#### 2.3.1.4 Bug Management and Review System

This section include the functionalities of the system provided to the developer and the reviewer

|  |  |  |
| --- | --- | --- |
| **Receive bug** | | |
| ***Requirement #:* #F\_2.3.1.4\_1** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_3 |
| ***Description:*** Allow developer to receive the bug which is assigned by the triager | | |
| ***Rationale:*** Developer wants to receive the bug | | |
| ***Source:*** Developer | | |
| ***Fit Criterion:*** The developer has received the bug | | |
| ***Dependencies:*** There is a bug that are assigned by the triager | | |
| ***Rank of Importance:*** Critical | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

|  |  |  |
| --- | --- | --- |
| **Attach patch to comment** | | |
| ***Requirement #:* #F\_2.3.1.4\_2** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_3 |
| ***Description:*** Allow developer to attach the patch on the comment, then the patch will be forwarded to the reviewer to review. The status of the bug will be updated to “Under Review” | | |
| ***Rationale:*** Developer wants to upload the patch | | |
| ***Source:*** Developer | | |
| ***Fit Criterion:*** The developer has uploaded the bug | | |
| ***Dependencies:*** There is a bug that are fixed by the developer | | |
| ***Rank of Importance:*** Critical | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

|  |  |  |
| --- | --- | --- |
| **Review uploaded patch** | | |
| ***Requirement #:* #F\_2.3.1.4\_3** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_4 |
| ***Description:*** Allow reviewer to review the patch the provided by the developer | | |
| ***Rationale:*** Reviewer wants to review the patch | | |
| ***Source:*** Reviewer | | |
| ***Fit Criterion:*** The reviewer has reviewed the patch | | |
| ***Dependencies:*** There is a patch that are uploaded by the developer | | |
| ***Rank of Importance:*** Critical | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

|  |  |  |
| --- | --- | --- |
| **Transfer patch to Triager if bug is unsolved** | | |
| ***Requirement #:* #F\_2.3.1.4\_4** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_4 |
| ***Description:*** Allow reviewer to transfer the patch to the triager if the bug have not been solved | | |
| ***Rationale:*** Reviewer wants to transfer the patch | | |
| ***Source:*** Reviewer | | |
| ***Fit Criterion:*** The reviewer has transferred the patch | | |
| ***Dependencies:*** There is a patch that are uploaded by the developer | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

#### 2.3.1.5 Search System

This section include the functionalities of the system provided to all the user in the system to search the bug or browse the bug

|  |  |  |
| --- | --- | --- |
| **Search bug** | | |
| ***Requirement #:* #F\_2.3.1.5\_1** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_1 |
| ***Description:*** Allow any users to search for specific bug by keywords | | |
| ***Rationale:*** Any users want to search for a bug | | |
| ***Source:*** Any users | | |
| ***Fit Criterion:*** The user has search for a bug | | |
| ***Dependencies:*** There is a bug that can be searched | | |
| ***Rank of Importance:*** Critical | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

|  |  |  |
| --- | --- | --- |
| **Browse bug** | | |
| ***Requirement #:* #F\_2.3.1.5\_2** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_1 |
| ***Description:*** Allow any users to browse for a bug | | |
| ***Rationale:*** Any users want to browse for a bug | | |
| ***Source:*** Any users | | |
| ***Fit Criterion:*** The user has browsed for a bug | | |
| ***Dependencies:*** There is a bug that can be browsed | | |
| ***Rank of Importance:*** Critical | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

#### 2.3.1.6 Report Generation System

|  |  |  |
| --- | --- | --- |
| **Report generation** | | |
| ***Requirement #:* #F\_2.3.1.6\_1** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_1 |
| ***Description:*** The system will automatically generate an hourly statistical report. The report included the number of bugs resolved and the best performed reporters and developers, which can be viewed by all users. | | |
| ***Rationale:*** The system will generate the report hourly | | |
| ***Source:*** System | | |
| ***Fit Criterion:*** The report has been generated | | |
| ***Dependencies:*** There is a bug that can be used for the statistical report | | |
| ***Rank of Importance:*** Critical | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

#### 2.3.1.7 Reputation System

|  |  |  |
| --- | --- | --- |
| **Reputation** | | |
| ***Requirement #:* #F\_2.3.1.7\_1** | ***Requirement Type:***  Functional | ***Use Case #:*** #UC\_5 |
| ***Description:*** The reputation is used to determine the performance of a user. It’s automatically calculated by the system based on several factors set by the administrator, when the user submits a bug report, and the developer provides a bug fix | | |
| ***Rationale:*** The reputation system will measure the performance of bug reporters and developers by calculating the reputation automatically to prevent any ambiguity when determining the “correct” reputation | | |
| ***Source:*** System | | |
| ***Fit Criterion:*** The reputation of a user gets calculated and updated correctly according to factors set by administrator | | |
| ***Dependencies:*** There is a target user to receive the reputation | | |
| ***Rank of Importance:*** Critical | | |
| ***Supporting Materials:*** N/A | | |
| ***History:***N/A | | |

### 2.3.2 NON-FUNCTIONAL REQUIREMENTS

|  |  |  |
| --- | --- | --- |
| ***Requirement #:* NREQ\_01** | ***Requirement Type:***  Reliability | ***Use Case #:*** None. |
| ***Description:*** The system should be functional 24 hours a day, and 7 days a week, excluding maintenance, which should take no longer than 1 hour. | | |
| ***Rationale:*** All users want to able to access system functionalities at all times, to avoid loss of productivity from lack of access to bug reports. | | |
| ***Source:*** Public User, Bug Reporter, Triager, Developer, Reviewer, Administrator | | |
| ***Fit Criterion:*** All users can access any system functionalities at all times (24 hours a day, 7 days a week) | | |
| ***Dependencies:*** None | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** None | | |
| ***History:***   * Created by Jingwang Teh - 29 August 2016 | | |

|  |  |  |
| --- | --- | --- |
| ***Requirement #:* NREQ\_02** | ***Requirement Type:***  Performance | ***Use Case #:*** None. |
| ***Description:*** The system should respond to user action within 7 seconds. | | |
| ***Rationale:*** All users want confirmation of their actions to the system | | |
| ***Source:*** Public User, Bug Reporter, Triager, Developer, Reviewer, Administrator | | |
| ***Fit Criterion:*** All | | |
| ***Dependencies:*** None | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** None | | |
| ***History:***   * Created by Jingwang Teh - 29 August 2016 | | |

|  |  |  |
| --- | --- | --- |
| ***Requirement #:* NREQ\_03** | ***Requirement Type:***  Performance | ***Use Case #:*** None. |
| ***Description:*** The system search functionality should return the queried results within 10 seconds. | | |
| ***Rationale:*** All users want to know the results of the search quickly so he can select the result or proceed with another search, instead of waiting for search results to appear. | | |
| ***Source:*** Public User, Bug Reporter, Triager, Developer, Reviewer, Administrator | | |
| ***Fit Criterion:*** Search results should display on screen within 1 minute. | | |
| ***Dependencies:*** None | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** None | | |
| ***History:***   * Created by Weicheng Yin - 29 August 2016 | | |

|  |  |  |
| --- | --- | --- |
| ***Requirement #:* NREQ\_04** | ***Requirement Type:***  Performance | ***Use Case #:*** None. |
| ***Description:*** The system should automatically generate new statistical report every 1 hour. | | |
| ***Rationale:*** The users want to view a report that is always up-to-date. | | |
| ***Source:*** Public User, Bug Reporter, Triager, Developer, Reviewer, Administrator | | |
| ***Fit Criterion:*** Report data displayed on screen changes every 1 hour to reflect the updated data. | | |
| ***Dependencies:*** None | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** None | | |
| ***History:***   * Created by Jingwang Teh - 29 August 2016 | | |

|  |  |  |
| --- | --- | --- |
| ***Requirement #:* NREQ\_05** | ***Requirement Type:***  Efficiency | ***Use Case #:*** None. |
| ***Description:*** The system should support at least 100 user profiles being stored in the database. | | |
| ***Rationale:*** The system needs to be capable of supporting at least 100 users using the system regardless of user group | | |
| ***Source:*** Public User, Bug Reporter, Triager, Developer, Reviewer, Administrator | | |
| ***Fit Criterion:*** The system database is able to store 100 user profiles, and profile data on the user profile screen can be retrieved by the system and displayed to the user within 7 seconds. | | |
| ***Dependencies:*** None | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** None | | |
| ***History:***   * Created by Weicheng Yin - 29 August 2016 | | |

|  |  |  |
| --- | --- | --- |
| ***Requirement #:* NREQ\_06** | ***Requirement Type:***  Efficiency | ***Use Case #:*** None. |
| ***Description:*** The system should support at least 1000 bug reports being stored in the database. | | |
| ***Rationale:*** All users are want to submit more than 10 bug reports each without being restricted to the total number of bug reports they can submit | | |
| ***Source:*** Bug Reporter, Triager, Developer, Reviewer, Administrator | | |
| ***Fit Criterion:*** The number of bug reports in the database exceeds 1000 and searching efficiency to find individual bug reports does not take more than 10 seconds. | | |
| ***Dependencies:*** None | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** None | | |
| ***History:***   * Created by Weicheng Yin - 29 August 2016 | | |

|  |  |  |
| --- | --- | --- |
| ***Requirement #:* NREQ\_07** | ***Requirement Type:***  Usability | ***Use Case #:*** None. |
| ***Description:*** The system should be consistent in all its user interfaces according to the Prototyped User Interface | | |
| ***Rationale:*** To provide the users a better user experience and conformity to existing knowledge about the system’s interface. | | |
| ***Source:*** Public User, Bug Reporter, Triager, Developer, Reviewer, Administrator | | |
| ***Fit Criterion:*** All pages in the system follows the interface provided in the prototyped user interface. | | |
| ***Dependencies:*** None | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** None | | |
| ***History:***   * Created by Jingwang Teh - 29 August 2016 | | |

|  |  |  |
| --- | --- | --- |
| ***Requirement #:* NREQ\_08** | ***Requirement Type:***  Usability | ***Use Case #:*** None. |
| ***Description:*** The system needs to provide users with built-in explanations of system features to help users learn to use the system, and are accessible by all users. | | |
| ***Rationale:*** To allow the users to quickly pick up and use the system. | | |
| ***Source:*** Public User, Bug Reporter, Triager, Developer, Reviewer, Administrator | | |
| ***Fit Criterion:*** The built-in explanations are accessible by all types of users, and explanations are displayed within 5 seconds. | | |
| ***Dependencies:*** None | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** None | | |
| ***History:***   * Created by Jingwang Teh - 29 August 2016 | | |

|  |  |  |
| --- | --- | --- |
| ***Requirement #:* NREQ\_09** | ***Requirement Type:***  Usability | ***Use Case #:*** None. |
| ***Description:*** The system should not require any specific technical knowledge of the system to be able to use it, as long as the users are computer-literate | | |
| ***Rationale:*** The users should not require any training to use the system, so it can be used by any organization that the client decides to sell the system to, without the need to retrain personnel. | | |
| ***Source:*** Public User, Bug Reporter, Triager, Developer, Reviewer, Administrator | | |
| ***Fit Criterion:*** All users are capable of using the system after 1 day of exploring and using the system. | | |
| ***Dependencies:*** None | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** None | | |
| ***History:***   * Created by Jingwang Teh - 29 August 2016 | | |

|  |  |  |
| --- | --- | --- |
| ***Requirement #:* NREQ\_09** | ***Requirement Type:***  Usability | ***Use Case #:*** None. |
| ***Description:*** The system should use English in all its User Interfaces. | | |
| ***Rationale:*** All users want to utilize their main communication language between each other while working in an organizational environment when using the system. | | |
| ***Source:*** Public User, Bug Reporter, Triager, Developer, Reviewer, Administrator | | |
| ***Fit Criterion:*** The system user interfaces are all in English. | | |
| ***Dependencies:*** None | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** None | | |
| ***History:***   * Created by Jingwang Teh - 29 August 2016 | | |

|  |  |  |
| --- | --- | --- |
| ***Requirement #:* NREQ\_10** | ***Requirement Type:***  Usability | ***Use Case #:*** None. |
| ***Description:*** The system should follow the W3C standards for its HTML front-end files. | | |
| ***Rationale:*** To conform with the international standards. | | |
| ***Source:*** Public User, Bug Reporter, Triager, Developer, Reviewer, Administrator | | |
| ***Fit Criterion:*** The html front-end files are all validated by W3C mark-up validator and have 0 errors. | | |
| ***Dependencies:*** None | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** None | | |
| ***History:***   * Created by Jingwang Teh - 29 August 2016 | | |

|  |  |  |
| --- | --- | --- |
| ***Requirement #:* NREQ\_11** | ***Requirement Type:***  Usability | ***Use Case #:*** None. |
| ***Description:*** The system should not require any specific technical knowledge of the system to be able to use it, as long as the users are computer-literate | | |
| ***Rationale:*** The users should not require any training to use the system, so it can be used by any organization that the client decides to sell the system to, without the need to retrain personnel. | | |
| ***Source:*** Public User, Bug Reporter, Triager, Developer, Reviewer, Administrator | | |
| ***Fit Criterion:*** All users are capable of using the system after 1 day of exploring and using the system. | | |
| ***Dependencies:*** None | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** None | | |
| ***History:***   * Created by Jingwang Teh - 29 August 2016 | | |

|  |  |  |
| --- | --- | --- |
| ***Requirement #:* NREQ\_12** | ***Requirement Type:***  Reliability | ***Use Case #:*** None. |
| ***Description:*** The system should be able to recover from system failure within 24 hours. | | |
| ***Rationale:*** The downtime of the system must be minimized to ensure minimal impact on organization’s productivity during the downtime. | | |
| ***Source:*** Public User, Bug Reporter, Triager, Developer, Reviewer, Administrator | | |
| ***Fit Criterion:*** The system failure is resolved within 24 hours. | | |
| ***Dependencies:*** None | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** None | | |
| ***History:***   * Created by Jingwang Teh - 29 August 2016 | | |

