
# BUG TRACKER

A Report for the Evaluation 3 of Project 2

*Submitted by*

**SHASHANK SHEKHAR**
**(1613105111 / 16SCSE105079)**

***in partial fulfillment for the award of the degree***

***of***

**Bachelor of Technology**

**IN**

**Computer Science and Engineering with Specialization of**

**Cloud Computing and Virtualization**

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

**Under the Supervision of**

**Dr. P. MUTHUSAMY**

**Associate Professor**

**APRIL/MAY- 2020**

# TABLE OF CONTENTS

# 1. Abstract

There are so many applications and programs which tracks the bugs and issues of a project. They are also used to trace the lifecycle of the projects, tasks and requirements. But there are problems with almost all of them and even when some of them are fine, they are not open sourced. This project is being developed in keeping in mind that the project will have all the essential features of a bug tracker in the initial rollout and then the project will be updated with more features in the future. This project is aimed to simplify different life cycle of a software development project.

In this work the most popular issue or bug tracking systems will be examined and are used as a reference point to collect data about the essentials of a bug tracker and to get acquainted with the tools and then in future some of the features might integrate with this project. This project will have these features like Log an issue, will generate ticket, a dashboard and some of the searching functionalities like search by date or by application.

# 2. Introduction

Change is a constant feature of software development. All projects have their main objective to change something. A system or its parts are upgraded, fixed, or replaced, presumably to provide better or greater functionality, ease of use, reduction of operating expenses, etc.; however, change must be controlled in its introduction to a project. If not properly handled, change can slip the schedule, affect the quality, and even kill the project. As a project draws closer to its completion, the impacts of change are more severe. Clearly, a mechanism is needed to control change. A part of the overall change management approach is called Configuration Management. Configuration Management is the process of controlling and documenting change to a developing system. As the size of an effort increases, so does the necessity of implementing effective Configuration Management. It allows large teams to work together in a stable environment, while still providing the flexibility required for creative work. The key part of Configuration Management is Requirements Management. Requirements Management involves establishing and maintaining agreement between customer and developer on both technical and non-technical requirements. This agreement forms the basis for estimating, planning, performing, and tracking project activities throughout the project and for maintaining and enhancing developed software. Key activities include: planning the requirements phase establishing the requirements process controlling requirements changes minimizing the addition of new requirements tracking progress resolving issues with customers and developers holding requirements reviews. This work is concerned with software tools used in Software Management. They are called in many ways: Issue Tracking Systems, Trouble Tracking Systems, Bug Tracking Systems, Requirements Tracking Systems, etc.; however, their purpose remains the same: collecting requirements, their management, and tracking their progress. The lifecycle of the requirement and roles involved in this lifecycle will be explained. One of the goals of this part should also be to explain the importance of issue tracking. It takes a look at development and management methods like CMMI and ISO standards (which can also be used as metrics of development quality) and how they relate to this theme. The most popular tools today for issue / bug tracking will be examined.

### Project Description

An issue or a bug tracker is an essential tool for any web and software project. In order to make progress with our software projects, we need a simple, yet effective, workflow which allows us to report, document and track errors and failures which our software or website is causing. It will be an issue tracking system specifically designed for software development projects. Issue Tracker is a minimalistic approach for web-based project management systems. It will be minimalistic from a design perspective. But it definitely doesn't lack any key features. With Issue Tracker you can enable to create project roadmaps, milestones and you can even fully modify the issue reporting area to your needs.

### Purpose

Version control and bug tracking systems contain large amounts of historical information that can give deep insight into the evolution of a software project. Unfortunately, these systems provide only insufficient support for a detailed analysis of software evolution aspects. The purpose of this project is to address this problem and introduce an approach for tracking the issues in a web or software project. It will be a comprehensive and complete project management tool that will be easily able to track all the issue and bugs during the development phase of the project which will make the life of developer much easier and also the maintenance of the project will also become faster and easier.

### Motivations and Scope

- Logging the issue: Detect the issue and log it into the system and request for the change and info about the issue.
- Updating the tracking Id: Status of the ticket, it's priority and release date, etc.
- A dashboard which will show all the details and search functionality will be added.

## 3. Existing system

The bugs that are identified by tester in software testing phase are reported to Project Manager and developer through simple shared lists and emails. Most of the companies share this information through document called "Defect report". This procedure is error prone and there is ample chance of leaving some bugs unfixed and ignored as there is no particular tracking system in place. The team involved in the software development life cycle must be aware of the status of each and every bug reported. The existing system fails to fulfill this requirement thus it affects productivity and accountability of every member of team.

### Bugzilla

Bugzilla is probably the most well-known of the open source issue-management tools. It is used on many open source projects such as Mozilla, Eclipse, and many Linux distributions, and is well-adapted to large, open projects.

### User interface

Easily said, Bugzilla UI is strictly functional. There is nothing very nice about it, it provides plenty of functions within a small space, and in the beginning, the user can feel quite uncomfortable and lost; however, after discovering it, the user will find out that it is not very complicated and working with it is straightforward.

### Ticket system

Bugzilla supports a fairly complete, albeit hard-coded workflow model (see diagram below). Although this workflow model cannot be customized in Bugzilla, it usually proves sufficient for most organizations.

