Software Requirement Specification
for
Bug-Tracker
Version 1.0 approved

Jesse Wood

January 11, 2020

Contents

1 Introduction

1.1 Purpose

The product whose software requirements are specified in this document is a Bug-tracker. Bug tracking is a process of logging or monitoring bugs during software development (IBM 2020). The software is a stand-alone system to be used as a bug-tracker while developing a project.

1.2 Document Conventions

In the context of this document, each of the requirements will be listed in decending priority. Fundamental requirements for this software will be explored in more detail than others. This document follows the IEEE SRS conventions.

1.3 Intended Audience and Reading Suggestions

The intended audience for this document is academics, developers and users. This document is set out sections defined in the table of contents Contents, which explore different aspects of the systems requirements. Users of this software could find themselves only interested in a few sections, such as the User Documentation or System Features. Where as academics and developers may find the contents of the entire document to be relevant if they are looking into a similar software.

1.4 Product Scope

This software is designed to track bugs in projects. Bug-tracking is a useful way to develop large scale software and/or develop software in a team. A goal of this project will be to follow the Model View Controller design pattern (Gamma 1997). Using a design pattern will make the software reusable and easier to develop further in the future. Expanding on the MVC model, maximising the modularity of this software will help make the project adaptable and promote further re-use in the future.

1.5 References

Each of the following references are available in the form of an online pdf. Note that the availibility of website references may change subject to time.

References

Gamma, E (1997). *Design Patterns, Elements of Reusable Object Orientated Software*. Pearson Education (US).

IBM (Jan. 2020). *Bug-tracking, Improve product development by tracking software errors and defects*. URL: https://www.ibm.com/topics/bug-tracking.

Martin, Robert (2008). *Clean Code, A Handbook of Agile Software Craftsmenship*. Pearson Education.

OpenSource (Jan. 2020). *Open Source Initiative, The MIT License*. URL: https://opensource.org/licenses/MIT.

2 Overall Description

2.1 Product Perspective

The development of this software originated out of necessity. One of the most crucial and time consuming tasks of any software development is the debugging process. As software developers we plan to mimise the prevelance of bugs within our systems, however hard we may try, they will still perculate throughout our code bases. Bug-tracking is a process of managing bugs in a system into a priority hierarchy, such that each bug can be addressed with respect to the severity of its impact on the system. Another important part of a bug-tracker is the ability to change the statis of a bug (e.g., when solved), this allows the workflow of a project to be monitored so that duplicate resources aren't assigned to fixing the same issue.

2.2 Product Functions

OBJECT CLASS DIAGRAM HERE

- Add a project
- Add a bug to a project
- Change status of a bug
- Log-in

2.3 User Classes and Characteristics

Author
Collaborator
Audience

INCLUDE USE CASE DIAGRAMS HERE

2.4 Operating Environment

The operating environment for this software will be as follows. The software will be deployed in the form of a web app that relies on html/css/js framework To Be Decided (TBD). These frameworks are constantly updated, so the software must be maintained over time, to ensure that future releases of the framework used do not introduce bugs. As long as the users' browswer supports the framework of the software, this system will be able to be run accross multiple operating systems. Therefore, only one version will have to be released. A more technical version of the software could be implemented for a terminal based environment for advanced use later if needed acrshorttbd.

2.5 Design and Implementation Constraints

The TBD database system will be used to store the user informaiton. This introduces contraints to the storage capacity for user information. The amount of information stored cannot exceed the free-tier limit provided by the TBD database provider, or else the software will no longer be able to introduce new information.

Using a database makes the software vulnerable to security exploits such as SQL Injection (**sqlInjection**) and XSS (**xss**). The projects, bugs and user account informaiton will have to be stored in a database. This is likely to hold sensitive information, especially the user information, therefore it is of paramount importance to ensure the software in secure against possible security exploits such as these.

The software will follow the design convention of MVC. The Model/View/Controller is a very popular design pattern aimed at increasing flexibility and reuse Gamma 1997. Developers intending to contribute to this project will have to follow the same design convention in any of their additions to the code base.

It would be expected that the software also follows the important principles of writing clean code (Martin 2008). Crafting software using these principles is pivotal to maintaing usable codebase.

Test driven development is a programming standard that is required during the development of this software. The increasingly popular DevOps approach it crucial in software development. Keep the test suit for this software up-to-date and all encompasing is of paramount importance to proving the functionality of the software.

2.6 User Documentation

The high-level components of the software will be documented in the source code. Following the documentation standards mentioned on the previous section **??**, the code-base of the software is well-documented for other developers.