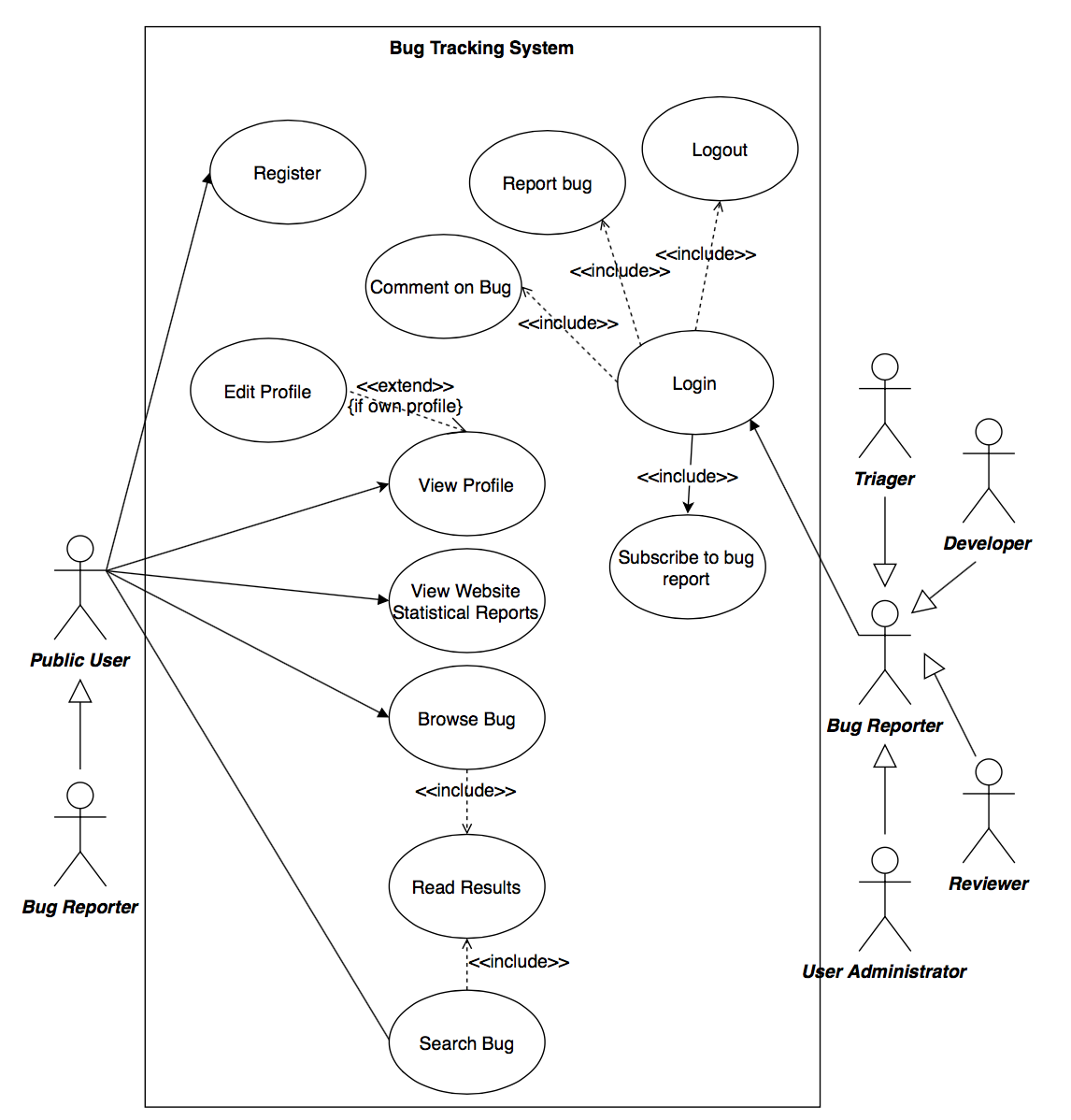
|  |  |  |
| --- | --- | --- |
| ***Requirement #:* NREQ\_13** | ***Requirement Type:***  Reliability | ***Use Case #:*** None. |
| ***Description:*** The system should support up to 50 users at any time. | | |
| ***Rationale:*** The system should be capable of withstanding the load from as many as 50 users interacting with the system. | | |
| ***Source:*** Public User, Bug Reporter, Triager, Developer, Reviewer, Administrator | | |
| ***Fit Criterion:*** The system allows 50 concurrent users at any time. | | |
| ***Dependencies:*** None | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** None | | |
| ***History:***   * Created by Jingwang Teh - 29 August 2016 | | |

|  |  |  |
| --- | --- | --- |
| ***Requirement #:* NREQ\_14** | ***Requirement Type:***  Security | ***Use Case #:*** None. |
| ***Description:*** The system needs to log the users out after 15 minutes of idling. | | |
| ***Rationale:*** To prevent users from getting their accounts accessed by unauthorized individuals in their proximity when they are not around. | | |
| ***Source:*** Public User, Bug Reporter, Triager, Developer, Reviewer, Administrator | | |
| ***Fit Criterion:*** The system logs the user out after 15 minutes of idling. | | |
| ***Dependencies:*** None | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** None | | |
| ***History:***   * Created by Weicheng Yin - 29 August 2016 | | |

|  |  |  |
| --- | --- | --- |
| ***Requirement #:* NREQ\_15** | ***Requirement Type:***  Performance | ***Use Case #:*** None. |
| ***Description:*** The system should be able to calculate an appropriate reputation for bug reporters and developers within 7 seconds of submitting the bug report or bug fixes | | |
| ***Rationale:*** The bug reporter and developer wants to know their performance for their submission of bug report or fix without waiting for too long. | | |
| ***Source:*** Bug Reporter, Developer | | |
| ***Fit Criterion:*** The system is able to calculate the reputation for bug reporter and developer within 7 seconds | | |
| ***Dependencies:*** None | | |
| ***Rank of Importance:*** Essential | | |
| ***Supporting Materials:*** None | | |
| ***History:***   * Created by Weicheng Yin - 29 August 2016 | | |

### 2.3.3 USE-CASE SPECIFICATIONS

#### 2.3.3.1 Public User and Bug Reporter



Use case: #UC\_1

**Description**

The public users only have the read-only permission in our system. The Bug Reporter have the lowest permission level of the registered user in our system. It is inherited by the Triager, Developer, Reviewer. They have different permissions in other system for particular duties. The administrator is responsible for maintaining the system

|  |  |
| --- | --- |
| ***Name:*** Register | ***ID:*** #PUBLIC\_1 |
| ***Stakeholders and goals:*** Allow public user to sign-up to the system | |
| ***Description:*** Public user can sign-up to the system | |
| ***Actors:*** Public user | |
| ***Triggers:*** Public user want to sign-up to the system | |
| ***Normal flow:***   1. Click “Sign-up” on the top right corner 2. System prompt user input his username, email address and password 3. User provides username, email address, and password 4. System verifies the information 5. Done | |
| ***Sub-flows:*** N/A | |
| ***Alternative/Exceptional flows:*** N/A | |

|  |  |
| --- | --- |
| ***Name:*** View profile | ***ID:*** #PUBLIC\_2 |
| ***Stakeholders and goals:*** Allow public user to view user profile | |
| ***Description:*** Public user can view user’s profile | |
| ***Actors:*** Public user | |
| ***Triggers:*** Public user want to view user’s profile | |
| ***Normal flow:***   1. Search user name 2. Click the user’s username in the user list 3. View user profile 4. Done | |
| ***Sub-flows:*** N/A | |
| ***Alternative/Exceptional flows:*** N/A | |

|  |  |
| --- | --- |
| ***Name:*** Browse bug reports | ***ID:*** #PUBLIC\_3 |
| ***Stakeholders and goals:*** Allow public user to browse the bug | |
| ***Description:*** Public user can browse the bug | |
| ***Actors:*** Public user | |
| ***Triggers:*** Public user want to browse bug | |
| ***Normal flow:***   1. Click “Browse” on the top menu 2. View the list of the result 3. Done | |
| ***Sub-flows:*** N/A | |
| ***Alternative/Exceptional flows:*** N/A | |

|  |  |
| --- | --- |
| ***Name:*** Search bug | ***ID:*** #PUBLIC\_4 |
| ***Stakeholders and goals:*** Allow public user to search bug | |
| ***Description:*** Public user can search for specific bug by keywords | |
| ***Actors:*** Public user | |
| ***Triggers:*** Public user want to search for specific bug | |
| ***Normal flow:***   1. Input keywords in the search bar then click “Search” button 2. System returns the related results 3. Done | |
| ***Sub-flows:*** N/A | |
| ***Alternative/Exceptional flows:*** N/A | |

|  |  |
| --- | --- |
| ***Name:*** View the statistical reports | ***ID:*** #PUBLIC\_5 |
| ***Stakeholders and goals:*** Allow any user to view the statistical reports | |
| ***Description:*** Any user can view the statistical reports, its include the number of bugs solved per week and the best performed reporters and the developer | |
| ***Actors:*** Any user | |
| ***Triggers:*** User want to view the statistical reports | |
| ***Normal flow:***   1. Click “View statistical” 2. View the statistic | |
| ***Sub-flows:*** N/A | |
| ***Alternative/Exceptional flows:*** N/A | |

|  |  |
| --- | --- |
| ***Name:*** Subscribe to bug report | ***ID:*** #PUBLIC\_6 |
| ***Stakeholders and goals:*** Allow bug reporter to subscribe to bug report | |
| ***Description:***All registered user can subscribe to bug report for keep in touch on the bug report | |
| ***Actors:*** Any user | |
| ***Triggers:*** The bug report are subscribed by user | |
| ***Normal flow:***   1. View the bug 2. Click “subscribe” | |
| ***Sub-flows:*** N/A | |
| ***Alternative/Exceptional flows:*** N/A | |

|  |  |
| --- | --- |
| ***Name:*** Log in | ***ID:*** #REPORT\_1 |
| ***Stakeholders and goals:*** Allow registered user to login to the system | |
| ***Description:*** Registered user can login to the system | |
| ***Actors:*** Registered user | |
| ***Triggers:*** Registered user want to login to the system | |
| ***Normal flow:***   1. Click “Login” 2. System prompt user to input username and password 3. System verifies username and password 4. Done | |
| ***Sub-flows:*** N/A | |
| ***Alternative/Exceptional flows:*** N/A | |

|  |  |
| --- | --- |
| ***Name:*** View profile | ***ID:*** #REPORT\_2 |
| ***Stakeholders and goals:*** Allow user to view user profile | |
| ***Description:*** User can view user’s profile | |
| ***Actors:*** Registered user | |
| ***Triggers:*** Registered user want to view user’s profile | |
| ***Normal flow:***   1. Click [username] on the top right corner (For his own profile) 2. Done | |
| ***Sub-flows:*** N/A | |
| ***Alternative flow: (View others profile)***   1. Search user name 2. Click the user’s username in the user list 3. Done | |