### Conclusion

Bugzilla is a tried-and-true solution that supports large projects and user bases. Its workflow features are more than sufficient for most organizations. On the downside, Bugzilla is particularly complicated to install and maintain, and the user interface would not win any prizes for design or usability. Its reporting features will do the job, though they are not particularly user-friendly.

## CodeBeamer

CodeBeamer is not just a simple Issue tracking system, but it is a Collaboration Development Platform with integrated Application Lifecycle Management. Under this title we can imagine set of services such as Project related Document Management, Wiki, Forum, Online chat and of course Issue tracking integrated with version control (SVN). It's available for free for students and evaluators; however, for business use it is commercial. As an example of public use, we can name for example JavaForge.com community, which host multiple Java open-source projects and is running above CodeBeamer.

### User interface

The user interface looks to have a better structure than Bugzilla UI; however, it still would not win the price for usability. As you can see below in the use-case scenario, it is necessary to go quite deep into the UI to add new issue requests. It would help to move some most used functions to some extra panels, or other similar reorganizations. While using CodeBeamer UI, there was no problem to find anything, only many things involved too many steps.

### Ticket system

CodeBeamer provides a default workflow, which can be customized within the user interface. The workflow diagram below contains the following node types:

Yellow node - denotes the start state. A new tracker item will appear in this status.
Blue nodes - denote the regular states. It will move among these during most of the tracker item life cycle.
Orange nodes - as there are only leaving transitions for these, one cannot move items to them, but can move items from these to the regular items. These nodes are primarily used for proper migration from legacy trackers.

### Conclusion

CodeBeamer is just that what it calls itself to be: Collaboration Development Platform. It provides a variety of tools to support both management and development, and in addition, provides also its integration plug-ins into most common IDEs (this will be mentioned in next chapter). It is also easily configurable and maintainable. The UI is seen as a small minus, where the most common tasks are not topped and remain “hidden” in their categories – then can be found when one searches, but they are not evident.

# 4. Proposed system

Bug tracking system is essentially and effectively implemented to monitor the status of bugs in an application. All the bugs that are identified are stored in a database. Each bug is assigned with a unique bug id and respective status of bug. Bugs can be created and updated with ease. Specific user accounts to control the access and maintain security are incorporated into the application.

Bug tracking system which is implemented on Java provides an overview on standards of coding of the developers involved. Employee accountability can be tracked and analysed on daily basis by using report generation option.

This web-based business application is a great tool for assigning and tracking issues and tasks during software development and any other projects that involve teams of two or more people.

- Bug tracker is used to access and view the status of bugs in an application on the dashboard provided.
- Bugs will be identified and stored in a database.
- Bugs are given a unique token id for identification and a status about the current situation of bugs are assigned.
- Bugs are updated and created with minimalistic effort
- Change request can be executed and info about the tickets can be seen.
- Dashboard will show current status of the ticket and will have functionality to search for a ticket using ID, date or by application.
- Logging a bug event is easy.

## Bug Tracking System Modules

This application is divided into following modules:

Project Manager/Admin
Developer
Tester

### Module Description

Project Manager/Admin

1. Admin can add new user or can update the details of the existing user.
2. Creates the project and assigns them to the employees for completing.
3. Analysing the project progress assigned to particular employee.
4. Adding bug types, severity and status.
5. Maintaining Project details, developer details and tester details.

Developer

1. View list of assigned projects
2. Responsible for bug status updation
3. Reset and edit his information like passwords

Tester

1. View list of assigned projects
2. Add bug details
3. View Bug status

## 5. Implementation or architecture diagrams

Pseudocode

**Input:** xml files of bugs X = {x1, x2,….,xn}
Source code S
**Output:** all methods causing related bugs
1. Init: parsing X;
Extracting all class names and method names from S;
2. for each xi in X do
3. BF = bug_features_extractor(xi)
4. for each method mi in BF do
5. M(mi1, mi2,….,mit) = methods_extractor(mi,S);
6. Sims(mi1, mi2,….,mit) = similarity_calculation(mi);
7. if(sim($m_\infty$ ,m1) > 0)
8. return $m_\infty$ ;

Begin

Input: Username, Password, User type /*Enter valid username and password*/
Connect to Server /*Validations*/
Check for the authorization of user
Enter name and password
Search in the database(login)
If match found
Connect user Profile Module

```
        Else
                Display “User authentication failed”
        EndIf
    End
```

## Architectural Design

Architectural Design is a process of decomposing large complex system into small subsystems. These subsystems are meant for providing some related services.


