Project Proposal
Level 2 Industry Based Project

Mantis - Multi-tenant Issue Tracking and Reporting System

The Court of Owls

Faculty of Information Technology
University of Moratuwa
2020

Index Number	Name
184025L	M.D.N. Chandrasiri
184038E	E.A.Y..R. Edirisinghe
184116R	W.P.C.P. Pathirana
184152X	W.A.D. Sandarekha
184157R	I.K.G.D.S.N. Senanayake

Supervisor Name : Dr. Priyanga D. Talagala

Table of Contents

Introduction

An error can be defined as a deviation from accuracy[7]. In mass systems a software bug is a failure or a drawback in a system that directs to an unexpected result or time-consuming effort. Some of the common categories of bugs are

- Functionality errors - Errors which can be occurred with complexity or by mistakes
- Communication errors - Miscommunication between 2 parties
- Syntactic Errors - Errors with misspelled words, Incorrect wordings
- Missing command errors - Missing command errors
- Error handling errors - Errors occurring at the usage stage
- Calculation Errors - Data type mismatching, Syntax errors
- Control flow Errors - Failures in step by step process [7]

Reporting these kinds of errors immediately is essential to get the best result out of the system. Therefore, defect the issue, categorize them, reporting them are the best practices in the quality control stage [7]. Issue tracking is a key component of the System Development Life Cycle until the post-delivery phases in the project. There are several advantages in error and issue handling systems

- Dividing the issues according to the scope
- Tracking the time consumption in the developer side
- Managing the impression of customers
- Clear reporting system can be implemented

To fulfill this requirement there are several issue tracking systems in the industry such as JIRA, Bugzilla, Github issues, Trello. However, none of the products can provide a complete solution to make a platform to combine the customer and developer side approach. Therefore, here we propose a solution which includes customer friendly issue/bug ticketing system with onboard adding, management of bugs using “Kanban” boards by developers, time tracking systems, bug pool system, easy reporting system without using any third party tools.

Background and Motivation

Bug tracking is the process of capturing, reporting, and managing data on bugs that occur in software projects.[1]We examined several research papers relevant to this area and identified some major issues faced by current software developers and clients.

Bug report reassignments are a common part of the bug-fixing process. It is useful to determine the best person to fix a bug. But the popular view is that inefficient reassignments are harmful. There are some primary reasons for reassignments such as finding the root problem of the bug, determining the ownership, poor bug report quality, workload balancing.

[1]Most common reason for reassignments is because people do not identify the root cause problem that they are willing to fix. A concept related to root cause is 'ownership'. It defines the status (easy, medium, hard) of the bug and which team (Administrator, Developer, Manager, QA, Manager) is responsible for this bug. In large software projects such as Windows Vista, ownership of components can often be unclear. Therefore, physically all of them cannot get involved with the bug fixing process.

Bug reporting quality is also an important consideration in bug tracking systems. If a bug report is poorly written or contains too little information, it will take more time to fix the bug. Therefore, developers cannot fix it as the client needs.[1]

Once a bug gets a team to fix it, some developers might be busy with other tasks. So, they will be handed over to their teammates. But if they are also busy, they need to be reassigned again and again. It can be beneficial since bugs might get fixed sooner. But it can be very easy for everyone to join the process in one system like bug tracking and reporting web applications.

Sometimes bug reports should contain private user data. This may cause the client's and developer's privacy. Therefore, we should submit error reports that allow developers to fix the bugs while not disclosing any personal information.[6]

At the fixed state of the bug, many follow up questions asked by the bug reporters, and often they do not play an active role in the discussion of the bug. Also, reporters do not engage with the user community in the bug fixing process and do not keep them updated about the status of the bug by the developer. This is the most important issue of bug tracking. If we can avoid this problem, we can provide a good service to our clients and keep them updated on the process.[5]

When clients get a software failure, they have an option of submitting a bug report that provides information about the failure (how it happened) to the developer. If the bug report contains enough information, the developer can try to review it. Unfortunately, most of the bug reports created by the clients are usually large. It will take a long time to analyze and identify issues. If we can generate important tags, bug status, relevant media files, it will be more efficient than before. [4,3,2]

There are many popular SaaS-based bug tracking systems such as JIRA, Bugzilla, Github issues. Even the several competitive products in the market, none of them provide solutions for the above problems. Most systems are either developer side or client side and they are not user friendly. Therefore, we are interested in developing a software product to solve those problems and provide a better debugging platform for developers and clients.