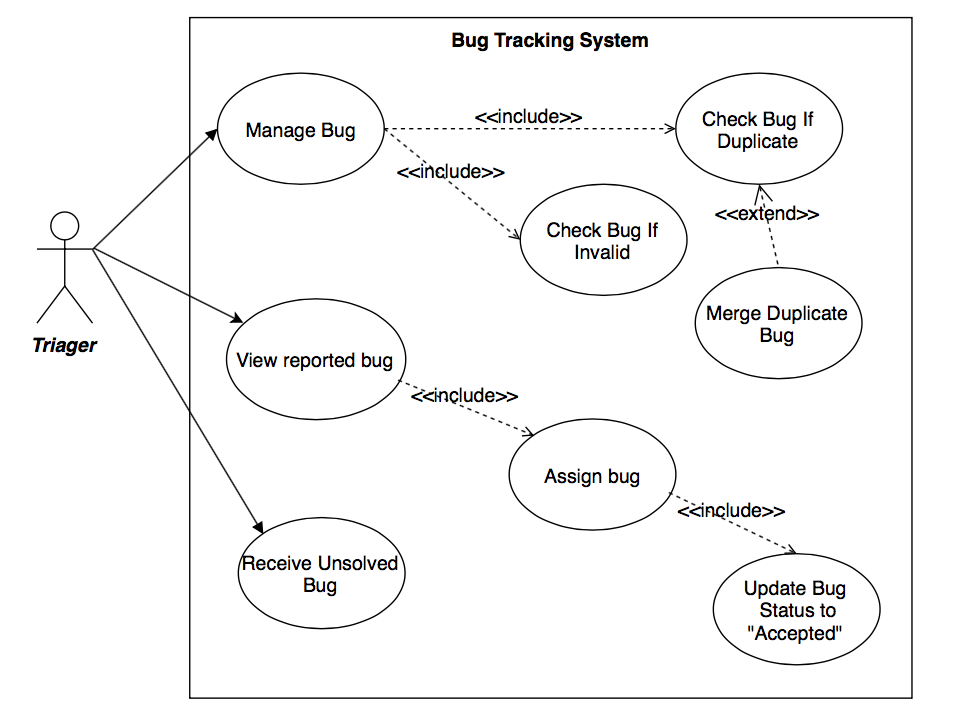
|  |  |
| --- | --- |
| ***Name:*** Edit profile | ***ID:*** #REPORT\_3 |
| ***Stakeholders and goals:*** Allow user to edit his own profile | |
| ***Description:*** User can edit his own profile | |
| ***Actors:*** Registered user | |
| ***Triggers:*** Registered user want to edit his own profile | |
| ***Normal flow:***   1. Click [username] on the top right corner 2. Edit profile 3. User confirm 4. System verifies data of the user’s input 5. Done | |
| ***Sub-flows:*** N/A | |
| ***Alternative/Exceptional flows:*** N/A | |

|  |  |
| --- | --- |
| ***Name:*** Comment on bug | ***ID:*** #REPORT\_4 |
| ***Stakeholders and goals:*** Allow registered user to comment on bug | |
| ***Description:*** Registered user can comment on a bug | |
| ***Actors:*** Registered user | |
| ***Triggers:*** Registered user want to comment on a bug | |
| ***Normal flow:***   1. Search for bug 2. Comment on bug | |
| ***Sub-flows:*** N/A | |
| ***Alternative/Exceptional flows:*** N/A | |

|  |  |
| --- | --- |
| ***Name:*** Report bug | ***ID:*** #REPORT\_5 |
| ***Stakeholders and goals:*** Allow registered user to report bug | |
| ***Description:*** Registered user can report bug | |
| ***Actors:*** Registered user | |
| ***Triggers:*** Registered user want to report a bug | |
| ***Normal flow:***   1. Click “New bug” 2. User fill in the form    1. Title    2. Code / file 3. Submit | |
| ***Sub-flows:*** N/A | |
| ***Alternative/Exceptional flows:*** N/A | |

|  |  |
| --- | --- |
| ***Name:*** Logout | ***ID:*** #REPORT\_6 |
| ***Stakeholders and goals:*** Allow registered user to logout from the system | |
| ***Description:*** Registered user can logout from the system | |
| ***Actors:*** Registered user | |
| ***Triggers:*** Registered user want to logout from the system | |
| ***Normal flow:***   1. Click “Logout” on the top right corner 2. Done | |
| ***Sub-flows:*** N/A | |
| ***Alternative/Exceptional flows:*** N/A | |

#### 2.3.3.2 Triager



Use case: #UC\_2

**Description**

The Triager is responsible for managing all quality aspects of a bug. His duty is to assign the bug to a suitable developer for solving it. Checking if the bug is duplicated or invalid.

|  |  |
| --- | --- |
| ***Name:*** Receive Unsolved bug | ***ID:*** #TRIAGER\_1 |
| ***Stakeholders and goals:*** Allow triager to receive the unsolved bug | |
| ***Description:*** Triager can receive the unsolved bug that returned by the reviewer | |
| ***Actors:*** Triager | |
| ***Triggers:*** Triager wants to receive the unsolved bug | |
| ***Normal flow:***   1. Check “Receive unsolved bug” | |
| ***Sub-flows:*** N/A | |
| ***Alternative/Exceptional flows:*** N/A | |

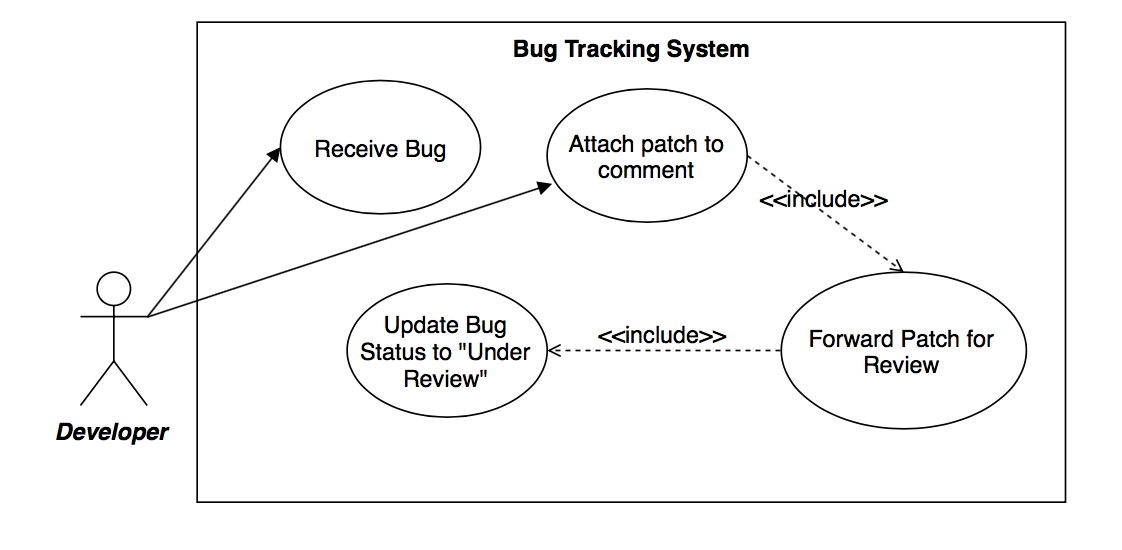
|  |  |
| --- | --- |
| ***Name:*** Manage bug | ***ID:*** #TRIAGER\_2 |
| ***Stakeholders and goals:*** Allow triager to manage the quality of the submitted bug | |
| ***Description:*** Triager can manage the quality aspects of the bug such checking if it is a duplicate or invalid bug | |
| ***Actors:*** Triager | |
| ***Triggers:*** When triager going to manage the quality aspect of a bug | |
| ***Normal flow:***   1. View bug information from received bug 2. Check the quality of the bug reported by the reporter (see sub-flows 1. 2.) 3. Check the quality aspect of the bug (see sub-flows 3. 4.) | |
| ***Sub-flows:***   1. Check whether the bug is invalid or not 2. Check whether the bug is duplicated or not | |
| ***Alternative/Exceptional flows:*** N/A | |

|  |  |
| --- | --- |
| ***Name:*** Merge duplicate bug | ***ID:*** #TRIAGER\_3 |
| ***Stakeholders and goals:*** Allow triager to merge the bug | |
| ***Description:*** Triager can merge the duplicated bug | |
| ***Actors:*** Triager | |
| ***Triggers:*** When triager going to merge the bug | |
| ***Normal flow:***   1. Check which bug are duplicated with 2. Merge the bug | |
| ***Sub-flows:*** | |
| ***Alternative/Exceptional flows:*** N/A | |

|  |  |
| --- | --- |
| ***Name:*** Review reported bug | ***ID:*** #TRIAGER\_4 |
| ***Stakeholders and goals:*** Allow triager to review the reported bug | |
| ***Description:*** Triager can review the reported bug and add technical information to it | |
| ***Actors:*** Triager | |
| ***Triggers:*** When triager going to review the reported bug | |
| ***Normal flow:***   1. Review the bug that are reported by the bug reporter 2. Add technical information according to the bug | |
| ***Sub-flows:*** N/A | |
| ***Alternative/Exceptional flows:*** N/A | |

|  |  |
| --- | --- |
| ***Name:*** Assigning bug | ***ID:*** #TRIAGER\_5 |
| ***Stakeholders and goals:*** Allow triager to assign the bug to a developer | |
| ***Description:*** Triager can assign the bug to a developer to fix it | |
| ***Actors:*** Triager | |
| ***Triggers:*** When triager going to assign the bug to a developer | |
| ***Normal flow:***   1. Choose developers to assign to 2. System will update the bug status to “Accepted” when the developer accepted to fixed the bug 3. Assign | |
| ***Sub-flows:*** N/A | |
| ***Alternative/Exceptional flows:*** N/A | |

#### 2.3.3.3 Developer



Use case: #UC\_3

##### Description

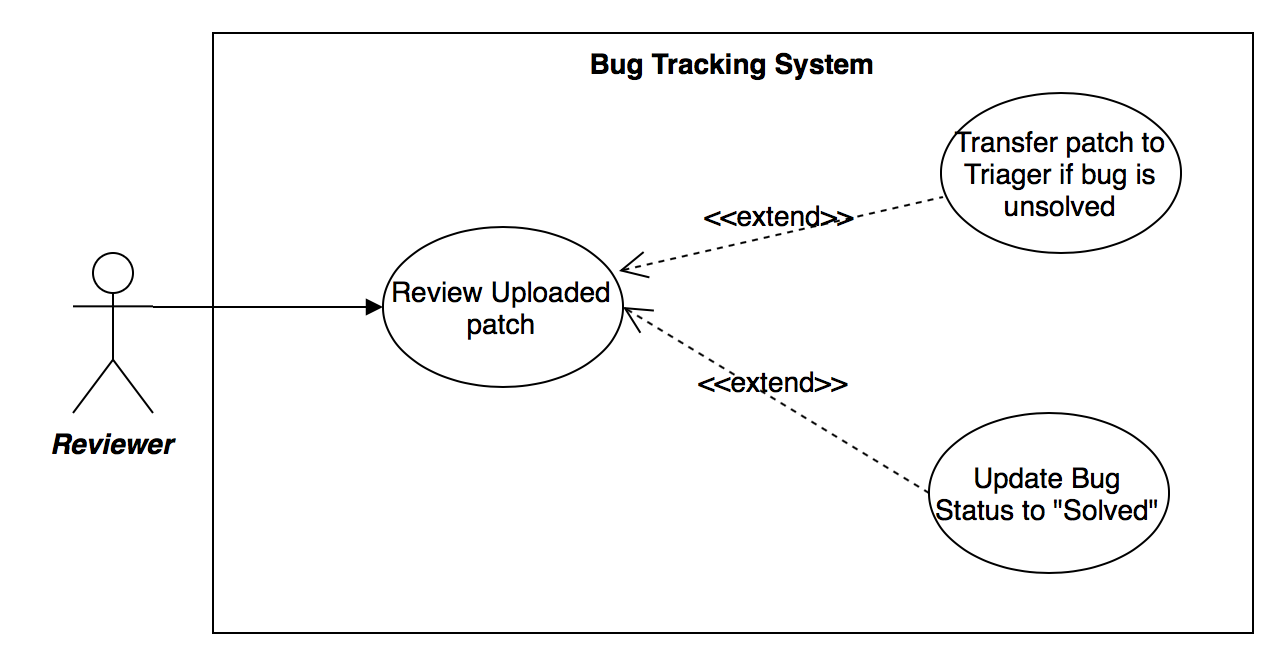
The developers are responsible for fixing bugs that are assigned to them by a triager.

|  |  |
| --- | --- |
| ***Name:*** Receive bug | ***ID:*** #DEVELOP\_1 |
| ***Stakeholders and goals:*** Allow the developer to receive the bug | |
| ***Description:*** The developer can upload the code to the system | |
| ***Actors:*** Developer | |
| ***Triggers:*** When the developer going to receive the bug | |
| ***Normal flow:***   1. Receive the bug that was forwarded by the triager | |
| ***Sub-flows:*** N/A | |
| ***Alternative/Exceptional flows:*** N/A | |

|  |  |
| --- | --- |
| ***Name:*** Attach patch on comment | ***ID:*** #DEVELOP\_2 |
| ***Stakeholders and goals:*** Allow the developer to attach the patch on the comment | |
| ***Description:*** The developer can attach the code on the comment | |
| ***Actors:*** Developer | |
| ***Triggers:*** When the developer wants to upload the code | |
| ***Normal flow:***   1. Upload the patch as an attachment on the comment | |
| ***Sub-flows:*** N/A | |
| ***Alternative/Exceptional flows:*** N/A | |

|  |  |
| --- | --- |
| ***Name:*** Forward patch for review | ***ID:*** #DEVELOP\_3 |
| ***Stakeholders and goals:*** Allow the developer to forward the patch | |
| ***Description:*** The developer can forward the patch to the reviewer | |
| ***Actors:*** Developer | |
| ***Triggers:*** When the developer wants to forward the patch | |
| ***Normal flow:***   1. Select the reviewer 2. Forward the patch 3. Done | |
| ***Sub-flows:***   1. The system will update the patch status to “Under review” | |
| ***Alternative/Exceptional flows:*** N/A | |