Fig 1: Architectural design for Bug Tracker

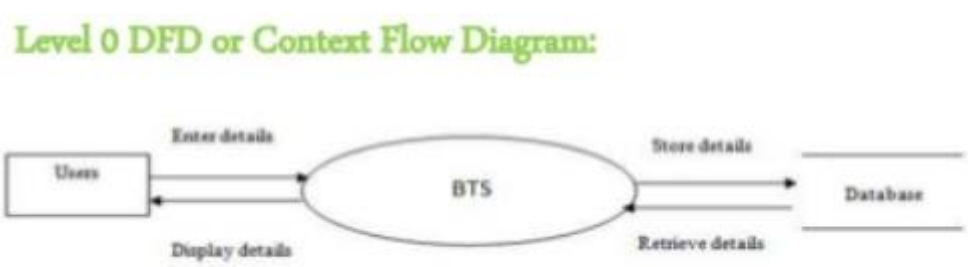


Fig 2: Level 0 Data Flow Diagram for Bug Tracker

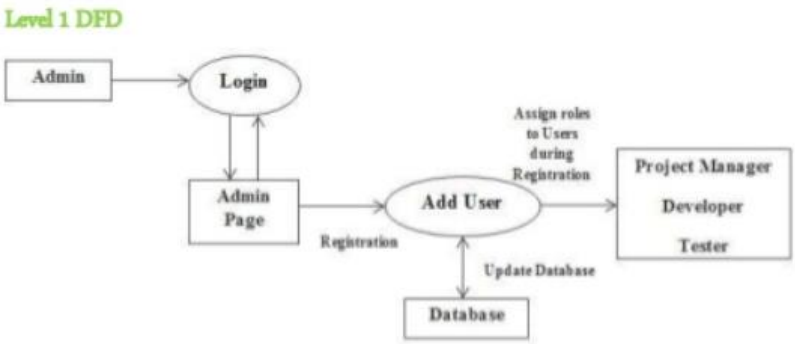


Fig 3: Level 1 Data Flow Diagram for Bug Tracker

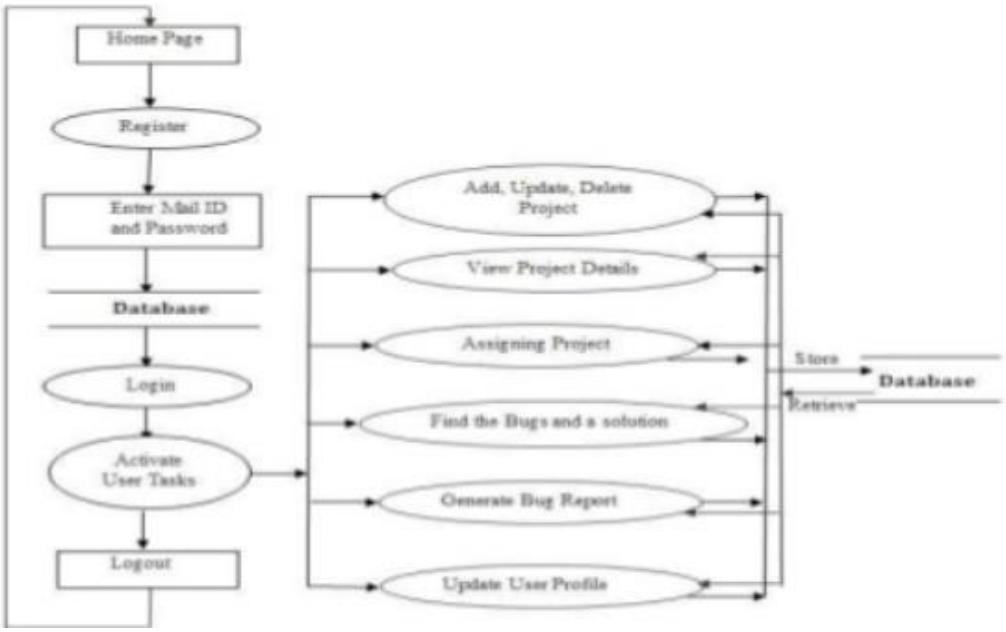


Fig 4: Level 2 Data Flow Diagram for Bug Tracker

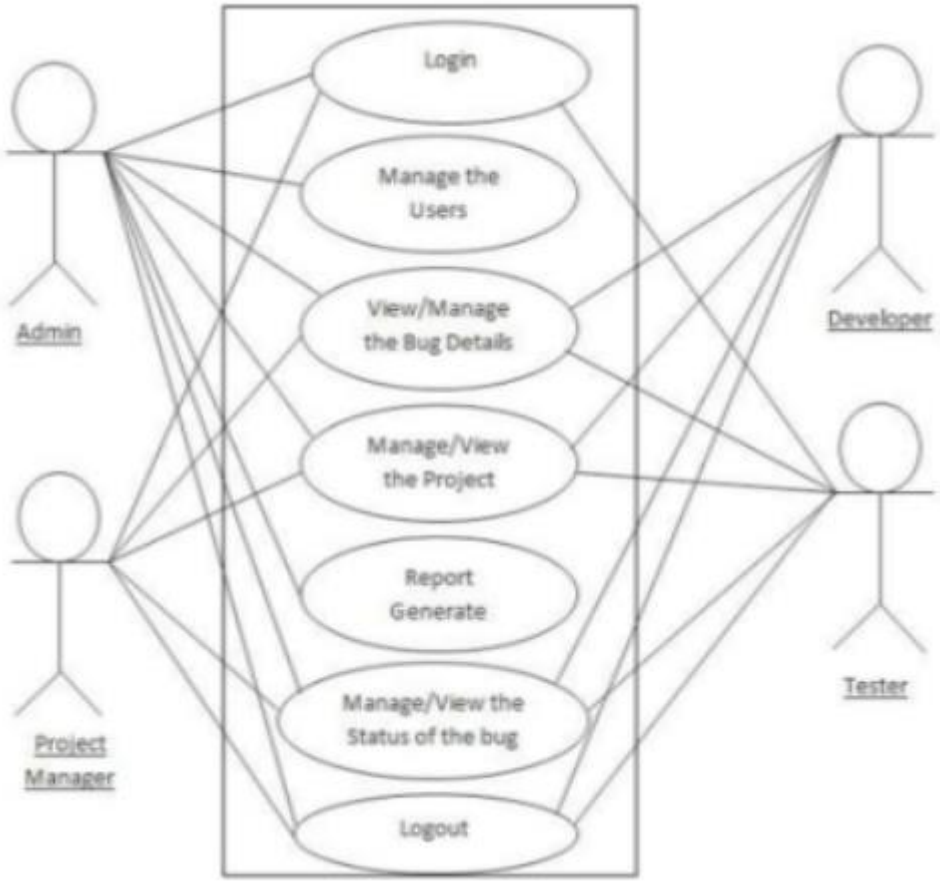


Fig 5: Use case diagram for Admin

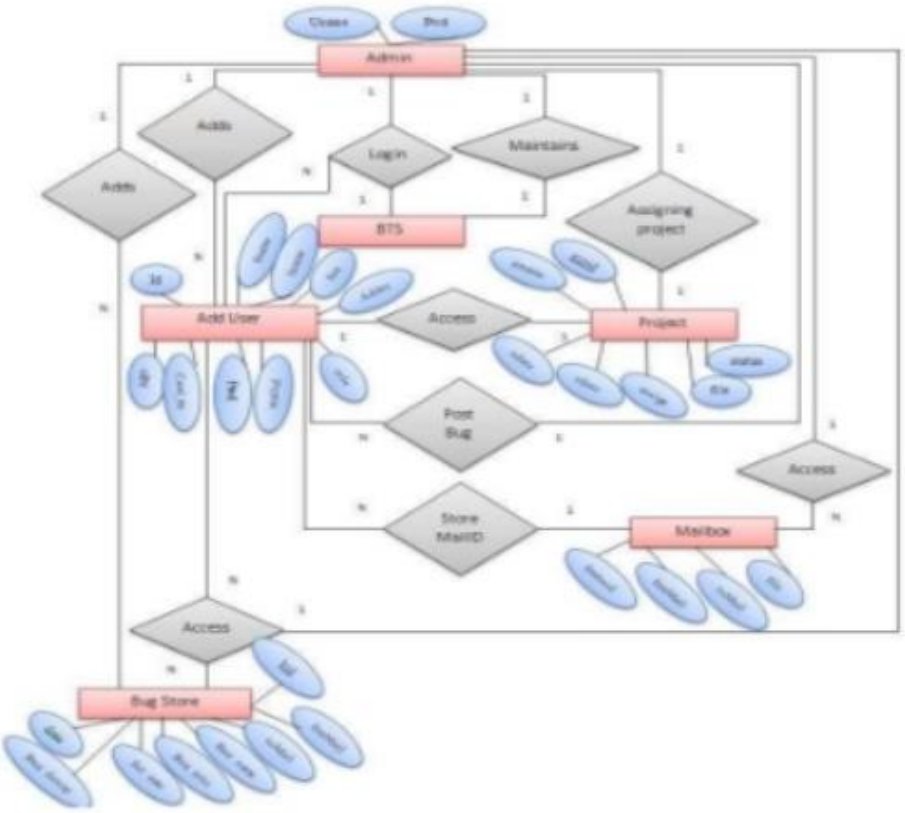

Fig 6: ER diagram for Bug Tracker

# 6. Output / Result / Screenshot

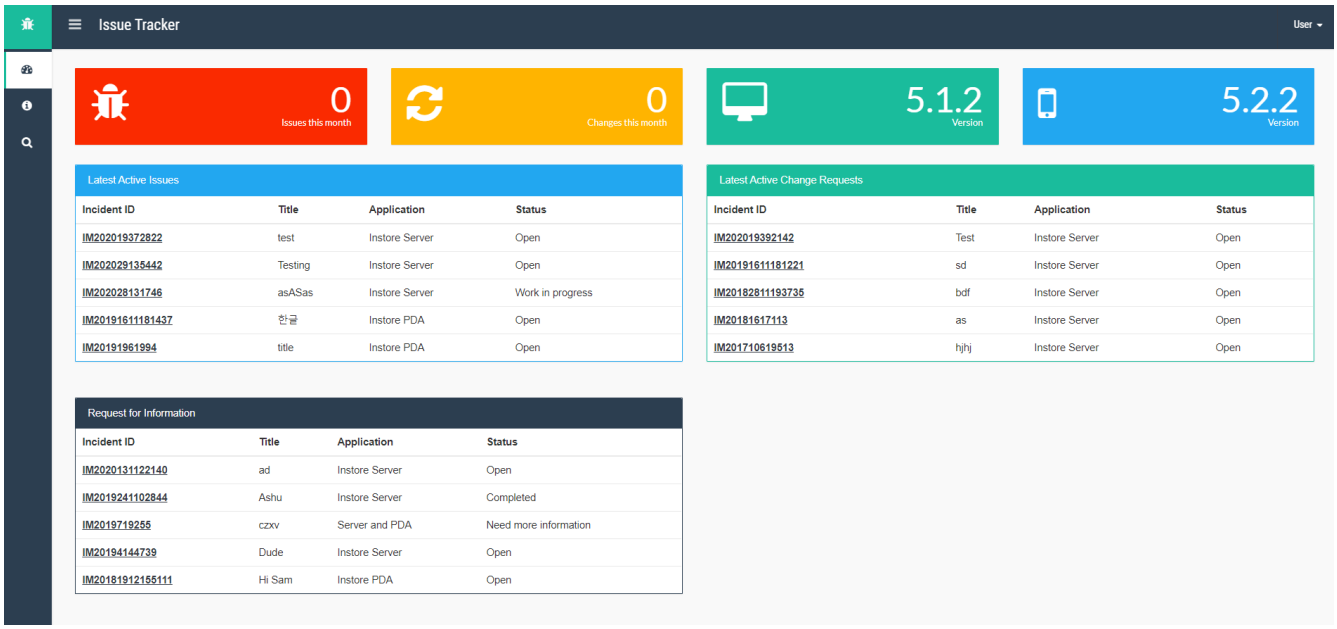


Fig 7: Home page of the bug Tracker

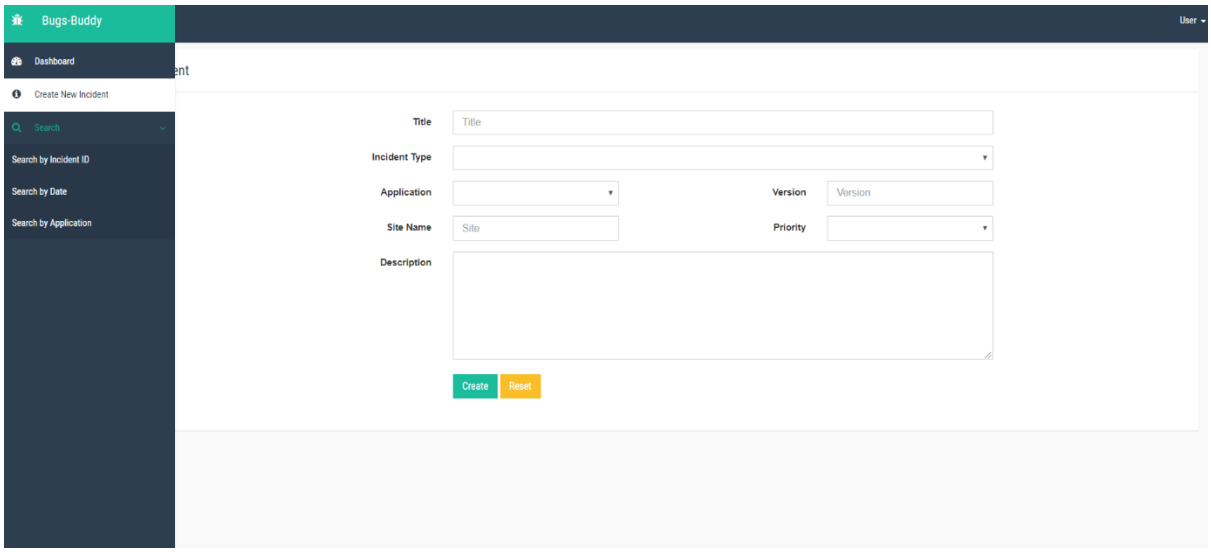


