# Fall 2021 CS 497 Capstone Project
# Progress Report
# Web Application: Bug Tracking System

Kiran Limbu
Advisor: Hee Jung (Sion) Yoon, Ph.D.
BS in Applied Computing
School of Technology & Computing (STC)
City University of Seattle (CityU)
limbukiran@cityuniversity.edu, yoonhee@cityu.edu

## Abstract

Software testing is an important part of the software development process. Companies use bug tracking tools to help manage bugs discovered during the software testing process. Luckily there are many bug tracking tools out in the market. However, they are not of the same standard. If the tool is not looked at in-depth before considering for the project, it can be more troublesome than being of help. This paper explores the past work done by various sources to answer what makes a good bug tracking system.

Study shows that a good bug tracking system needs to produce a good report. To produce a good report, the bug tracking system needs to collect clear information. In order to collect clear information, a bug tracking system should contain important features that require a reporter to answer when creating a report. This information is helpful to those looking to design a bug tracking system that collects clear and important information. As a proof-of-concept, "BugsPot", a web-based bug tracking system was designed that requires reporters to add features that are thought to be important by developers.

**Keywords:** bug tracking system, bug, reporter, developer, communication, severity, report.

### 1. INTRODUCTION

The software can have hundreds or thousands of defects that need to be evaluated, monitored, and fixed. Defects occur when the code (application) executes in a manner other than expected. These defects are sometimes called issues, problems, bugs, errors, and mistakes. Bug tracking is the process of logging and monitoring these defects that are found during software testing. Over time, the number of defects gets multiplied. A bug tracking system is used to effectively manage them (IBM, 2021). The Quality Assurance (QA) team is a test team that finds the defects in the software and reports the defect in the bug tracking system. This informs developers of errors found in their work. Once bugs are reported, developers are responsible for fixing them. However, how soon the bugs are fixed depends on how good the report is.

In this paper, past research reports from various authors were reviewed to determine the properties of a good bug tracking system. In the process, a few bug-tracking tools that are available in the market were reviewed. One might think all the bug tracking tools are the same. But after reading reports from various authors, one realizes that every tool collects and displays information differently. Furthermore, the way certain information is considered important varies. Research by Bettenburg et al. (2008) on "What makes a good bug report?" shows that often what people think is important information to fix a bug is different from developers' needs.

For demonstration purposes, I built a prototype that is easy to follow and requires reporters to include all the information that is deemed important by the developers. Figure 0 shows a new bug report layout inside the BugsPot application.

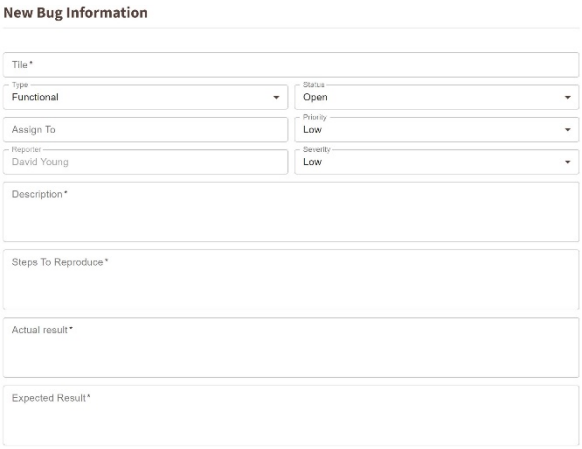

Figure 0: BugsPot add new bug page

**Problem Statement**

If the issues in the software are not properly handled, it can frequently crash the program, making the software look like a low-quality product and possibly increasing customers' frustration and dissatisfaction with the product. This can dampen the organization's reputation, not to mention the increase in cost in fixing the software. "An earlier study by IBM found that defects found in post-production or after release can cost 15 times more to fix compared to errors resolved early in development" (Ibm.com, 2021). In addition to that, authors Andy Glover and Matt Archer give us ten different reasons why bugs should be fixed as soon as you find them. To mention a few, the authors state that unfixed bugs camouflage other bugs, lead to duplicate effort, lead to inaccurate estimates, and so on (Glover et al., 2021).