#### 2.3.3.4 Reviewers



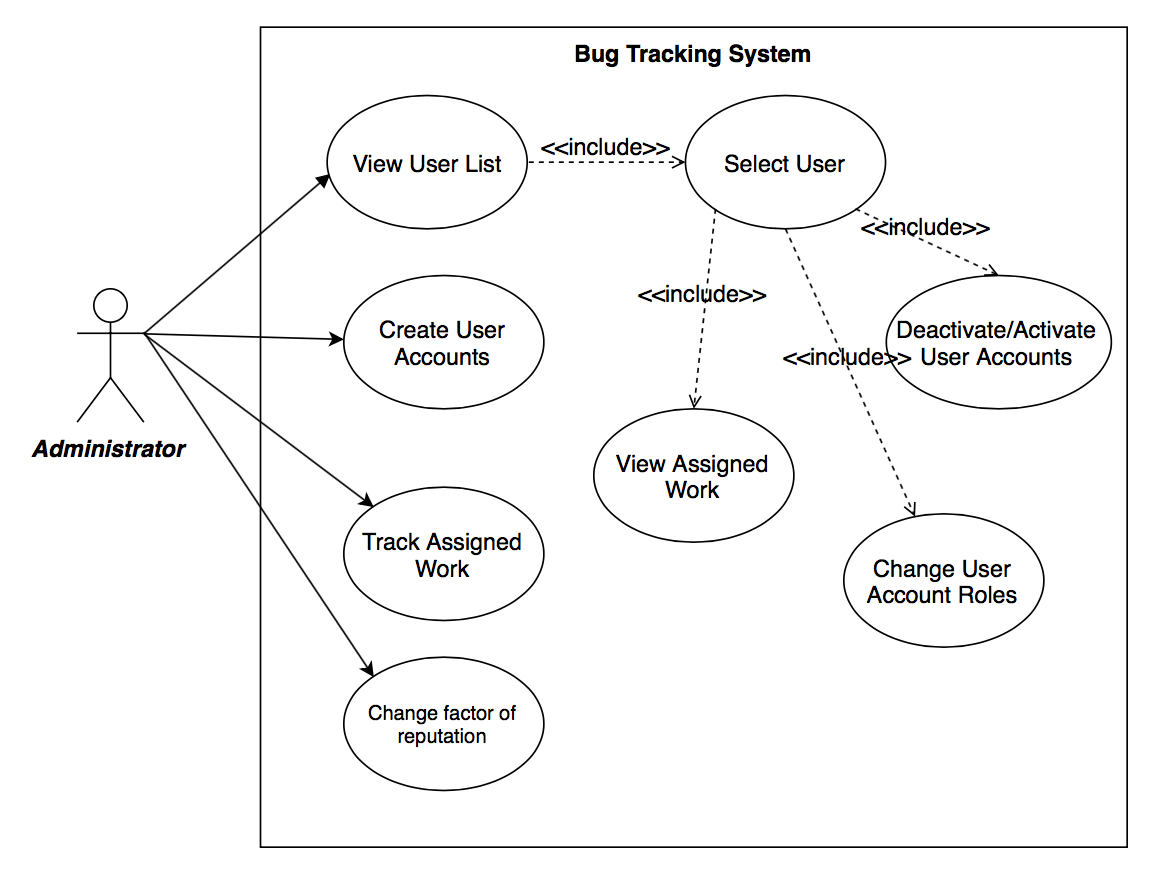
Use case: #UC\_4

##### Description

The Reviewers are responsible for checking to see if a bug has been fixed with the patch provided by the developer.

|  |  |
| --- | --- |
| ***Name:*** Review patch | ***ID:*** #REVIEW\_1 |
| ***Stakeholders and goals:*** Reviewer can check the bug status | |
| ***Description:*** Reviewer can check the bug to see whether it has been solved or not | |
| ***Actors:*** Reviewer | |
| ***Triggers:*** When the reviewer checks the bug status | |
| ***Normal flow:***   1. Check the patch 2. Check whether the bug have been fixed or not | |
| ***Sub-flows:***   1. If the bug has been fixed    1. Update the bug status to solved 2. If the bug has not been fixed    1. Decrease the reputation of the developer by the system    2. Transfer the patch to the triager for further action | |
| ***Alternative/Exceptional flows:*** ... | |

#### 2.3.3.5 Administrator



Use case: #UC\_5

**Description**

The administrator has the highest permissions in accessing the system. They are responsible for maintaining the system.

|  |  |
| --- | --- |
| ***Name:*** Deactivate/Activate User | ***ID:*** #ADMIN\_1 |
| ***Stakeholders and goals:*** Deactivate or activate user | |
| ***Description:*** Administrator can deactivate or activate user’s account so that the corresponding user will be no longer be able to login / will be able to login | |
| ***Actors:*** Administrator | |
| ***Triggers:*** If the admin wants to deactivate or activate a user’s account | |
| ***Normal flow:***   1. Log in 2. System verifies username and password. 3. Search for user (Directly type in user’s name / search from user list) 4. Go into user profile 5. Deactivate or activate user 6. Confirm 7. Done | |
| ***Sub-flows:*** N/A | |
| ***Alternative/Exceptional flows:*** N/A | |

|  |  |
| --- | --- |
| ***Name:*** Create user account | ***ID:*** #ADMIN\_2 |
| ***Stakeholders and goals:*** Create user | |
| ***Description:*** Administrator can create a user | |
| ***Actors:*** Administrator | |
| ***Triggers:*** If the admin wants to create a user | |
| ***Normal flow:***   1. Log in 2. System verifies username and password. 3. Click “Create user” 4. Type in account information 5. Assign role 6. Confirm 7. Done | |
| ***Sub-flows:*** N/A | |
| ***Alternative/Exceptional flows:*** N/A | |

|  |  |
| --- | --- |
| ***Name:*** Change user’s role | ***ID:*** #ADMIN\_3 |
| ***Stakeholders and goals:*** To change a user’s role | |
| ***Description:*** Administrator can change a user’s role | |
| ***Actors:*** Administrator | |
| ***Triggers:*** If the admin wants to change the user’s role | |
| ***Normal flow:***   1. Log in 2. System verifies username and password. 3. Search for user (Directly type in user’s name / search from user list) 4. Go into user’s profile 5. Click “Change role” 6. Choose role 7. Confirm 8. Done | |
| ***Sub-flows:*** N/A | |
| ***Alternative/Exceptional flows:*** N/A | |

|  |  |
| --- | --- |
| ***Name:*** View user list | ***ID:*** #ADMIN\_4 |
| ***Stakeholders and goals:*** To view the user list | |
| ***Description:*** Administrator can change user’s role | |
| ***Actors:*** Administrator | |
| ***Triggers:*** If the admin wants to view the user list | |
| ***Normal flow:***   1. Log in 2. System verifies username and password. 3. Click “View user” 4. Admin can select a user from the list 5. Done | |
| ***Sub-flows:*** N/A | |
| ***Alternative/Exceptional flows:*** N/A | |

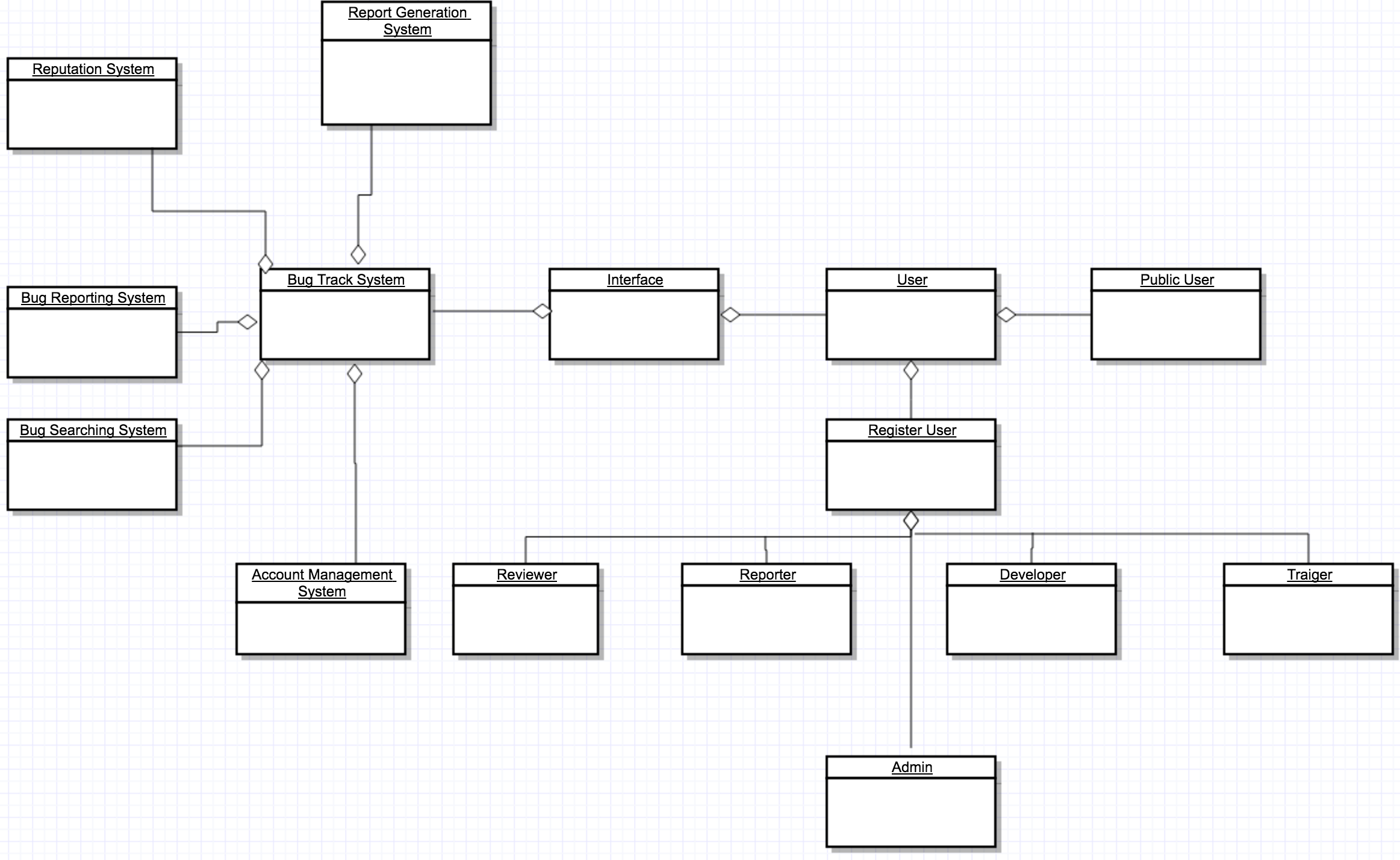
|  |  |
| --- | --- |
| ***Name:*** Track assigned work | ***ID:*** #ADMIN\_5 |
| ***Stakeholders and goals:*** To track the assigned work and status | |
| ***Description:*** Administrator can track the work that is assigned to each developer, and check the status of it. | |
| ***Actors:*** Administrator | |
| ***Triggers:*** If admin wants to track the work assigned | |
| ***Normal flow:***   1. Log in 2. System verifies username and password. 3. Click “Track work” 4. View the list 5. Done | |
| ***Sub-flows:*** N/A | |
| ***Alternative/Exceptional flows:*** N/A | |

|  |  |
| --- | --- |
| ***Name:*** Change factor of reputation | ***ID:*** #ADMIN\_6 |
| ***Stakeholders and goals:*** To change the factor of reputation | |
| ***Description:*** Administrator can change the factor of the reputation which are given by the system | |
| ***Actors:*** Administrator | |
| ***Triggers:*** If admin wants to change the factor | |
| ***Normal flow:***   1. Log in 2. System verifies username and password. 3. Click “Change factor” 4. Change the factor 5. Done | |
| ***Sub-flows:*** N/A | |
| ***Alternative/Exceptional flows:*** N/A | |