Problem in brief

Main purpose of a bug tracker is creating a method to report bugs to the developer. But some bug trackers now provide facilities to developers to manage their workflow when debugging. The problem is bug trackers now exist are too client focused or too developer focused.

Clients may face problems such as,

- Complex UI
- Not having a simple way to explain their issue
- Not having in built facility to screenshot / screen record facility
- Inability to track the debugging progress of the bugs they reported

From developer's side,

- Not having a proper way to organize the workflow
- Developers should use third party software to create sprints and Kanban boards
- Not having a proper way to refer previously debugged bugs
- Most of the bug trackers do not provide a report after resolving the bug

Aims and Objectives

- Aim
    - The aim of this project is to develop a system to bring all user roles (User/Manager/Developer/QA etc) to one platform to solve the issues/bugs regarding a selected product with the use of latest technologies in the industry.
        - Reactjs
        - Django
        - Django Rest
        - Postgresql
- Objectives
    - Critical review of bug tracking systems
    - In depth study of latest developments in technologies
    - Design and create wireframes for the interfaces
    - Develop separate systems each user roles
    - Evaluation of the proposed solution
    - Preparation of final documentation

Proposed Solution

The multi-tenant issue tracking and reporting system should provide an interactive web-based platform for both internal users and external users.

- Internal Users:
  - Administrators
  - Managers
  - Quality Assurance Engineers
  - Developers
- External Users:
  - Customers

The system involves five main components and each component may involve a generic process or specific schedule in the process of issue tracking. Those five main components and their main objectives within the system can be derived as follows,

1. Authentication and Authorization

   This is a general admin dashboard where we control all the authentication and authorization steps as well as providing permissions to each user in the system. This follows the Role Based Access control(RBAC) method to provide permissions to users. This should have below multi-tenant capabilities,

   - Users can login to the system.
   - Users can make a password request.
   - Allow defining a super admin
   - Allow admin to create users
   - Allow admin to create and provide user roles.
     - Administrators
     - Developers
     - QA
     - Manager
     - Customer
   - Allow admin to define and provide permissions for user roles.
     - Example 1: QA can only view filed bugs and change status.
     - Example 2: Customers can only create an issue and view the progress rather than viewing the developer side of the system.
   - Allow admin to create projects and assign users to the projects.

2. Bug capture log

Bug capture log is an interface where customers can engage with the system. Customers can provide issues to the company according to a given manner and are able to get in touch with the progress of the issues. This should have below capabilities,

- Customers can login into the system with provided credentials.
- Customers can view the project where he/she is a part of.
- Customers can select a project and file an issue.
- Issue filing process can be listed down as follows,
    - Customer selects “Create new Issue”
    - Issue window allows input of a description
    - Issue window allows uploading files
    - Issue window allows screen recordings.

Recording screen is the most vital feature in issue creation. Customers should be able to turn on-screen recording and record the bug. Existing libraries can be used to achieve this feature or can use built-in screen capturing options. Once the recording is done, it should be attached as a video to the issues.

- Customers can comment on created issues if necessary.

3. Bug management system

Bug management system is also a UI which can only be accessed by internal users. This is where bug cycle processes. The generic process of bug management system contains the following steps.

a. Once a user creates an issue it will be visible in the “Backlog” of bug management system which can be taken as a pool of bugs.

b. Managers can review those issues and can add them to a sprint.

c. Within the sprint there is a separate Kanban board for each issue and it will go through 5 Kanban lanes.