Bug tracking system helps keep track of bugs and ensures that bugs found in the system get fixed. Doing so assists in creating high-quality products and helps achieve improved customer satisfaction. Additionally, when used properly, the Quality Assurance (QA) team can easily connect the cause of the defects to the source (for example, change in code). This easy traceability can help save production time and in turn lower the cost of development.

**Motivation**

Testing and bug tracking is an essential part of the application development process. An application can have many defects that need to be resolved. To do this there are many people involved. To name some, they can be testers (reporters), developers, project managers, product owners, and so on. It will help me understand my role and responsibilities and therefore help me communicate better with other team members. This research paper is the outcome of my desire to be a good application developer. Along with the research, as a prototype, I am also developing "BugsPot", a bug tracking application.

**Approach**

Applied Computing introduces different technologies in various fields like database, cybersecurity, web design, networking, and many more. Those knowledge and skills were useful in understanding the research materials in my project.

A bug tracking system is a valuable asset in the application development life cycle. BugsPot is going to be a web application that will require a database to store bugs reports. The application is going to use SQL to communicate with the database. With the knowledge of cybersecurity and networking, I plan on embedding secure processes into my work. Security will be baked into the design of the application.

**Conclusion**

As an aspiring application developer, it is important for me to know about bug tracking systems and their process. I hope to gain a deep understanding of the bug tracking system. In addition to that, the prototype should also serve as a portfolio piece when applying for a job. I will be able to showcase my knowledge and ability in developing web applications and creating test cases.

## 2. BACKGROUND

Bugs occur when the code (application) executes in a manner other than expected. Bugs can come in various forms. Even though they are sometimes used interchangeably, the root cause for each can be different. Bugs can be called wrong where the requirements are implemented incorrectly. Bugs could be the result of missing to implement a requirement of the customer. Sometimes a bug can also be implemented more than the customer's requirement. Bugs are also sometimes called errors which indicate developers made some mistake in their coding. Failure is another name for a bug. It means the inability of an application to perform its required functions within specified performance requirements. Bug is an error found in the code while in the development environment before it reaches the customer (360logica, 2021).

In the article "Report: Software failure caused $1.7 trillion in financial losses in 2017", the author Scott Matteson states that a report from a

software testing company Tricentis revealed that 3.6 billion people were affected by software failures and caused $1.7 trillion in financial losses. The author further adds, “Software bugs were the most common reason behind these failures, but proper testing would have eliminated these issues, as well as at least some of the security vulnerabilities and usability glitches” (Matteson, 2018).

That is why software testing is an important part of the software development process. In earlier days, testing was done in later phases of the software life cycle. However, nowadays it is very common to find testing introduced in earlier stages of the software life cycle. Many teams are using a methodology called continuous testing where testing and feedback are conducted at all stages of development.

Usually, Quality Assurance (QA) team conducts the testing operations. When a bug is found, a bug report is created. This bug report is crucial information for developers. They depend on these reports to find the issues in their coding and hopefully fix them on time before the next release. However, these reports are not of the same standard. They all vary depending on the reporter. Should there be a standard to these reports and if yes, what are the key features required in the report?

## 3. RELATED WORK

Literature review touches on various work done by other people in the past. It shows what are the key features required in the bug report. It talks about why some of the bugs never get resolved and it shows how clear communication between reporter and developer is essential in resolving bugs quicker. Finally, it also talks about how there are many bugs tracking systems out in the market but not all collect important information needed by developers.

### Literature Review

There has been much research done in the field of bug tracking systems. They all make a huge contribution to improving the overall process of bug tracking tools.

Zimmermann et al. (2009) proposed the idea that one can improve a bug tracking system by enabling an easy way to gather information. The authors developed a prototype based on information-centric to help other developers. The authors draw a conclusion that current bug tracking systems do not effectively collect all the information needed by developers. This results in prolonging bug’s life, and therefore, improvements to the way bug tracking systems collect information are needed (Zimmermann et al., 2009).