### 2.3.4 FUNCTIONAL REQUIREMENTS AND USE CASES MATRIX

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | #UC\_1 | #UC\_2 | #UC\_3 | #UC\_4 | #UC\_5 |
| #F\_2.3.1.1\_1 | X |  |  |  |  |
| #F\_2.3.1.1\_2 | X |  |  |  |  |
| #F\_2.3.1.1\_3 | X |  |  |  |  |
| #F\_2.3.1.1\_4 | X |  |  |  |  |
| #F\_2.3.1.2\_1 | X |  |  |  |  |
| #F\_2.3.1.2\_2 |  |  |  |  | X |
| #F\_2.3.1.2\_3 |  |  |  |  | X |
| #F\_2.3.1.2\_4 |  |  |  |  | X |
| #F\_2.3.1.2\_5 |  |  |  |  | X |
| #F\_2.3.1.2\_6 |  |  |  |  | X |
| #F\_2.3.1.2\_7 |  |  |  |  | X |
| #F\_2.3.1.3\_1 | X |  |  |  |  |
| #F\_2.3.1.3\_2 | X |  |  |  |  |
| #F\_2.3.1.3\_3 | X |  |  |  |  |
| #F\_2.3.1.3\_4 |  | X |  |  |  |
| #F\_2.3.1.3\_5 |  | X |  |  |  |
| #F\_2.3.1.3\_6 |  | X |  |  |  |
| #F\_2.3.1.3\_7 |  | X |  |  |  |
| #F\_2.3.1.3\_8 |  | X |  |  |  |
| #F\_2.3.1.3\_9 |  | X |  |  |  |
| #F\_2.3.1.4\_1 |  |  | X |  |  |
| #F\_2.3.1.4\_2 |  |  | X |  |  |
| #F\_2.3.1.4\_3 |  |  |  | X |  |
| #F\_2.3.1.4\_4 |  |  |  | X |  |
| #F\_2.3.1.5\_1 | X |  |  |  |  |
| #F\_2.3.1.5\_2 | X |  |  |  |  |
| #F\_2.3.1.6\_1 | X |  |  |  |  |
| #F\_2.3.1.7\_1 |  |  |  |  | X |

# 3.0 DOMAIN MODEL



# 4.0 PROJECT MANAGEMENT

## PROJECT COST

The estimation of the cost for the development of this software product will use the Constructive Cost Model (COCOMO) to calculate the amount of effort required for the project, then based on the development time:

**Effort Applied (E)** = a(KLOC)b

= 2.4 \* (3)1.05

= 7.60656462156

= 7.6 man-months

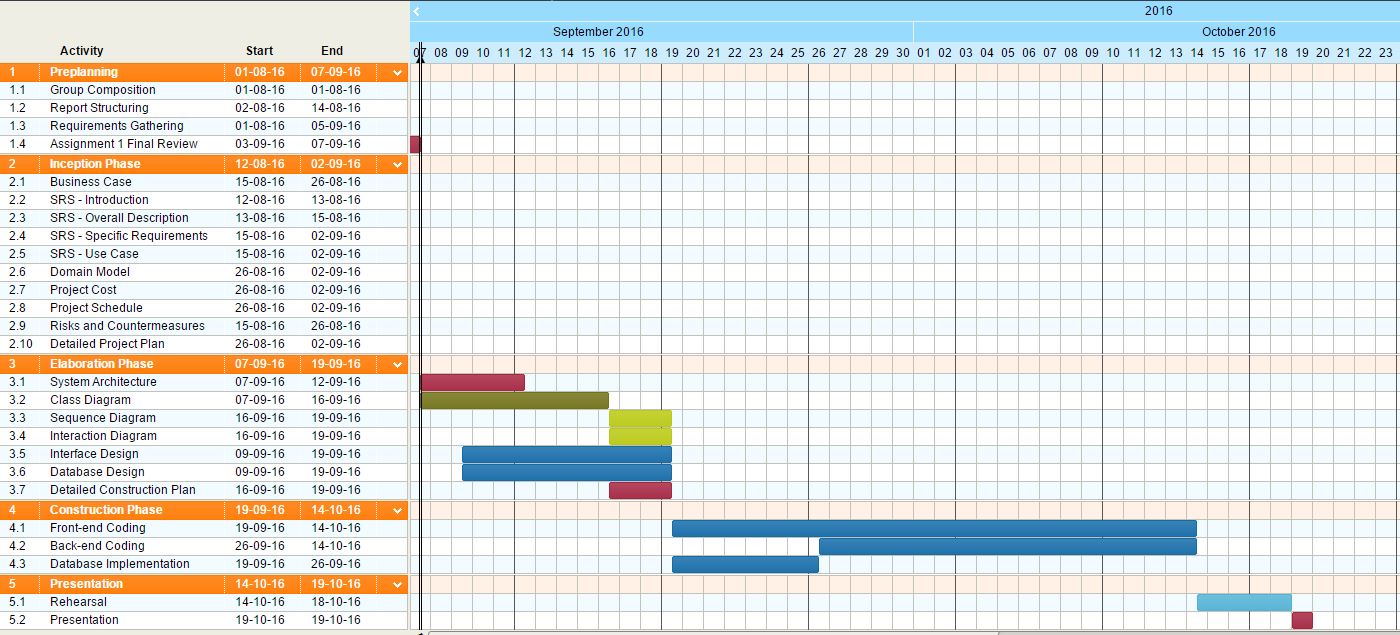
With this effort spread between 4 people, it should take approximately 1.9 months, or 8.25596 weeks. If we allow time for error or unexpected circumstances, 13 weeks should be ample time to complete the project. A total cost of $15,000 should ensure that we are paid the appropriate amount for both our analysis and development time. $15,000 / 4 = 3750; 3750 / 8 = 468.75. Hence, the project for the client will cost approximately $468.75 per week for our services of both analysis and development of the new system.

So therefore, by analysing costs of similar projects for a medium sized website, the cost is in fact quite reasonable because of the features and time that will be put into developing this website. Furthermore, the revenue that the client will generate by selling this website software will allow them to make a profit on the cost of hiring us for the development of the software. Finally, due to the nature of the software being developed, the client will have to expect a development cost of this proportion. This is because there must be a great deal of requirements analysis, which takes quite a large amount of time.

Since the Bugzilla software is free, the software being developed for the client will include more features and functions compared to that of Bugzilla. Otherwise, it would be a waste of time and money to develop a system that has the same features of that which is free of charge. In conclusion, the project cost is reasonable because of the features, time spent on the project, and expertise of the development team which are all factors when considering the final cost of a project. Of course this didn’t factor in the issues that may arise that will cost more money, but this is minimum amount required for the development of the system.

## PROJECT SCHEDULE





## PROJECT RISK ASSESSMENT

**Unrestricted Bug Reporting**

|  |
| --- |
| **Description:** Bug reports can be submitted by a registered user. However, the quality of the bug report is not guaranteed and the system can be subjected to bug report spamming, where bug reports are incorrect and are the results of mental frustration of the users from dealing with the bugs. |
| **Countermeasure:** Such reports can be handled by restricting the number of bug reports submitted based on the reputation that the user has, or based on a strict number set by the user administrator. |

**Excessive Accumulation of Bug Reports Over Time**

|  |
| --- |
| **Description:** As the system is utilized over time, the number of bug reports will accumulate, and the system may be slowed down due to a large number of entries in the database. |
| **Countermeasure:** This can be dealt with by creating a scalable database model, categorizing and indexing bug reports in the database, and the triager can manage and merge bug reports into the correct categories. |

**Complex Bug Reports that Describes Multiple Bugs**

|  |
| --- |
| **Description:** Bug reports can potentially be associated with multiple types of bugs, whether it be related or non-related, and a complex one can make it difficult to be managed and solved. |
| **Countermeasure:** It is up to the triager and his technical expertise to break down the bug report into multiple smaller reports so it can be easily assigned to the respective developer to be solved. It is also possible to implement a tagging system that organizes bug reports into multiple distinct sections. |

**Requirement Changes**

|  |
| --- |
| **Description:** As the system gets developed, additional features are suggested based on requirements gathered after the inception phase, due to the requirement changes or some features that was not identified clearly at the early phases. |
| **Countermeasure:** This can be solved by gathering requirements in short intervals while constantly reassessing the current requirement to ensure the requirements are implemented into the system functionalities. |

**Project Creep**

|  |
| --- |
| **Description:** Aside from requirement changes, project creep can also happen where multiple requirements and features are suggested and added uncontrollably to the extent where time or budget runs out. |
| **Countermeasure:** This can be avoided by prioritizing the core functionalities of the system to be implemented before assessing the needs of additional features, and by negotiating with the client on the prioritization of such features. |

**Library Support for Implementation**

|  |
| --- |
| **Description:** The lack of proper libraries to support the implementation of system functionalities can be critical when developing the system. |
| **Countermeasure:** A detailed understanding of the system architecture needs to be established to highlight existing tools and libraries, as well as the impact of using said libraries. |

**Schedule Inconsistency and Productivity Issues**

|  |
| --- |
| **Description:** Due to long project timelines, project schedule may not be consistently followed and the urgency for completing the project can be dulled, and may result in time being wasted overall. |
| **Countermeasure:** This can be avoided by dividing the long schedule into multiple milestones and smaller schedules, which lets us focus on the short term plan, while keeping an eye on the less detailed long term plan. |

**Data Loss**

|  |
| --- |
| **Description:** Database failure is a potential risk to the system’s functionalities. If the host computer storing the database encounter problems like hard drive failure or data corruption, the system will not be able to function normally. It is also a problem if the cloud backup service or version control software storing our documents or server data becomes unavailable. |
| **Countermeasure:** To solve this, we will back up our database frequently and store copies of it in multiple computers as well as in external hard drives separated from the main computer while using version control to synchronize the various data in the multiple computers. |

**Miscommunication and Language Barrier**

|  |
| --- |
| **Description:** Being in a group of various cultural background can make communication difficult and may result in miscommunication among team members. |
| **Countermeasure:** We can overcome this by communicating in a more concise and simple manner, while making sure everyone understands the topic before proceeding to the next. |

# 5.0 PROJECT PLAN FOR INCEPTION, ELABORATION, AND CONSTRUCTION PHASE

## INCEPTION PHASE

This phase is the planning phase, where the business case is established, scope of the project is determined through the Software Requirement Specification, and the initial project cost and schedule is estimated. A detailed plan for the whole project is also produced in order to determine the tasks required to achieve the desired goal of the project.

The outcome of the inception phase includes:

* A Complete and Detailed Business Case
* A Complete and Detailed Software Requirement Specification consisting of Functional Requirements, Non-functional Requirements, and Use Case.
* A Complete and Detailed Domain Model
* A Complete and Detailed Project Cost and Schedule
* A Complete and Detailed Risks and Countermeasures
* A Complete and Detailed Project Plan of Inception, Elaboration, and Construction Phase

## ELABORATION PHASE

In this phase, the system architecture of the project is revised, validated and established. The existing use case is refined and class diagrams, sequence diagrams, and interaction diagrams are created to support the use case. Interface and Database designs are also created. Finally, a more detailed construction plan is produced.

The outcome of the elaboration phase should include:

* System Architecture
* Class Diagram
* Sequence Diagram
* Interaction Diagram
* Interface Design
* Database Design
* Detailed Construction Plan

## CONSTRUCTION PHASE

This is the implementation phase where majority of the coding is done to implement both front-end and back-end of the system according to requirements from inception phase and designs from the elaboration phase. Requirements will be constantly referred so as to ensure the features being implemented into the system meets the client’s needs.

The outcome of the construction phase will include:

* A complete system with both front-end, back-end, and database implemented to support requirements specified in the Software Requirement Specification.