- Open (This is the first stage in the issue handling process)
- In-progress
- QA
- Done

d. Manager/Developer/QA have access to the Kanban lane where they can drag and drop issues.

Also bug management system should have following capabilities,

- Internal users can login to the system.
- Internal users can select a project and view bug log of the project
- Internal users can,
    - Add comments to the bug.
    - Change severity and tag of the bug.
    - Assign himself/herself to the bug
    - Assign someone else to the bug
    - Attach files to the bug
    - User can add time spent for a bug (time tracking)

Apart from the above, specific user roles have additional capabilities also.

4. Bug solution pool

Bug solution pool is a knowledge base that stores solved bugs for future references. This allows a smooth knowledge share process in the internal organization. This part of the system should have following capabilities,

- Managers/Developers can mark a resolved bug to be added to the solution pool.
- Solution pool should list all added bugs and should allow full-text search.
- Managers/Developers can add comments, updates to bugs in the solution pool.

5. Bug Reports

Reports should be made to monitor project progress, customer satisfaction in order to aid in decision making. The following reports should be produced.

- Developer timesheet to identify each developer's effort
- Project timesheet to identify the effort put into one project
- Sprint summary to understand the number of issues to be carried out to next sprint
- Gantt Charts
- Full bug summary to identify which proportion of the bugs are resolved or not.

Resource Requirements

Hardware

- 1 GB RAM
- 1 CPU Core
- 24 GB SSD Storage
- 2 TB Transfer
- 40 Gbit Network In
- 125 Mbit Network Out

Web Server

- 1.6 GHz CPU
- 1.75 GB RAM
- 40GB HDD

Technologies

- Reactjs
- Django
- Django Rest
- Postgresql

Reference

[1] Guo, P. J., Zimmermann, T., Nagappan, N., & Murphy, B. (2011, March). " Not my bug!" and other reasons for software bug report reassignments. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work* (pp. 395-404).

[2] Fazzini, M., Prammer, M., d'Amorim, M., & Orso, A. (2018, July). Automatically translating bug reports into test cases for mobile apps. In *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis* (pp. 141-152).

[3] Hooimeijer, P., & Weimer, W. (2007, November). Modeling bug report quality. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering* (pp. 34-43).

[4] Bettenburg, N., Just, S., Schröter, A., Weiss, C., Premraj, R., & Zimmermann, T. (2008, November). What makes a good bug report?. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering* (pp. 308-318).

[5] Breu, S., Premraj, R., Sillito, J., & Zimmermann, T. (2010, February). Information needs in bug reports: improving cooperation between developers and users. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work* (pp. 301-310).

[6] Castro, M., Costa, M., & Martin, J. P. (2008). Better bug reporting with better privacy. *ACM SIGOPS Operating Systems Review*, *42*(2), 319-328.

[7] Neha B. (2020). 7 Types Of Software Errors That Every Tester Should Know. Retrieved from https://www.softwaretestinghelp.com/types-of-software-errors/