Hooimejer and Weimer (2007) proposed a model of quality bug reports based on a publicly available dataset from the Mozilla Firefox project. The model is designed to suggest important features to be added when building a bug report. According to their model severity field has a significant effect on the lifetime of the bug. Additionally, the authors’ finding shows the higher number of comments posted in response to the bug, and the number of attachments can make a huge difference. It can get the bug resolved in less time (Hooimejer & Weimer, 2007).

To find what makes a good bug report, Bettenburg et al. (2008) conducted a survey among developers and reporters. The survey result showed that developers’ needs were not always matched by what reporters were supplying.

When it came to important content in bug reports, developers ranked the following features to be important respectively: Steps to reproduce, stack traces, test cases, observed behavior, screenshots, and expected behavior. On the contrary, due to difficulty in reproducing or finding, only a few reporters would include the stack trace and code examples and test cases in their reports. The survey also revealed that some of the major issues with the bug reports were errors in steps to reproduce, incomplete information, wrong observed behavior, and error in test cases (Bettenburg et al., 2008).

Their key findings from the sample of 150,000 bug reports showed that Bug reports containing stack traces get fixed sooner, Bug reports that are easier to read have lower lifetimes and including code samples in your bug report increases the chances of it getting fixed sooner.

In 2020, Rahman et al. (2020) conducted a multimodal study to better understand the non-reproducibility of software bugs using 576 non-reproducible bug reports from Firefox and Eclipse. During their research, they found 11 key factors that might lead to non-reproducibility. Furthermore, they validated their finding with professional developers. The authors also revealed that with the non-reproducible bugs, the developers either close those bugs or solicit further information which can be time-consuming.

**Review Conclusions**

Developers and reporters need to be on the same page when creating a bug report. Developers are the ones that will be tackling the issue once the bug is reported. For them to be successful in their job they need enough information to know exactly what the issue is and where they can find it. Zimmermann et al. (2009) show us that not all bug tracking systems collect all the information needed by developers. This research begs the question of what that important information is, needed by the developers. According to Bettenburg et al. (2008), steps to reproduce, stack traces, observed behavior, screenshots, and expected behavior, respectively, are some of the important and useful information for the developers. Studies show that some of the major issues with the bug reports are errors in steps to reproduce and incomplete information (Bettenburg et al, 2008). Errors in steps to reproduce would lead to non-reproducible bugs.

With such a report, it is obvious why bug lifetime would be long and sometimes never resolved. Rahman et al.'s, (2020) research shows that there are 11 key factors that might lead to non-reproducibility. Those bugs are either closed or solicited for more information which again can lead to a long life for that bug (Rahman et al., 2020).

According to Hooimejer and Weimer (2007), the severity field can help shorten the lifetime of the bug. Additionally, the higher number of comments posted in response to the bug can also help resolve the issue faster.

## 4. APPROACH

In this project, both qualitative and quantitative methods were used. Quantitative was used to find answers to questions relating to 'What'. For example, what makes a good bug tracking system good? Qualitative was used to find answers to questions relating to 'Why'. For example, why are some issues/bugs resolved faster than others?

When building a prototype, the user's need was put as a top priority and that is what helped decide on the technology choices. For example, I wanted the application to be accessible from any device and therefore decided to work with web API rather than monolithic design. Bugs report needs to be stored in one place for easy and quick access.

**Requirement**

When designing BugsPot all the information gathered was considered and analyzed. Due to time constraints, the project focuses on minimal but important features (refer to Literature Review section) to help developers while the interface is simple and clean for easy readability.

Important features

1. Easy to navigate.
2. Clear communication between the developer and the user.
3. Login feature
4. Add Bug (title, severity, priority, status, area, version, producibility, bug description, steps to reproduce, results, file attachment, assign task and log with timestamp)
5. Search engine
6. Subscription functionality
7. Dashboard
8. Different access level

**Architecture Design**

Three different technology was used to develop this web-based bug tracking application. For the front end React was used. DOTNET Core for backend and SQL server to store data. Figure 1 shows how these technologies interact with each other.