META REPORT

# GROUP STRUCTURE, ROLE, AND ARTEFACTS

## GROUP STRUCTURE AND ROLE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Roles** | **Jingwang Teh** | **Weicheng Yin** | **Luke O’Callaghan** | **John Chun-Yu Chan** |
| **Project Manager** | **X** |  |  |  |
| **System Analyst** | **X** |  |  | **X** |
| **User Interface Designer** |  | **X** |  | **X** |
| **Software Architect** |  | **X** |  |  |
| **Data Designer** | **X** |  | **X** |  |
| **Implementer** |  | **X** |  | **X** |
| **Integrator** |  |  | **X** |  |
| **Test Designer** | **X** |  | **X** |  |
| **Tester** | **X** | **X** | **X** | **X** |
| **Configuration Manager** | **X** |  |  |  |

## INITIAL PROJECT PLAN AND ARTEFACTS DELIVERED

|  |  |  |  |
| --- | --- | --- | --- |
| **Artefact** | **Due** | **Responsible** | **Status & Comment** |
| **Business Case** | **19 August 2016** | Jingwang Teh & Luke O’Callaghan | **Completed.** |
| **SRS - Introduction** | **13 August 2016** | Weicheng Yin & Jingwang Teh | **Completed.** |
| **SRS - Overall Description** | **15 August 2016** | Weicheng Yin & Jingwang Teh | **Completed.** |
| **SRS - Specific Requirements (Functional Requirement)** | **29 August 2016** | Chun Yu Chan & Luke O’Callaghan | **Completed.** |
| **SRS - Specific Requirements (Non-Functional Requirement)** | **29 August 2016** | Jingwang Teh & Weicheng Yin | **Completed.** |
| **SRS - Use Case** | **26 August 2016** | Jingwang Teh & Chun Yu Chan | **Completed.** |
| **Domain Model** | **2 September 2016** | Weicheng Yin | **Completed.** |
| **Project Cost** | **2 September 2016** | Luke O’Callaghan | **Completed.** |
| **Project Schedule** | **2 September 2016** | Jingwang Teh | **Completed.** |
| **Risks and Countermeasure** | **19 August 2016** | Jingwang Teh & Chun Yu Chan | **Completed.** |
| **Candidate Architecture** | **2 September 2016** | Weicheng Yin | **Cancelled.**  **It was not required in inception phase.** |
| **Detailed Plan for Elaboration and Construction phase** | **2 September 2016** | Jingwang Teh | **Completed.** |

## ITERATION PLAN

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Milestone** | **Artefact** | **Completion Dates** | | |
| **Planned** | **Revised** | **Actual** |
| **Iteration 1 (Inception)** |  | 7 September 2016 |  | 7 September 2016 |
|  | **Initial Project Plan** | **24 August 2016** |  | **24 August 2016** |
|  | **Inception Iteration Plan** | **24 August 2016** |  | **24 August 2016** |
|  | Business Case | 19 August 2016 | 26 August 2016 | 26 August 2016 |
|  | SRS - Introduction | 13 August 2016 |  | 13 August 2016 |
|  | SRS - Overall Description | 15 August 2016 |  | 15 August 2016 |
|  | SRS - Specific Requirements (Functional Requirement) | 29 August 2016 | 2 September 2016 | 2 September 2016 |
|  | SRS - Specific Requirements (Non-Functional Requirement) | 29 August 2016 | 2 September 2016 | 2 September 2016 |
|  | SRS - Use Case | 26 August 2016 | 29 August 2016,  2 September 2016 | 2 September 2016 |
|  | Domain Model | 2 September 2016 |  | 2 September 2016 |
|  | Project Cost | 2 September 2016 |  | 2 September 2016 |
|  | Project Schedule | 2 September 2016 |  | 2 September 2016 |
|  | Risks and Countermeasure | 19 August 2016 | 26 August 2016 | 26 August 2016 |
|  | Candidate Architecture | 2 September 2016 | Taken out of Inception, as it was not needed in this phase. |  |
|  | Detailed Plan for Elaboration and Construction phase | 2 September 2016 |  | 2 September 2016 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Iteration 2**  **(Elaboration)** |  | 19 September 2016 |  |  |
|  | **Updated Project Plan** | **7 September 2016** |  |  |

# PROJECT MEETING

## Week 3 Meeting #1 - 12 August 2016

### Summary

This is our group’s first meeting. The goal for this first meeting is to ensure group and project structure integrity by assigning roles and identifying our project priorities.

Aside from that, we discuss about information gathered during the lab session in week 2 & 3, and also the viability and necessity of the options available for architecture, software, and other tools.

* Through the information we gathered from the lab session, we know that the client will sell the software system that we build to their clients. It has been highlighted by the client that our system is intended to be bought at a lower cost in order to be sold to their client at a higher price to generate revenue from it.
* We have discussed that we should implement the system in a web-based GUI format.
* However, we consider that C++ might not be the programming language that we want to use, as Java has better library support for implementing a web-based system. However, most of us have better understanding on C++ so we have yet to determine the programming language that we want to use.

We also give our ideas on several points of the project through a brainstorming session, involving the given examples and existing systems like Bugzilla, and come up with ways to approach and implement them for our system accordingly.

* We have discussed that the User Interface(UI) of the system and its layout should have a minimalistic design.
* The system’s homepage should contain functionality like search bug catalogue, and submit bug report. The header bar on the homepage should include shortcuts to register a new account, login, and password retrieval, as well as shortcuts to other pages within the system.
* The position of the header bar should be fixed so that users can easily navigate around the system without having to scroll all the way up.
* We have also discussed the viability of having our system support multiple products, each with its own category within the bug tracking system.

We were all notified of the current project documentation structure and templates as well.

Lastly, project priorities until the next meeting was discussed and then we distribute the required tasks to each group member and focused on ensuring each other’s understanding on the current project goal until the next meeting.

* Our project priority is set on gathering further requirements from the client in week 4 lab session in order to expand on our Software Requirement Specification (SRS), provide use case according to requirements gathered, analysing risks and countermeasures, and to find out the details required for the topic on detailed plans for whole project.

### Agenda

* Group-related
  + Assign Roles to Group Members.
* Project Content
  + Discuss about Information provided by Client during the Lab Session in Week 2 and Week 3.
  + Discuss on Existing Architecture, Environment, Software and Tools that are necessary.
  + Brainstorm Ideas on Ways to Approach and Implement the System according Project Outline.
  + Ensure Templates for Documentation are Known and Used by All Members.
* Project Direction
  + Discuss Project Priorities between Current Meeting and the Next Meeting and Assign Tasks to Each Group Member.

### Report

**Urgent Items**

There were no urgent items to report in this meeting as this is the first meeting we held.

**Non-urgent Items and Other Reports**

There were no non-urgent items or other reports to report in this meeting as this is the first meeting we held.

## Week 4 Meeting #2 - 19 August 2016

### Summary

In this second meeting, we have spent most of the time migrating our documents to GitHub for version control. We have also reviewed the Use Case, and improve upon it for review during lab with client. Business Case was also established.

Then, we distribute the tasks to each member to be handled with until the next meeting, including interface for Software Requirement Specifications, and further Use Case and Business Case improvements.

### Agenda

* Project Content
  + Establish Business Case.
  + Discuss and Review Use Case based on Functional Requirements from Q&A.
* Project Direction
  + Inform Group Members to Migrate Documents from Google Docs to GitHub for Version Control.
  + Assign Tasks to Each Group Member.

### Report

**Urgent Items**

Use Case was incomplete as it requires description to be defined for different scenarios

**Non-urgent Items and Other Reports**

Software Requirement Specification - Introduction marked as complete.

Software Requirement Specification - Overall Description marked as complete.

## Week 5 Meeting #3 - 26 August 2016

### Summary

In this meeting, we review our use case according to requirements gathered from previous Q&A and identified changes that need to be made, while associating functionalities with the existing draft of our user interface.

Also, we ensure all group members are aware of the current progress of the project, and distributes the remaining tasks to each group member. We also discuss the tasks that needs to be done in between the first assignment and the second assignment, as well as explaining the overview of the second assignment, such as the need for class diagrams and sequence diagrams.

### Agenda

* Project Content
  + Review Use Case description
  + Discuss about Functional Requirement, and associating it with Use Case
  + Discuss about candidate architecture
* Project Direction
  + Ensure each group member is aware of overall progress
  + Assign task to each group member

### Report

**Urgent Items**

Use Case revised and improved.

Functional Requirements Template explained and requires more input.

**Non-urgent Items and Other Reports**

Business Case reviewed and marked as complete.

Risk analysis reviewed and marked as complete.

Initial Project Plan and Inception Iteration Plan was established.

## Week 6 Meeting #4 - 2 September 2016

### Summary

In this meeting, we reviewed and polished our use case and functional requirement. We have also gone through the complete documentation with each group member reviewing the artefacts that they have delivered, and reviewing each member’s work diary.

### Agenda

* Project Content
  + Review Use Case and Functional Requirement
  + Review Non-functional Requirement
  + Review Domain model
* Project Direction
  + Go through with the group about all that was accomplished
  + Ensure each member knows their contribution to the first assignment and their responsibilities moving forward

### Report

**Urgent Items**

Use Case and Functional Requirement reviewed and revised, and polished up.

Non-functional requirement reviewed and completed.

Domain model reviewed and completed.

**Non-urgent Items and Other Reports**

Project Cost and Schedule reviewed and completed.

Candidate architecture removed as it was not necessary.

Detailed plan for elaboration and construction phase reviewed and completed

# WORK DIARIES

## Work Diary by Jingwang Teh

**Planned Work Schedule vs Actual Work Time**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week 2 (1 August 2016) to Week 3 (12 August 2016)** | | | |
| **Task to Complete** | **Planned Schedule** | **Actual Work Time** | **Completion** |
| Create initial structure for project documentation | 2 days | 2 days | Complete |
| Revise structure for project documentation | 1 day | 1 day | Complete |
| Create and revise template for meta-report (including group structure and role, artefacts delivered, and work diary) | 1 day | 1 day | Complete |
| Create template for risk and countermeasures | < 1 day | < 1 day | Complete |
| Prepare Q&A for week 2 lab session and record answers from Q&A | 1 day | 1 day | Complete |
| Prepare Q&A for week 3 lab session and record answers from Q&A | 1 day | 1 day | Complete |
| Prepare agenda for first meeting on week 3 (12 August 2016) | 1 day | 1 day | Complete |

**Report on Defects and Countermeasure**

|  |  |
| --- | --- |
| **Week 2 (1 August 2016) to Week 3 (12 August 2016)** | |
| **Defects** | **Countermeasure** |
| Documentation structure was not well-defined | Several revisions to improve documentation structures was done based on a clearer understanding on project requirements |

**Planned Work Schedule vs Actual Work Time**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week 3 (12 August 2016) to Week 4 (19 August 2016)** | | | |
| **Task to Complete** | **Planned Schedule** | **Actual Work Time** | **Completion** |
| Write report on first meeting | < 1 day | < 1 day | Complete |
| Restructure Q&A format | 1 day | 1 day | Complete |
| Write content for SRS - Introduction (Purpose, Scope, and Overview), and Identify subsystems associated with use case | 2 days | 2 days | Complete |
| Write content for SRS - Overall Description | 1 day | 1 day | Complete |
| Contribute and write for topic on risks and countermeasures | 1 day | 1 day | Complete |
| Revise structure for project documentation | 1 day | 1 day | Complete |
| Create template for requirements based on lecture | < 1 day | < 1 day | Complete |
| Prepare Q&A for week 4 lab session and record answers from Q&A | 1 day | 1 day | Complete |
| Create Repository on GitHub | 1 day | 1 day | Complete |
| Prepare agenda for meeting on week 4 (19 August 2016) | 1 day | 1 day | Complete |
| Review Use Case and Format used | 1 day | 1 day | Complete |

**Report on Defects and Countermeasure**

|  |  |
| --- | --- |
| **Week 3 (12 August 2016) to Week 4 (19 August 2016)** | |
| **Defects** | **Countermeasure** |
| None | None |

**Planned Work Schedule vs Actual Work Time**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week 4 (19 August 2016) to Week 5 (26 August 2016)** | | | |
| **Task to Complete** | **Planned Schedule** | **Actual Work Time** | **Completion** |
| Write report on week 4 meeting on 19Aug2016 | < 1 day | < 1 day | Complete |
| Move Documents to GitHub | 1 day | 1 day | Complete |
| Revise Artefacts to be Delivered | < 1 day | < 1 day | Complete |
| Create Iteration Plan | 1 day | 1 day | Complete |
| Review and Complete Business Case | < 1 day | < 1 day | Complete |
| Review and Complete Risks Assessment | 1 day | 1 day | Complete |
| Review Use Case | < 1 day | < 1 day | Complete |
| Prepare agenda for meeting on week 5 (26 August 2016) | 1 day | 1 day | Complete |

**Report on Defects and Countermeasure**

|  |  |
| --- | --- |
| **Week 4 (19 August 2016) to Week 5 (26 August 2016)** | |
| **Defects** | **Countermeasure** |
| None | None |

**Planned Work Schedule vs Actual Work Time**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week 5 (26 August 2016) to Week 6 (2 September 2016)** | | | |
| **Task to Complete** | **Planned Schedule** | **Actual Work Time** | **Completion** |
| Write report on week 5 meeting on 26Aug2016 | < 1 day | < 1 day | Complete |
| Create and Completed Project Schedule | 1 day | 1 day | Complete |
| Create and Completed Detailed Project Plans | 1 day | 1 day | Complete |
| Prepare Q&A for week 6 lab session and record answers from Q&A | 1 day | 1 day | Complete |
| Write Non-functional Requirements | 1 day | 1 day | Complete |
| Prepare agenda for meeting on week 6 (2 September 2016) | 1 day | 1 day | Complete |

**Report on Defects and Countermeasure**

|  |  |
| --- | --- |
| **Week 5 (26 August 2016) to Week 6 (2 September 2016)** | |
| **Defects** | **Countermeasure** |
| None | None |

**Planned Work Schedule vs Actual Work Time**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week 6 (2 September 2016) to Week 7 (7 September 2016)** | | | |
| **Task to Complete** | **Planned Schedule** | **Actual Work Time** | **Completion** |
| Write report on week 6 meeting on 2Sept2016 | < 1 day | < 1 day | Complete |
| Added Evidence of Version Control into Meta-report | 1 day | 1 day | Complete |
| Review through the complete documentation again for final polish | 2 days | 2 days | Complete |
| Prepare final Q&A for week 7 lab session and record answers from Q&A | 1 day | 1 day | Complete |
| Compile all documents and print out hard copy | < 1 day | < 1 day | Complete |