TBD Wiki eh. The code will have a Wiki available on the Github VCS for the software. This will explain the purpose for each of the individual components of the software at a high-level, with the in-line documentation providing further explanation when required.

TBD Help section. On the web-page there is a help section. This includes FAQ with links to the Wiki for the relevant information a user and/or developer may require.

2.7 Assumptions and Dependencies

The development environment for the software are as follows.

Javascript frameworks,

browser dependencies,

Vim is the text editor used to develop the software. Vim is a free and open source software. With the needed technical expertise, anyone can continux to develop this software, with no commerical software needed.

The project will be released under an MIT license. Therefore the reuse of the code and further development will follow the guidelines stipulated by this license (OpenSource 2020). It is assumed that developers and users of this software will adhere to the rules of the license it was released under.

Git is the Version Control System used to document the development of the project. It is assumed that any further development will follow git ettiquette. Improvements and re-works to the software can be forked of the master branch. Given the license, previously mentioned, further development of this software from other developers is welcome and encouraged.

3 External Interface Requirements

3.1 User Interfaces

The user interfaces for this software will be designed to work in a browser. The web app intefaces will be designed with mobile use considered. Therefore all of the following interfaces will also have cross-compbatibility with browsers on mobile devices.

There is a user interface for the log in screen. This will allow the user to log in to their account. Google verification will be provided should a user not want to create their own account

Another interface is the project menu. This allows the user to navigate to a particular project. With the possibility to create a new project, or find an existing one.

Each project has its own dashboard. On this interface a user can add/change the status of bugs. View the history of a project. Search for a bug within a project.

The last interface will be for the author of a project. Given the correct log in credentials a user will be able to edit information about a project. Such as whether it is public or private, its name, remove a project ... etc.

3.2 Hardware Interfaces

The hardware interfaces for this software include all devices that support browswers that support the TBD framework. This includes devices that run the an up to date version of Chrome and Firefox. Therefore this software supports hardware interfaces of Desktops, Andriod, Apple as well as the operating systems of Mac OS, Windows and Linux.

3.3 Software Interfaces

TBD The software requires a connection to an external database. This stores all the information regarding the projects and their bugs. Using this software interface requires the use of queries as well as read and write access to the database. The database is able to concurrently accessed by multiple users. Using a robust DBMS ensures that this database can be implemented.

Another software interface used is the TBD framework. Libraries from this framework are used to implement the GUI and query/read/write to the database. The software is implemented in a way such that all communication between the database and framework uses an interface. This allows the code to be refactored easily should the choice of database change in the future. Using the MVC helps,

such that the interactions with the database can be compartmentalized into the model component.

3.4 Communication Interfaces

The communication interface of email TBD is used to allow notificaitons for projects. This includes letting the user know when new bugs are contributed or if the status of a bug changes. The user can edit the thresholds for certain notifications in the settings GUI.

4 System Features

4.1 Project

A project represents a code base that a user intends to monitor bugs on. These can be added/removed/edited from/on the database.

4.2 Bugs

A bug represents a software error. It is a brief description of a problem accompanied with a state and a priority.

States for software bugs include:

- Active
- Test
- Verified
- Closed
- Reopened

Priorities for software bugs include:

- Catastrophic
- Impaired Functionality
- Failure of non-critical systems
- Very Minor

(IBM 2020)

4.3 View History

This feature allows a user to visualize the history of a project. A basic timeline of changes to the project. Analytics such as graphs of bug fixes over time, and average priority or frequency of bugs.

4.4 Log-in

This feature allows the user to log in and access their projects. This has Google authentication and also the ability for a user to register their own account, and sign in. This feature requires encryption, such that the users sensitive information such as emails and passwords are not leaked.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

5.2 Safety Requirements

Secure log in authentication is a safety requirement for this software.

5.3 Security Requirements

End to end data encryption for private projects for users

Encryption of user log-in information

Database must be secure from SQL Injection and XSS

5.4 Software Quality Attributes

Modularity - MVC

Testablity - TDD

Maintainability - Well-documneted

5.5 Business Rules

Author ... Delete and Rename projects

Author and collaborators ... see, add bugs, change status for private projects

Audience ... cant edit or post bugs without relevant permissions

6 Other Requirements

6.1 Glossary

Acronyms

DBMS Database Management System. 6

GUI Graphical User Interface. 6, 7

MVC Model View Controller. 2, 6, 9

SQL Structured Query Language. 9

TBD To Be Decided. 3, 4, 6, 7

TDD Test Driven Development. 9

XSS Cross Site Scripting. 9

6.2 Analysis Models

6.3 To Be Determined List