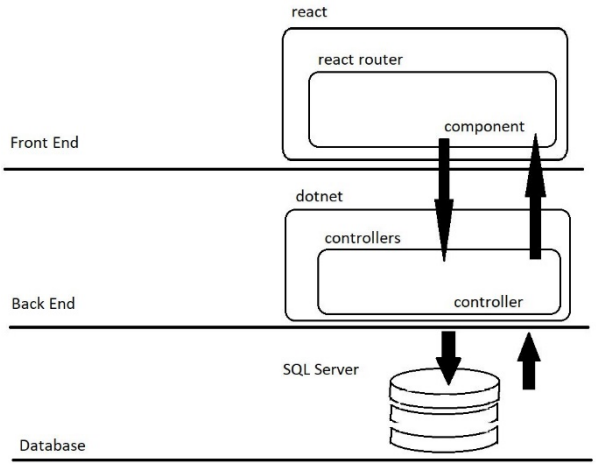


Figure 1: Interaction between different Technology

The application enforces role-based access control by requiring everyone to log in with their proper credential. This helps prevent unwanted changes to the bug life cycle. The application uses Jason Web Token (JWT) to authenticate users. Figure 2 shows the JWT authentication process.

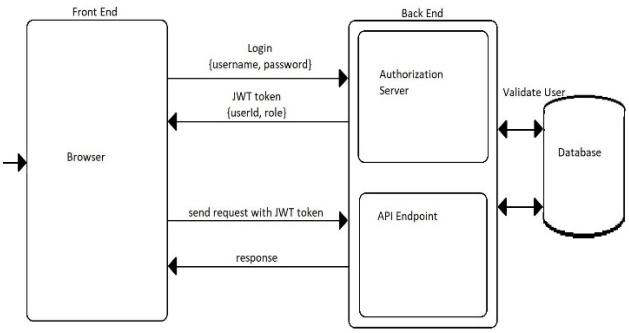


Figure 2: JWT authentication process

BugsPot is divided into 3 modules. They are as follows:

1. Project Management: In this module, the admin can add and update different projects. The reporter can add and update bugs in each project while developer can only update bugs.
2. User Management: In this module, the admin can add different users and update their information including the user's role. Users can update profile details and their passwords.
3. Dashboard: Dashboard is accessible to all the users. It is the first thing users see when they log in.

Users are divided into four groups: Admin, New User, Developer, and Reporter. New User is someone who has not been assigned a role. Below table (Figure 3) shows the access level for each user based on their role.

| Function | Admin | New User | Developer | Reporter |
|---|---|---|---|---|
| Project Management | * | | | |
| -Add new project | | | | |
| -Update project status | | | | |
| -Submit new bug | | | | * |
| -Update bug | | | * | * |
| User Management | * | | | |
| -Register new user | | * | | |
| -Update user details | | | * | * |
| -Change password | | | * | * |
| Dashboard | * | * | * | * |

Figure 3: Access level based on roles

When a new bug is added, a bug report is created in the BugsPot system with **Open** status. The bug gets assigned to a developer in which case the status changes to **Assigned**. The assigned developer starts working on the bug. The developer tries to reproduce the bug by following the information provided by the reporter. Changes are made to the code in an effort to fix the bug and once done the status gets updated to **fixed**. Tester once again visits the bug and follows the same procedure outlined in the bug report. If the output is the same as the result expected, the bug is fixed. Therefore, the status is changed to **close**. However, if the actual result does not match the expected result, then the status is changed to **Reopen** and soon **Assigned** to a developer for rework. Figure 4 shows a complete Bug Life Cycle inside BugsPot (softwaretestinghelp.com, 2021).

