Software Requirement Specification
for
Bug-Tracker
Version 1.0 approved

Jesse Wood

January 12, 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

This document follows the IEEE SRS convention (Taafe 2020). Fundamental requirements for this software will be explored in more detail than others.Each of the requirements will be listed in order of descending priority. All references can be found below in the References. Acronyms and technical terms can be found in the Glossary. All of the UML diagrams used throughout the document were made using PlantUML (PlantUML 2020).

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

Cooper, Allen (2014). *About Face: The Essentials of Interaction Design, 4th Edition*. Wiley.

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

GeeksForGeeks (Jan. 2020). *What are Hash Functions and How to choose a good Hash Function*. URL: https://www.geeksforgeeks.org/what-are-hash-functions-and-how-to-choose-a-good-hash-function/.

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

IETF (2011). *TOTP: Time-Based One-Time Password Algorithm*. Internet Engineering Task Force.

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.

OWASP (Jan. 2020). *Cross-site Scripting (XSS)*. URL: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS).

PlantUML (Jan. 2020). *PlantUML in a nutshell*. URL: https://plantuml.com/.

Taafe, Ed (Jan. 2020). *Software Requirements Specification*. URL: https://en.wikipedia.org/wiki/Software_requirements_specification.

w3schools.com (Jan. 2020). *SQL Injection*. URL: https://www.w3schools.com/sql/sql_injection.asp.

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 status 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
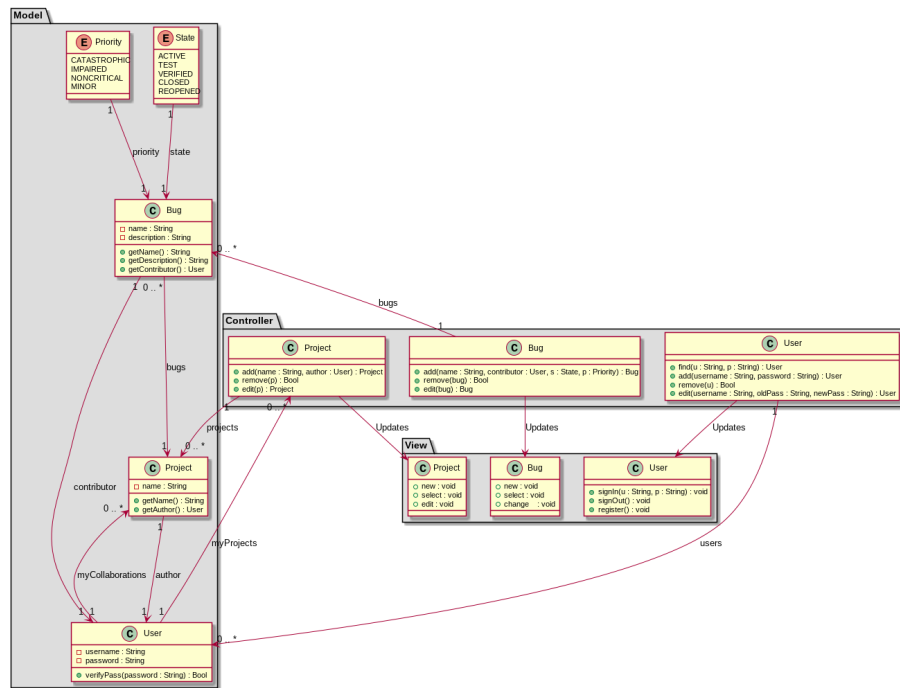
## 2.3 User Classes and Characteristics

This software has three main user classes. The *Author*, *Colloborater* and *Audience*. Each of these classes with have differently privelages related to a project.

An *Author* has permission to delete and edit a project at will. They can also

contribute bug reports and change the scope of a project from public to private or vice versa. An author is able to add/remove collaborators from a project, and remove any bug reports on their own projects.

A *Collaborater* is an actor that has been granted permissions by the author. These privaleges allow the user to contribute bug reports to a project. However they cannot edit or delete the project itself.

The *Audience* includes users that do not come under the aforementioned actors. They are able to view public projects. However they cannot contribute bug reports to these projects. Also they cannot view private projects.
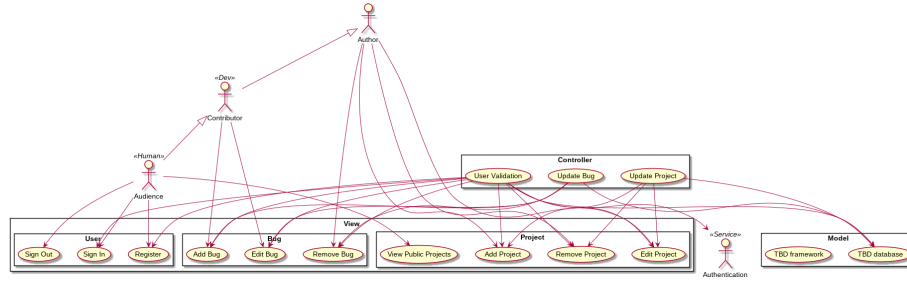
2.4 Operating Environment

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 TBD.

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 (w3schools.com 2020) and XSS (OWASP 2020). 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 Design and Implementation Constraints, 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

Javascript frameworks,

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. The user interface will be designed with usability being the primary focus. It will implement some common UI design conventions discussed in literature (Cooper 2014)

The log in screen is the first interface the user will see. 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.

Their will be a settings interface for the authors of projects. 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 software supports any device that supports a TBD browsers. It can run on Desktop, Laptop and Mobile devices so long as they support the TBD framework.

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 Bugs

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

Stimulus/Response Sequences

- create bug with a priority and state

- change the state of a bug

- record the bug contributor, time, etc ...