Fig 8: New Incident Creation Page

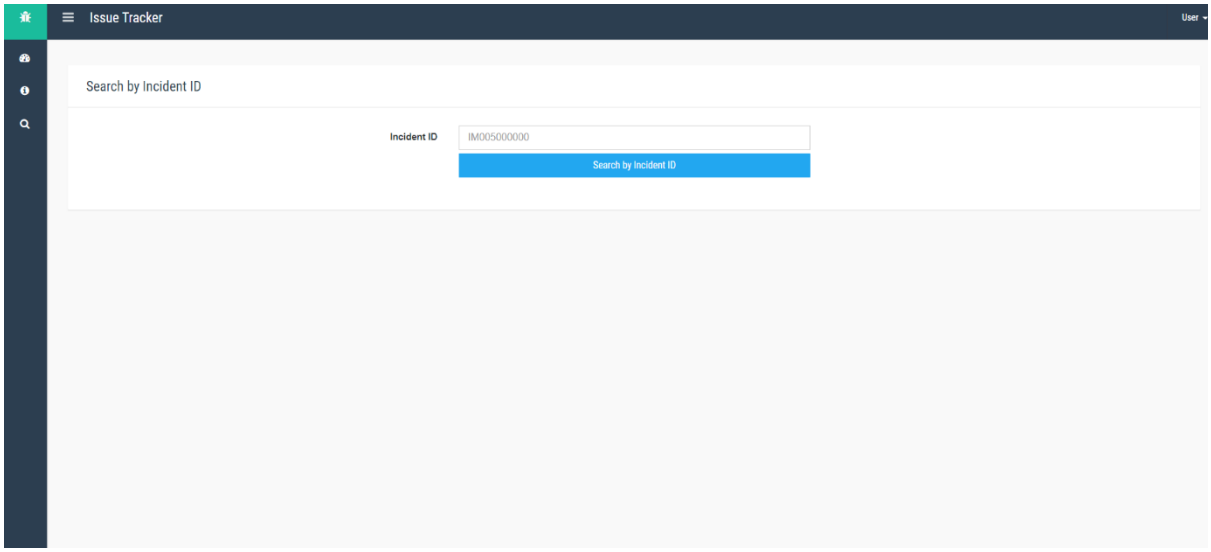


Fig 9: Search by ID Page

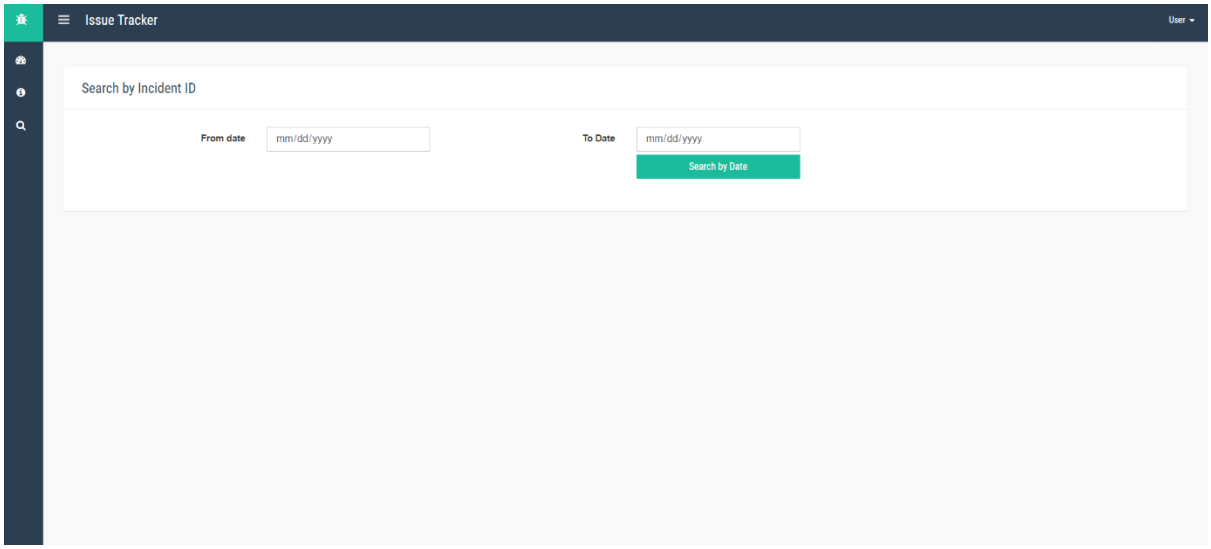


Fig 10: Search by Date Page

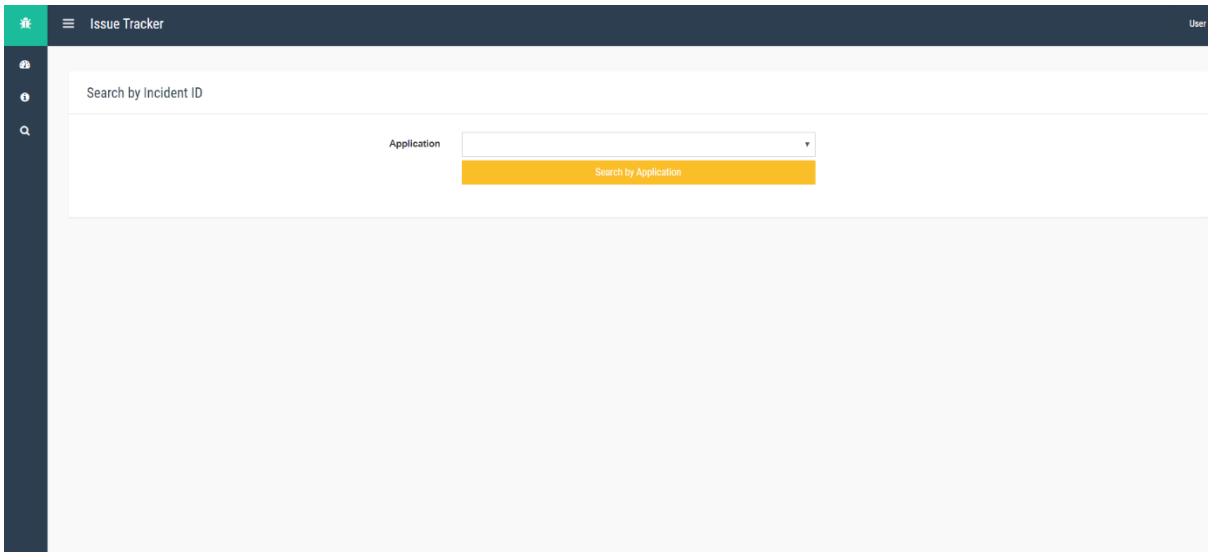


Fig 11: Search by Application No page

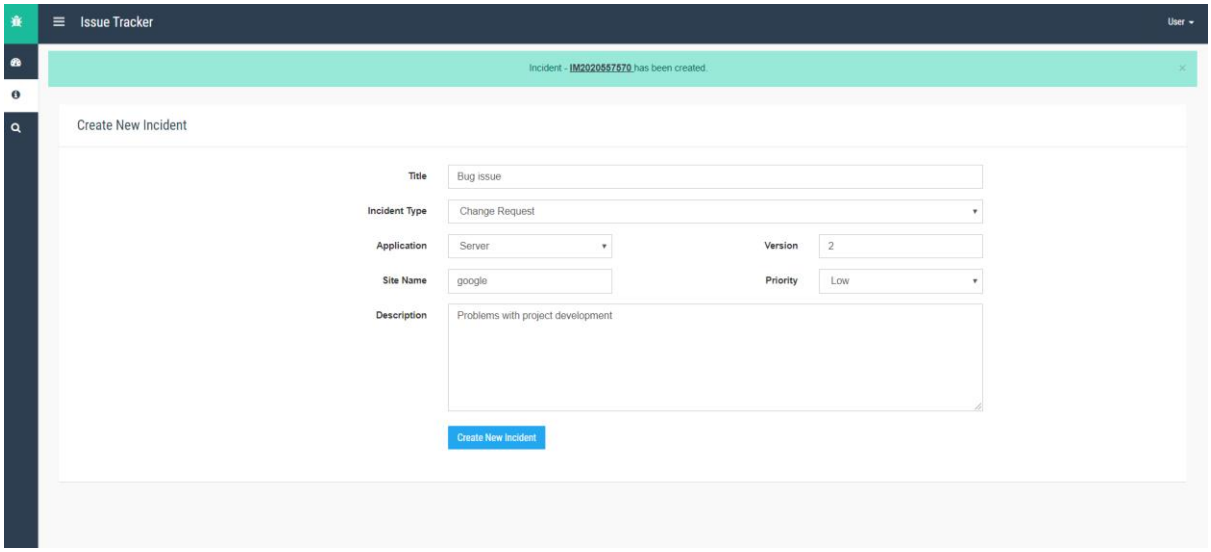


Fig 12: New incident created with Incident Id

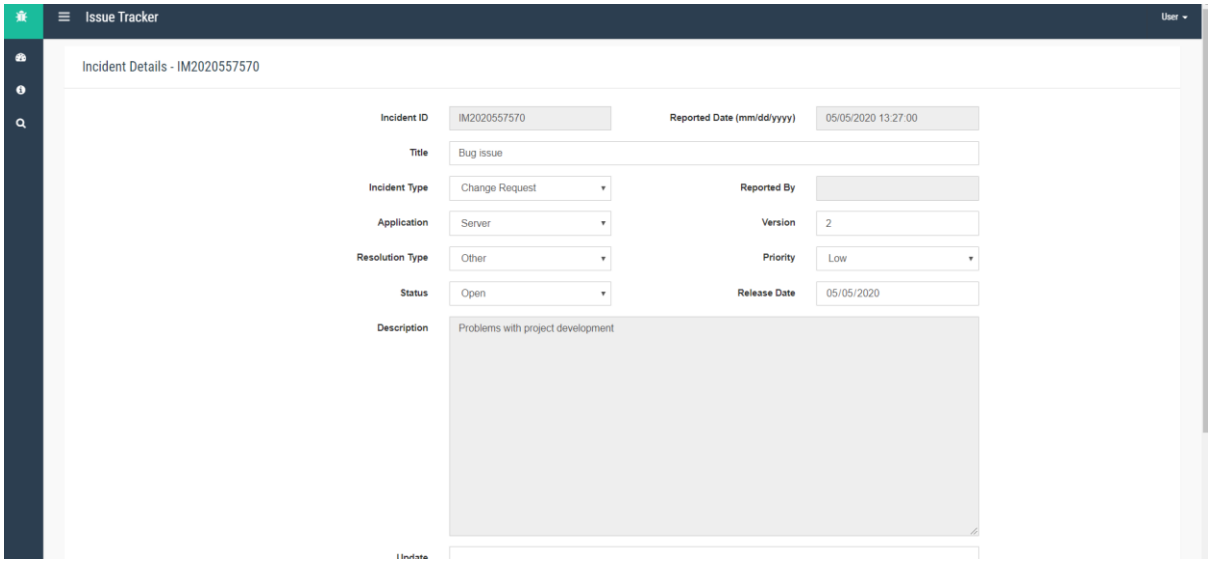


Fig 13: Details of an Incident

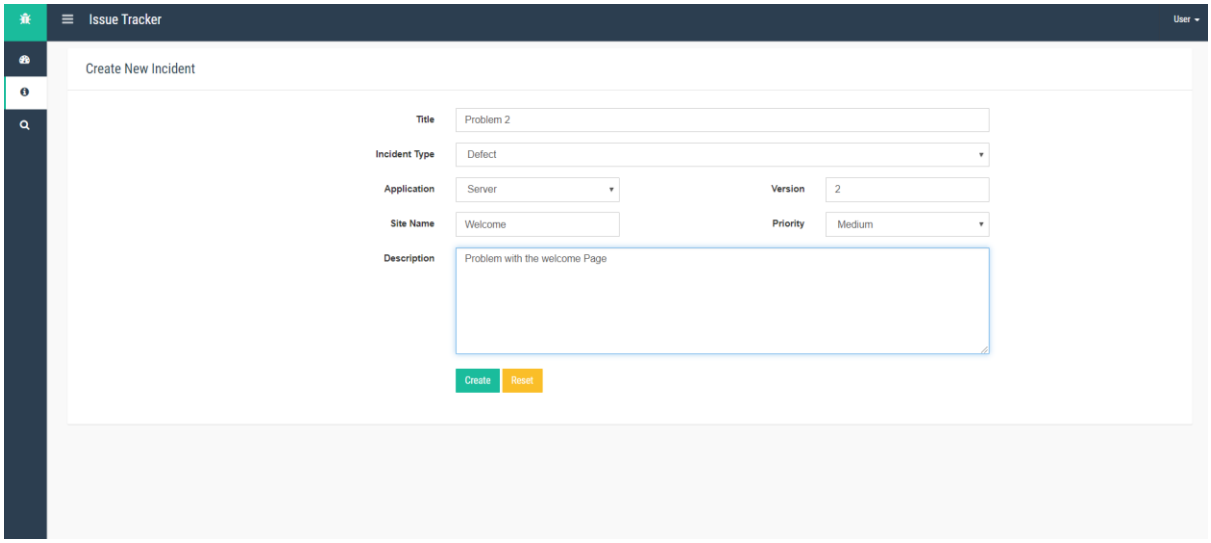


Fig 14: Details of a new incident filled as a demo

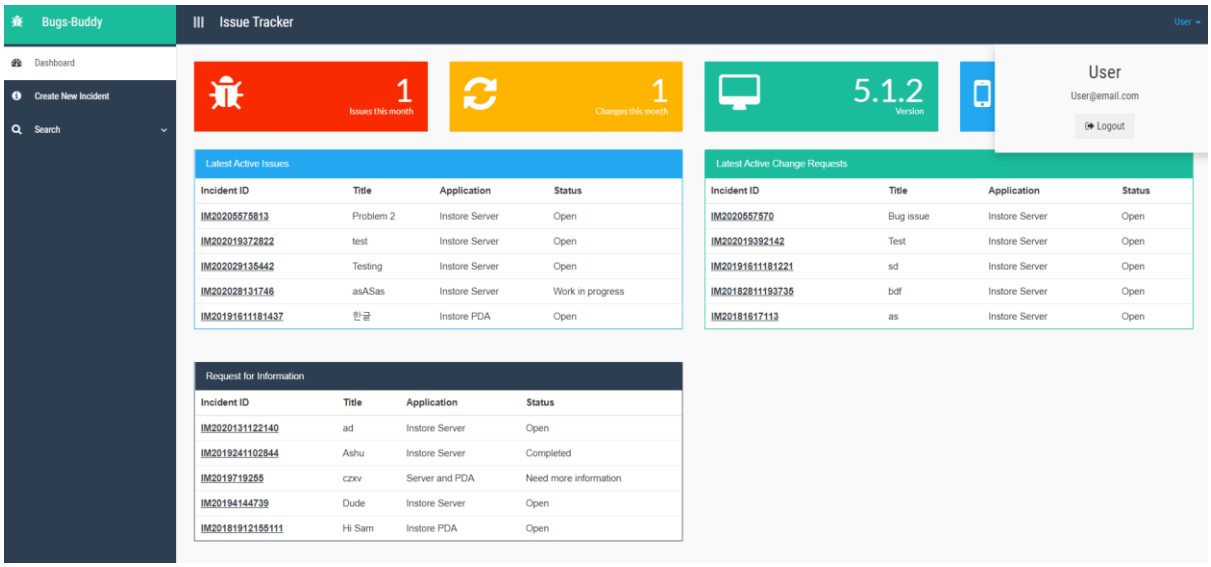


Fig 15: Logout screen on the Right side of the Window