**Report on Defects and Countermeasure**

|  |  |
| --- | --- |
| **Week 6 (2 September 2016) to Week 7 (7 September 2016)** | |
| **Defects** | **Countermeasure** |
| None | None |

## Work Diary by Weicheng Yin

**Planned Work Schedule vs Actual Work Time**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week 2 (1 August 2016) to Week 3 (12 August 2016)** | | | |
| **Task to Complete** | **Planned Schedule** | **Actual Work Time** | **Completion** |
| Draft requirement | 1 week | 1 week | Complete |

**Report on Defects and Countermeasure**

|  |  |
| --- | --- |
| **Week 2 (1 August 2016) to Week 3 (12 August 2016)** | |
| **Defects** | **Countermeasure** |
| None | None |

**Planned Work Schedule vs Actual Work Time**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week 3 (12 August 2016) to Week 4 (19 August 2016)** | | | |
| **Task to Complete** | **Planned Schedule** | **Actual Work Time** | **Completion** |
| Draft requirement | 1 week | 1 week | Complete |
| Contribute to Q&A | 1 day | <1 day | Complete |
| UI design | 1 day | 1 day | Complete |

**Report on Defects and Countermeasure**

|  |  |
| --- | --- |
| **Week 3 (12 August 2016) to Week 4 (19 August 2016)** | |
| **Defects** | **Countermeasure** |
| None | None |

**Planned Work Schedule vs Actual Work Time**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week 4 (19 August 2016) to Week 5 (26 August 2016)** | | | |
| **Task to Complete** | **Planned Schedule** | **Actual Work Time** | **Completion** |
| Contribute to Q & A | 1 day | < 1 day | Complete |
| Code implementation design | 1 day | 1 week | Complete |
| Deep system analysis | 1 day | 1 week | Complete |

**Report on Defects and Countermeasure**

|  |  |
| --- | --- |
| **Week 4 (19 August 2016) to Week 5 (26 August 2016)** | |
| **Defects** | **Countermeasure** |
| None | None |

**Planned Work Schedule vs Actual Work Time**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week 5 (26 August 2016) to Week 6 (2 September 2016)** | | | |
| **Task to Complete** | **Planned Schedule** | **Actual Work Time** | **Completion** |
| Contribute to Q & A | < 1 day | < 1 day | Complete |
| SRS non-functional requirement | 1 week | 1 week | Complete |
| Code environment set up | 3 days | 1 week | Complete |

**Report on Defects and Countermeasure**

|  |  |
| --- | --- |
| **Week 5 (26 August 2016) to Week 6 (2 September 2016)** | |
| **Defects** | **Countermeasure** |
| None | None |

**Planned Work Schedule vs Actual Work Time**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week 6 (3 September 2016) to Week 7 7 September 2016)** | | | |
| **Task to Complete** | **Planned Schedule** | **Actual Work Time** | **Completion** |
| SRS non-functional requirement | 2 days | 1 day | Complete |
| Code implementation design | 1 day | 1 day | Complete |
| Deep system analysis | 2 days | 2 days | Complete |

**Report on Defects and Countermeasure**

|  |  |
| --- | --- |
| **Week 5 (26 August 2016) to Week 6 (2 September 2016)** | |
| **Defects** | **Countermeasure** |
| None | None |

## Work Diary by Luke O’Callaghan

**Planned Work Schedule vs Actual Work Time**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week 2 (5 August 2016) to Week 3 (12 August 2016)** | | | |
| **Task to Complete** | **Planned Schedule** | **Actual Work Time** | **Completion** |
| Draft requirements | 1 week | 1 week | Complete |

**Report on Defects and Countermeasure**

|  |  |
| --- | --- |
| **Week 2 (5 August 2016) to Week 3 (12 August 2016)** | |
| **Defects** | **Countermeasure** |
| None | None |

**Planned Work Schedule vs Actual Work Time**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week 3 (12 August 2016) to Week 4 (19 August 2016)** | | | |
| **Task to Complete** | **Planned Schedule** | **Actual Work Time** | **Completion** |
| Draft requirements | 1 week | 1 week | Complete |
| Contribute to the Q&A | During class time | During class time | Complete |

**Report on Defects and Countermeasure**

|  |  |
| --- | --- |
| **Week 3 (12 August 2016) to Week 4 (19 August 2016)** | |
| **Defects** | **Countermeasure** |
| None | None |

**Planned Work Schedule vs Actual Work Time**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week 4 (19 August 2016) to Week 5 (26 August 2016)** | | | |
| **Task to Complete** | **Planned Schedule** | **Actual Work Time** | **Completion** |
| Draft requirements | 1 week | 1 week | Complete |
| SRS functional requirement review | 1.5 weeks | 2 weeks | Complete |

**Report on Defects and Countermeasure**

|  |  |
| --- | --- |
| **Week 4 (19 August 2016) to Week 5 (26 August 2016)** | |
| **Defects** | **Countermeasure** |
| None. | None. |

**Planned Work Schedule vs Actual Work Time**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week 5 (26 August 2016) to Week 6 (1 September 2016)** | | | |
| **Task to Complete** | **Planned Schedule** | **Actual Work Time** | **Completion** |
| Draft requirements | 1 week | 1 week | Complete |
| Write cost analysis | 1 day | 1 day | Complete |

**Report on Defects and Countermeasure**

|  |  |
| --- | --- |
| **Week 5 (26 August 2016) to Week 6 (1 September 2016)** | |
| **Defects** | **Countermeasure** |
| None | None |

**Planned Work Schedule vs Actual Work Time**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week 6 (1 September 2016) to Week 7 (5 September 2016)** | | | |
| **Task to Complete** | **Planned Schedule** | **Actual Work Time** | **Completion** |
| Finalize requirements | 1 week | 1 week | Complete |
| Review cost analysis | 1 day | 1 day | Complete |

**Report on Defects and Countermeasure**

|  |  |
| --- | --- |
| **Week 6 (1 September 2016) to Week 7 (5 September 2016)** | |
| **Defects** | **Countermeasure** |
| None | None |

## Work Diary by John Chun-Yu Chan

**Planned Work Schedule vs Actual Work Time**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week 2 (5 August 2016) to Week 3 (12 August 2016)** | | | |
| **Task to Complete** | **Planned Schedule** | **Actual Work Time** | **Completion** |
| Draft requirements | 4 days | 4 days | Complete |

**Report on Defects and Countermeasure**

|  |  |
| --- | --- |
| **Week 2 (5 August 2016) to Week 3 (12 August 2016)** | |
| **Defects** | **Countermeasure** |
| None | None |

**Planned Work Schedule vs Actual Work Time**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week 3 (12 August 2016) to Week 4 (19 August 2016)** | | | |
| **Task to Complete** | **Planned Schedule** | **Actual Work Time** | **Completion** |
| Risks and countermeasures | 1 week | 1 week | Complete |

**Report on Defects and Countermeasure**

|  |  |
| --- | --- |
| **Week 3 (12 August 2016) to Week 4 (19 August 2016)** | |
| **Defects** | **Countermeasure** |
| None | None |

**Planned Work Schedule vs Actual Work Time**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week 4 (19 August 2016) to Week 5 (26 August 2016)** | | | |
| **Task to Complete** | **Planned Schedule** | **Actual Work Time** | **Completion** |
| Use Case Diagram | 2 weeks | 2 weeks | Complete |
| SRS functional requirement | 2 weeks | 2 weeks | Complete |
| UI draft | Until design phase | - | Until design phase |
| Risks and countermeasures | 1 week | 2 weeks | Complete |

**Report on Defects and Countermeasure**

|  |  |
| --- | --- |
| **Week 4 (19 August 2016) to Week 5 (26 August 2016)** | |
| **Defects** | **Countermeasure** |
| None | None |

**Planned Work Schedule vs Actual Work Time**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week 5 (26 August 2016) to Week 6 (1 September 2016)** | | | |
| **Task to Complete** | **Planned Schedule** | **Actual Work Time** | **Completion** |
| Use Case Diagram | 2 weeks | 2 weeks | Complete |
| SRS functional requirement | 2 weeks | 2 weeks | Complete |

**Report on Defects and Countermeasure**

|  |  |
| --- | --- |
| **Week 5 (26 August 2016) to Week 6 (1 September 2016)** | |
| **Defects** | **Countermeasure** |
| None | None |

**Planned Work Schedule vs Actual Work Time**

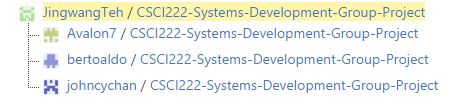
|  |  |  |  |
| --- | --- | --- | --- |
| **Week 6 (1 September 2016) to Week 7 (5 September 2016)** | | | |
| **Task to Complete** | **Planned Schedule** | **Actual Work Time** | **Completion** |
| Review Use Case Diagram | 1 day | 3 days | Complete |
| Review SRS functional requirement | 1 day | 3 days | Complete |
| Edit report | 1 week | 1 week | Complete |

**Report on Defects and Countermeasure**

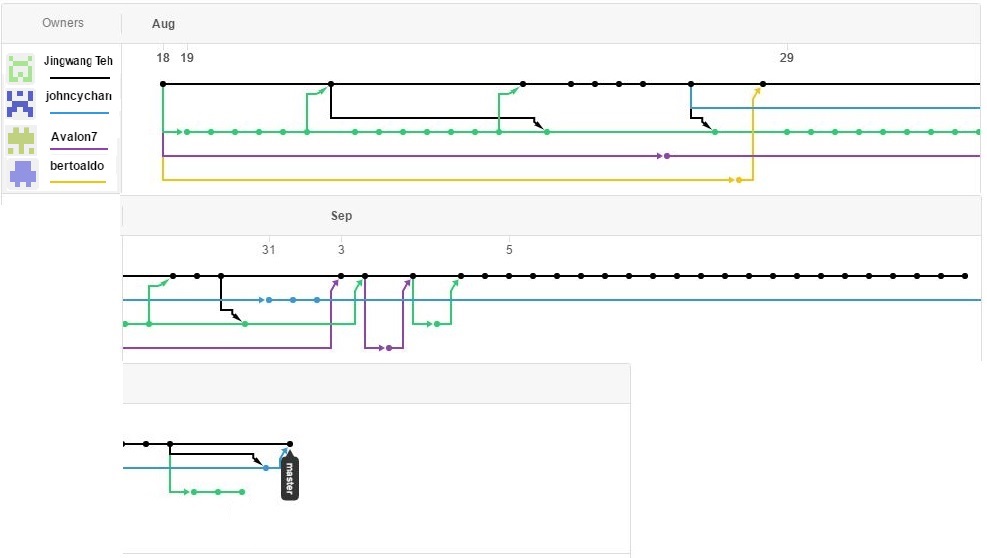
|  |  |
| --- | --- |
| **Week 6 (1 September 2016) to Week 7 (5 September 2016)** | |
| **Defects** | **Countermeasure** |
| None | None |

# EVIDENCE OF VERSION CONTROL

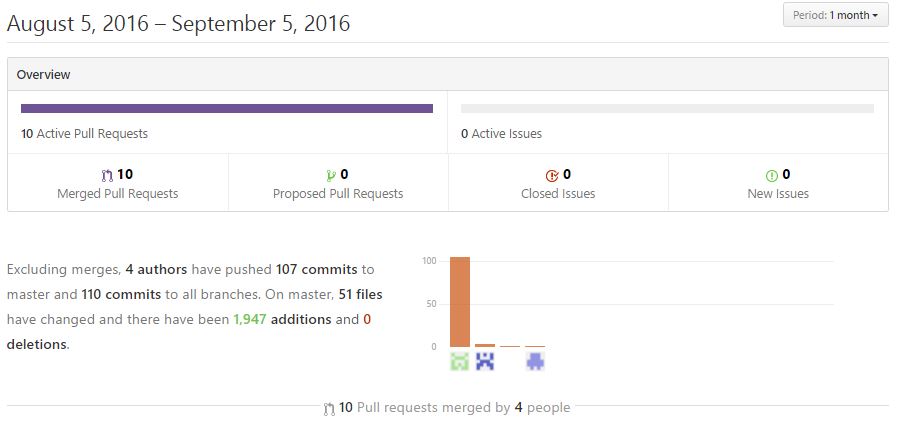
## MEMBERS



## COMMITS



## PULL REQUESTS



...