Functional Requirements:

- Each bug has one state and priority

- Each bug is unique

- Bugs store information about state changes

- Bugs have a contributor

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)

4.2   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.

Stimulus/Response Sequences:

- create project

- add/remove collaborators

- add bugs to the project

- View History of bugs

- View Project information (collaboraters/author/date initiated)

Functional Requirements:

- Collaboraters/Author can edit project

- Author can add/remove collaboraters to project

- Author can edit details of the project

- Bugs can be added/removed from a project

4.3   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.

Stimulus/Response Sequences

- A user can enter a username and password

- Google Authentication

- Register an account

- A user can log in

Functional Requirements:

- No duplicate accounts

- Strong passwords

- store the account information securely

- Account is linked to its own and collobaration projects

5 Other Nonfunctional Requirements

5.1 Performance Requirements

The database must be able to handle concurrency. Since there will be multiple users attempting to access the information at the same time. Modern RDBMS like MySQL or PostgreSQL handle this. The rationale behind this is to allow multiple users to access the software at once. These users need to have the same results if performing the same actions.

5.2 Safety Requirements

Secure log in authentication is a safety requirement for this software. The software will have to meet the safety certifications Google requires to include Google Authentication. This requires the software to meet Time-Based One-Time Password (TOTP) algorithm specifications (IETF 2011). This can be implemented using Javascript (JS) frameworks. It is a requirement for this software. Users find software with Google Authentication more trustworthy and reputable.

5.3 Security Requirements

End to end data encryption is requried for private projects for users. If a user sets a projects scope to private, it means for whatever reason they do not wish to make the contents of this project publicly available. It is in the best interest for developing this software that all the information regarding private projects remains encrypted. It will require a key only available to the author of the project to decrypt the projects contents. This means should the private projects on the database leak, they will still remain encrypted in a non-human readible format.

Encryption of user log-in information. It is important that the users' sensitive information, such as their passwords are stored on the database in an encrpted from. The authentication service can perform a Hash Function (GeeksForGeeks 2020). This ensures that the password does not have to stored as a string on the database. Instead it can be stored as a hash. Should the sensitive user information on the database be released, the passwords will still remain encrypted.

Database must be secure from SQL Injection and XSS. To prevent database leaks from even happening in the first place the program is required to be secure. For example, to prevent SQL injection any user input that interacts with the database must be sanatized. This involves checking for hidden SQL code within user input, and sanitizing the input, before allowing it to be entered into the database.

5.4 Software Quality Attributes

The software demonstrates modularity through the use of the MVC design pattern (Gamma 1997). This design pattern is an excellent example of modularity in a code base. It uses abstraction to seperate the software into three distincts modules which. The Model View Controller (MVC) modules make future development of this software easier, even for new developers. So long as they are familiar with this common design pattern.

Testability is perhaps one of the most important aspects of developing reliable software. Through the use of TDD, this software creates its own test suite during development. This test suite helps to demonstrate the functionality of the software. It make integration testing new features for un-intended side effects a piece of cake, since the test suite is already written. Using TDD is one of the most effective ways to demonstrate that the software meets its own requirements.

Maintainability is a quality attribute that software inherits through modularity and testability. Focussing on modularity by using Model View Controller (MVC) means that each module has low coupling. The test is written first, then the code to pass it. This means that extra functionality or dead code will not be prevelent in the code base. This in combination with the modularity will ensure high cohesion.

5.5 Business Rules

The individuals who can perform specific functions is explained in detial in this previous section, User Classes and Characteristics. To reiterate the business rules for this software the reasoning behind this will be explained.

An *Author* can Delete and Rename Projects and Bugs. This rule applies only to the projects they have created. This privelage is extended to the user such that they can manage their own projects. Colloboraters do not have these permissions because an author is the only actor that should be able to decide whether or not a project is needed.

Authors and *Collaborators* can see, add bugs, change status for private projects. Some projects may have more than one developer. The author is able to add one or more collobaraters to their own project. This allows a team to track bugs for a project. This functionality is not extended to the audience, this is because an author may not welcome the input of random contributors.

Lastly, the *Audience* cant edit or post bugs without relevant permissions. They still have the ability to view the history of public projects. This adds a community aspect to the software. Users can see if other contributors have solved the same or similar bugs to what they are currently experiencing.

6   Other Requirements

6.1   Glossary

Acronyms

CSS Cascading Style Sheet. 6

DBMS Database Management System. 9

GUI Graphical User Interface. 9, 10

HTML Hypertext Markup Language. 6

IEEE Institute of Electrical and Electronics Engineers. 3

JS Javascript. 6, 13

MVC Model View Controller. 3, 7, 9, 14

RDBMS Relational Database Management System. 13

SQL Structured Query Language. 6, 13

SRS Software Requirement Specification. 3

TBD To Be Decided. 6, 7, 9, 10

TDD Test Driven Development. 14

TOTP Time-Based One-Time Password. 13

XSS Cross Site Scripting. 6, 13

Glossary

bug An error or flaw in software that produces unintended side effects . 3

concurrency Multiple computations are happening at the same time . 13

database A stuctured set of data stored on a server . 6, 9, 10, 12, 13

debugging The process of identifying and removing errors from software . 5

encryption Encoding a message or information so that only authorized parties can access it. 12, 13

hash A function that converts one value to another . 13

key Random string created for scrambling/unscrambling data . 13

maintainability Software that exhibits high cohesion and low coupling . 14

modularity Software which is seperated into independent modules. 14

MySQL MySQL is an open-source relational database management system . 13

PostgreSQL Free and open-source relational database management system . 13

string A datatype in software, an array of characters. E.g., "Hello World!". 13

testability Testing is made quick, easy and enjoyable . 14

6.2   Analysis Models

6.3   To Be Determined List

TBD To Be Decided. 6, 7, 9, 10