Appendix

High Level Architecture Diagram

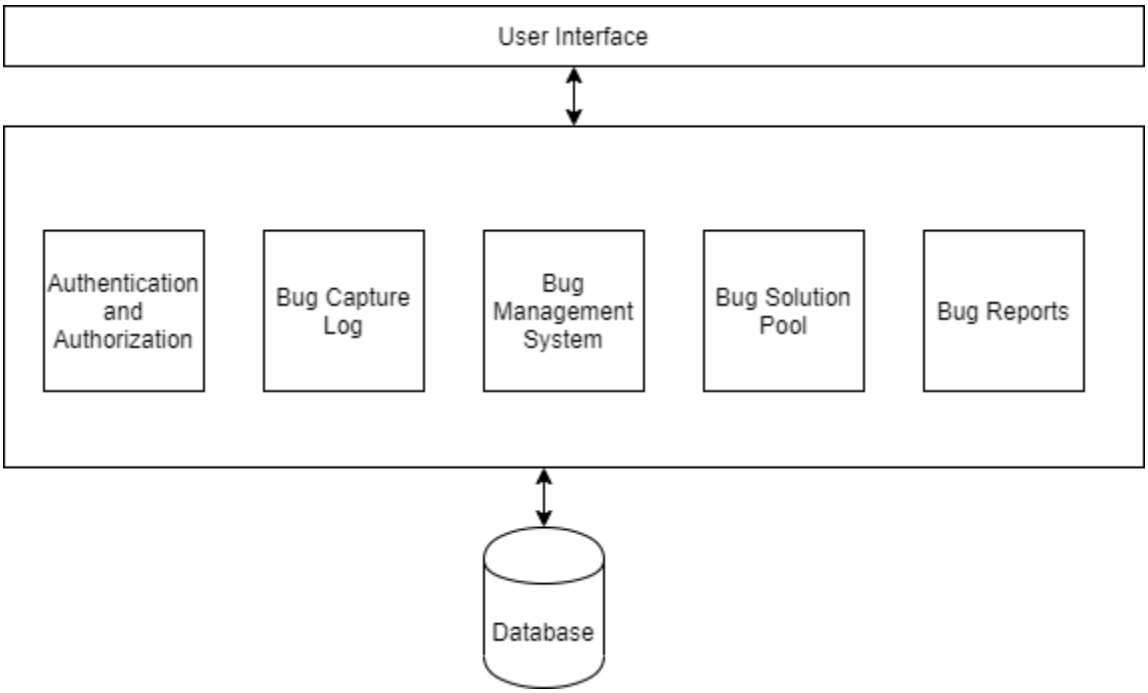

Figure 1 System Diagram

Action Plan

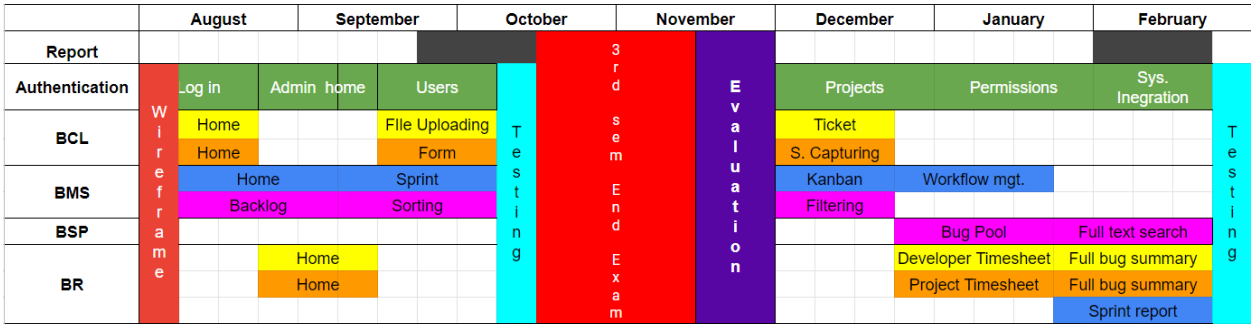

Figure 2 Action Plan

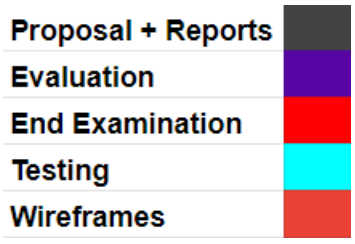

Figure 4 Color Code

Figure 3 Group Member Color Code

https://docs.google.com/spreadsheets/d/14MG1Py0LZbuGtYB32J5oxlwFkn9mUMheGXvFGACs7dE/edit?usp=sharing

Above link will provide a link to the soft copy of the action plan.