# APPENDICES

## LAB Q&A

### Week 2 Q&A with Tutor in Lab

|  |  |  |
| --- | --- | --- |
| **Question #1** | | **Category:** Client Information |
| **Questions** | What type of client organization is it? | |
| **Answers** | It is an adventurer capitalist type of organization where the client buys systems with a lower price due to cheap labour and sell it to their clients at a higher price | |

|  |  |  |
| --- | --- | --- |
| **Question #2** | | **Category:** Platform |
| **Questions** | What is the platform that the system should be developed on? | |
| **Answers** | Web-based and Desktop PC | |

|  |  |  |
| --- | --- | --- |
| **Question #3** | | **Category:** User Interface / Platform |
| **Questions** | What type of User Interface (UI) should the system be implemented in? | |
| **Answers** | Most bonus is awarded to systems with a web deployment, and least bonus for command line interface.  It is encouraged to use open source code for front-end development | |

|  |  |  |
| --- | --- | --- |
| **Question #4** | | **Category:** Programming Language |
| **Questions** | What programming languages is required for the implementation of the system? | |
| **Answers** | Object-oriented programming language is necessary, therefore, it should be either C++ or Java.  It is a must to use our own code for back-end development. | |

|  |  |  |
| --- | --- | --- |
| **Question #5** | | **Category:** Registration |
| **Questions** | Is creating a user account mandatory? What are the requirements for creating a user account? | |
| **Answers** | Registration is not mandatory.  Public users can use software system but with restricted access (Read Only)  It is necessary to create an account and login into that account for bug reporting and to access any other features of the software system. | |

|  |  |  |
| --- | --- | --- |
| **Question #6** | | **Category:** Registration |
| **Questions** | What information is required when creating a user account? | |
| **Answers** | It is up to us to determine the information required, but it should be enough information to uniquely identify an account, such as email, username, and password. | |

|  |  |  |
| --- | --- | --- |
| **Question #7** | | **Category:** Database |
| **Questions** | How should the database be stored? | |
| **Answers** | Database should be stored locally on a computer, and don’t do distributed database. | |

|  |  |  |
| --- | --- | --- |
| **Question #8** | | **Category:** Database Security |
| **Questions** | How secure should the passwords stored in the database be? | |
| **Answers** | Plain text is fine, but password hashing can be considered a stretch goal. | |

|  |  |  |
| --- | --- | --- |
| **Question #9** | | **Category:** User Group |
| **Questions** | What is the work flow between user groups? | |
| **Answers** | Public User   * Register, login, logout, and retrieve password * Reads about bugs * Search bugs catalogue * *(No account registration required - Read Only Access)*   Bug Reporter   * Report bugs, comment on existing bug reports * *(Bug Reporter and user groups below require account registration as they have more than just Read Only Access, as they are given different Write permissions for different user groups)*   Triage   * Distributes bug reports to corresponding developer   Developer   * Fix bug * Submit bug report for review * *(Bug status changes to REVIEW status)*   Bug Reviewer   * Reviews bugs and determine if bug has been fixed or not * If bug has been fixed, *changes bug status to COMPLETE* * If bug has *not*been fixed, *transfer bug report to triage, and mark as IN PROGRESS*   System Administrator   * The technicians are the system administrators | |

### Week 3 Q&A with Tutor in Lab

|  |  |  |
| --- | --- | --- |
| **Question #1** | | **Category:** Client Information |
| **Questions** | What types of characteristics should the system have in order to meet the user’s adventurer capitalist organization’s needs? | |
| **Answers** | The system should be open source, customizable, and sellable | |

|  |  |  |
| --- | --- | --- |
| **Question #2** | | **Category:** Client Information |
| **Questions** | How does the bug tracking system integrate with the organization’s system? | |
| **Answers** | The bug tracking system is a standalone system, therefore, there is no need to take into consideration of the effects of the implementation of a bug tracking system within another organization. | |

|  |  |  |
| --- | --- | --- |
| **Question #3** | | **Category:** User Group |
| **Questions** | Who manages the user groups within the system? | |
| **Answers** | User Administrator manages the user groups and can promote or demote users to another user group.  *(Required)* There should be at least *One* user administrator at all times.  *(Optional)* User Administrator can have a different login page separated from the normal users.  *(This user group is created by the technicians)* | |

|  |  |  |
| --- | --- | --- |
| **Question #4** | | **Category:** Registration / User Group |
| **Questions** | Upon registration, how do they choose which user group they belong to? | |
| **Answers** | After registration, all users are considered bug reporters and only have access to functionalities given to bug reporter user group.  It is up to the user administrator group to promote or demote the user to another user group. | |

|  |  |  |
| --- | --- | --- |
| **Question #5** | | **Category:** Database Security |
| **Questions** | Who can access the database? | |
| **Answers** | There should be no direct access to the database by any of the user group including user administrator. It is up to the technician to manage the database. | |

|  |  |  |
| --- | --- | --- |
| **Question #6** | | **Category:** Database |
| **Questions** | Do we create our own catalogue of bugs? | |
| **Answers** | No, a data file will be given for bug descriptions. | |

### Week 4 Q&A with Tutor in Lab

|  |  |  |
| --- | --- | --- |
| **Question #1** | | **Category:** Project Information |
| **Questions** | What should be included for the detailed plan for the whole project? | |
| **Answers** | Break into development plans, deliverables, brief descriptions, look at requirements in order to determine plan | |

|  |  |  |
| --- | --- | --- |
| **Question #2** | | **Category:** Project Information |
| **Questions** | How should version control be used within the inception/elaboration phase?  Example: Google docs | |
| **Answers** | Cannot consider google docs, use git for docs | |

|  |  |  |
| --- | --- | --- |
| **Question #3** | | **Category:** Project Information |
| **Questions** | How should Use Case be organized? | |
| **Answers** | Use case organized based on subsystem or user-centric | |

|  |  |  |
| --- | --- | --- |
| **Question #4** | | **Category:** User Interface |
| **Questions** | What requirements should be considered for the user interface? | |
| **Answers** | The user interface should basically be functional, and as long as it looks okay.  It is a good idea to implement a way in the user interface to quickly find information. | |

|  |  |  |
| --- | --- | --- |
| **Question #5** | | **Category:** User Interface |
| **Questions** | In what way should the bug be displayed? | |
| **Answers** | It does not matter where to show bug descriptions and status, as long as the screen flows well. | |

|  |  |  |
| --- | --- | --- |
| **Question #6** | | **Category:** User Group |
| **Questions** | Who can comment on bug reports? | |
| **Answers** | Everyone has the ability to comment, except the public user. | |

|  |  |  |
| --- | --- | --- |
| **Question #7** | | **Category:** User Group |
| **Questions** | Who gets to have the final decision regarding the user group that the users are assigned to? | |
| **Answers** | Only user administrator can determine roles, and the final decision is decided by the user administrator. | |

|  |  |  |
| --- | --- | --- |
| **Question #8** | | **Category:** User Group |
| **Questions** | Can user administrator delete an account? | |
| **Answers** | User administrator can deactivate user account.  It is a bad idea to delete the account. | |

|  |  |  |
| --- | --- | --- |
| **Question #9** | | **Category:** Reputation |
| **Questions** | How to rate a bug report? | |
| **Answers** | It is up to the discretion of the triager to decide the rating for a bug report  There is a rating of 0 - 5.  0 - no detail  5 - lots of data, date, time, version of s/w, details on what the user was doing etc.. | |

|  |  |  |
| --- | --- | --- |
| **Question #10** | | **Category:** Reputation |
| **Questions** | How should rating system work? | |
| **Answers** | Rating system should be like this:  -Developer fix is rated by the reviewer  -Reporter bug report is rated by the triager. | |

|  |  |  |
| --- | --- | --- |
| **Question #11** | | **Category:** Reputation |
| **Questions** | What criteria is there for reputation system? | |
| **Answers** | There is a different reputation system for developer and bug reporter.  There should be factors that are modifiable by super administrator to determine amount of reputation to give, based on those factors.  Amount of reputation is automatically given based on values set for those factors.  There should not be negative reputation by default. However, it should be possible if super administrator wants it. | |

|  |  |  |
| --- | --- | --- |
| **Question #12** | | **Category:** Report Generation |
| **Questions** | Who can access the statistical reports? | |
| **Answers** | Bug report can be accessed by any user, even the public, that is if a report is to be shown in a publicly accessible place.  It should show the stats of the bug tracking software and should be displayed on the web page if possible. | |

|  |  |  |
| --- | --- | --- |
| **Question #13** | | **Category:** Report Generation |
| **Questions** | How should the report be represented? Pdf? Web page? etc.. | |
| **Answers** | It can be any, but the most desired one is on the web page and real time. Also, the minimum requirement of this report is that it is analytical, that is, it has had some processing applied to the data which makes the data meaningful. For example, bugs reported per week, | |

|  |  |  |
| --- | --- | --- |
| **Question #14** | | **Category:** Report Generation |
| **Questions** | Where should statistical reports be generated in the website, and what is the interval between updates? | |
| **Answers** | Report generation can be displayed in user profile, or in different locations.  It should update automatically in intervals, preferably 1-hour interval.  It should also display live data and showing data in graphs is preferred. | |

|  |  |  |
| --- | --- | --- |
| **Question #15** | | **Category:** Database |
| **Questions** | Where can we obtain a list of users for the system? | |
| **Answers** | There is a challenge in extracting the user list from xml data.  Password and email are not provided in the xml file.  Roles are not provided in the xml file. | |

### Week 6 Q&A with Tutor in Lab

|  |  |  |
| --- | --- | --- |
| **Question #1** | | **Category:** Registration |
| **Questions** | Is there a need for password reset as a use case of the public user? | |
| **Answers** | There is no need for password reset | |

|  |  |  |
| --- | --- | --- |
| **Question #2** | | **Category:** User Group |
| **Questions** | What other tasks can registered user do? | |
| **Answers** | Registered users can subscribe to a bug report that is not theirs. | |

|  |  |  |
| --- | --- | --- |
| **Question #3** | | **Category:** User Group |
| **Questions** | What is the unit of focus in the bug tracking system? Bug or Bug Report | |
| **Answers** | All bug reports describing a bug is considered duplicate and should be merged under one branch, where the bug report that was submitted earliest will be the master bug report.  Therefore, the unit of focus is on the bug itself, that comprises of many bug reports. | |

|  |  |  |
| --- | --- | --- |
| **Question #4** | | **Category:** User Group |
| **Questions** | Expanding on triager, what other tasks can users in this user group do? | |
| **Answers** | Triagers can:  -review bug report and make sure bug report has technical information  -add technical keywords to bug report  -assign bug to developer  Triager should also have the technical knowledge, that helps him:  -guess the components of bug report  -identify priority/severity of bug | |

|  |  |  |
| --- | --- | --- |
| **Question #5** | | **Category:** User Group |
| **Questions** | Does the developer need a download code function? What other tasks can developer do? | |
| **Answers** | There is no need for a download code function for developer.  Developer can upload bug fixes by attaching it in comments. These comments will be given priority when showing comments.  Expanding on the tasks that developers can do:  -A collaboration/communication channel between developer, such as messaging, can be considered as a stretch goal. | |

|  |  |  |
| --- | --- | --- |
| **Question #6** | | **Category:** Reputation |
| **Questions** | How are reputations for bug reporter and developer calculated? | |
| **Answers** | It is calculated automatically, based on factors set by administrator, NOT set manually. Factors include:  -bug report importance  -over 180 characters  -file attached  -priority  Note: developers can get negative reputation | |

|  |  |  |
| --- | --- | --- |
| **Question #7** | | **Category:** Reputation |
| **Questions** | How will the factors for the reputation system work? | |
| **Answers** | The factors can be hardcoded, and can be set by administrator. | |

|  |  |  |
| --- | --- | --- |
| **Question #8** | | **Category:** Domain Model |
| **Questions** | Is the class diagram required, and what is the difference between class diagram and domain model? | |
| **Answers** | Class diagram is not required as it is only needed in elaboration phase, however, domain model is necessary in inception phase.  Domain model is an open design describing the major classes. | |

### Week 7 Q&A with Tutor in Lab

|  |  |  |
| --- | --- | --- |
| **Question #1** | | **Category:** Project Information |
| **Questions** | What type of evidence is required regarding the use of version control? | |
| **Answers** | Pull Requests, Commits, and Members are required. | |

|  |  |  |
| --- | --- | --- |
| **Question #2** | | **Category:** Project Information |
| **Questions** | Is candidate architecture required as an inception activity? | |
| **Answers** | No, even though it is part of inception phase, it is only necessary in elaboration phase. | |

|  |  |  |
| --- | --- | --- |
| **Question #3** | | **Category:** Project Information |
| **Questions** | Where should meta-report be placed in documentation? | |
| **Answers** | It should be placed after the full report. | |

|  |  |  |
| --- | --- | --- |
| **Question #4** | | **Category:** Project Information |
| **Questions** | What should be included in appendices, and where should it be placed in documentation? | |
| **Answers** | It can include Q&A from labs, and be placed after meta-report.  It is not required to be added but helps to clarify certain aspects of information in report, that would otherwise have marks deducted. | |

# MEMBER CONTRIBUTION ASSESSMENT

|  |  |  |
| --- | --- | --- |
| **Member** | **Contribution** | **Signature** |
| Jingwang Teh | Contributed |  |
| Weicheng Yin | Contributed |  |
| Luke O’Callaghan | Contributed |  |
| John Chun-Yu Chan | Contributed |  |