# Conclusion/Future Enhancement

Bug Tracking System helps to detect and manage the bugs in software products effectively. This project Bug Tracker can be used to track the bugs in the project modules and assist in troubleshooting errors for testing and for development processes. This project highly avoids all sources of delay in bugs reporting level within the project modules in the software industry. As application is deployed in a company server, it is much more secure.

In Future we will adding more features to this application. We have tried our best to build a bug tracker which is very efficient in finding bugs and removes duplication bugs finding and gives good information about the bugs which helps a lot in troubleshooting the bug related problems.

# References


1. Jalbert, N., & Weimer, W. (2008, June). Automated duplicate detection for bug tracking systems. In *2008 IEEE International Conference on Dependable Systems and Networks with FTCS and DCC (DSN)* (pp. 52-61). IEEE.
2. Fischer, M., Pinzger, M., & Gall, H. (2003, September). Populating a release history database from version control and bug tracking systems. In *International Conference on Software Maintenance, 2003. ICSM 2003. Proceedings.* (pp. 23-32). IEEE.
3. Zimmermann, T., Premraj, R., Sillito, J., & Breu, S. (2009, May). Improving bug tracking systems. In *2009 31st International Conference on Software Engineering-Companion Volume* (pp. 247-250). IEEE.
4. Just, S., Premraj, R., & Zimmermann, T. (2008, September). Towards the next generation of bug tracking systems. In *2008 IEEE Symposium on Visual Languages and Human-Centric Computing* (pp. 82-85). IEEE.
5. Kelbaugh, M., Diederich, D., & Bush, E. (2010). *U.S. Patent No. 7,657,872*. Washington, DC: U.S. Patent and Trademark Office.
6. Serrano, N., & Ciordia, I. (2005). Bugzilla, itracker, and other bug trackers. *IEEE software, 22*(2), 11-13.
7. Lotufo, R., Passos, L., & Czarnecki, K. (2012, June). Towards improving bug tracking systems with game mechanisms. In *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)* (pp. 2-11). IEEE.
8. Murphy, G., & Cubranic, D. (2004, June). Automatic bug triage using text categorization. In *Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering*.
9. 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).
10. Del Fabro, M. D., Bézivin, J., & Valduriez, P. (2006, October). Model-driven tool interoperability: An application in bug tracking. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"* (pp. 863-881). Springer, Berlin, Heidelberg.