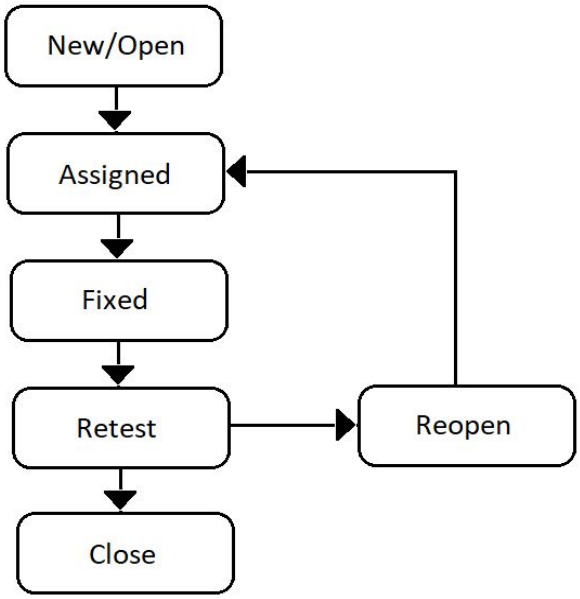


Figure 4: Bug Life Cycle

**Demo and Source code**
**Demo Link:** https://youtu.be/AqomuA-ZURA
**Source Code Link:**
https://github.com/kiranlimbu/C-sharp-Projects/tree/master/Bug%20Tracking%20Tool/BugsPot

## 5. DATA COLLECTION

Both, qualitative and quantitative approach was taken when collecting data for this project. Materials gathered for this project are mixed with peer-reviewed journals, and thoughts and feedback from professional QAs, gathered from various sources located at the end of this work.

Most of the peer-reviewed materials were used as a quantitative approach in finding scalable answers. Figure 5 shows a questionnaire presented to APACHE, ECLIPSE, and MOZILLA developers and reporters by Bettenburg et al. (2008). According to the authors, the survey was designed to collect facts on how developers use

the information in bug reports and what problems they face. Similarly, bug reporters were asked what information they provide, and which is most difficult to provide (Bettenburg et al, 2008).

| **Contents of bug reports** | | | |
|---|---|---|---|
| D1: Which of the following items have you previously been used when fixing bugs?<br>D2: Which three items helped you the most? | | | |
| R1: Which of the following items have you previously provided when reporting bugs?<br>R2: Which three items were the most difficult to provide?<br>R3: In your opinion, which three items are most relevant for developers when fixing bugs? | | | |
| Product | Hardware | Observed behavior | Screenshots |
| Compo-nent | Operation system | Expected behavior | Code examples |
| Version | Summary | Steps to reproduce | Error reports |
| Severity | Build information | Stack traces | Test cases |

| **Problems with bug reports** | | | |
|---|---|---|---|
| D3: Which of the following problems have you encountered when fixing bugs?<br>D4: Which three problems caused you the most delay in fixing bugs? | | | |
| You were given wrong: | There were errors in: | The reporter used: | Others: |
| Product name | Code examples | Bad grammar | Duplicates |
| Component name | Steps to reproduce | Unstructured text | Spam |
| Version number | Test cases | Prose text | Incomplete information |
| Hardware | Stack traces | Too long text | viruses |
| Operating system | | Non-technical language | |
| Observed behavior | | No spell check | |
| Expected behavior | | | |

Figure 5: Questionnaire presented to Developers (D) and Bug reporters (R).

In addition to the materials mentioned above, three different bug tracking tools were selected from the list of 20 Best Bug Tracking Tools (Hamilton, T., 2021), installed, and experimented with to understand the bug tracking system in the market.

1. Jira: It is a powerful software, developed by Atlassian, that allows bug tracking and agile project management.
2. Bugzilla: It is a popular web-based, free bug tracking system originally written by Terry Weissman.
3. Mantis: It is an open-source, web-based bug tracking system written in PHP that works with MySQL, MS SQL, and PostgreSQL databases.

## 6. DATA ANALYSIS

Figures 6 and 7 show the result from the above questionnaire that was presented to developers (D) and reporters (R) from Apache, Eclipse, and Mozilla, by Bettenburg et al. (2008). From Figure 6 we can see that most used items are steps to reproduce (83%), stack traces (57%), test cases (51%), observed behavior (33%), screenshots (26%), and expected behavior (22%) (Bettenburg et al., 2008).

As a QA/Tester/reporter when submitting a report, it is important to make the developer understand and experience the bug for an expected fix. For that developer needs to be able to replicate the bug and investigate the root cause of the error in the code (Ajonit Tutorials, 2021). Therefore, we see results fevering features like steps to reproduce, stack traces and test cases. However, that does not mean features with a lower rating are unrelated information. Every software is different, and bugs found in that software are also different. Features that may seem less important to a bug may be more important information for another bug.

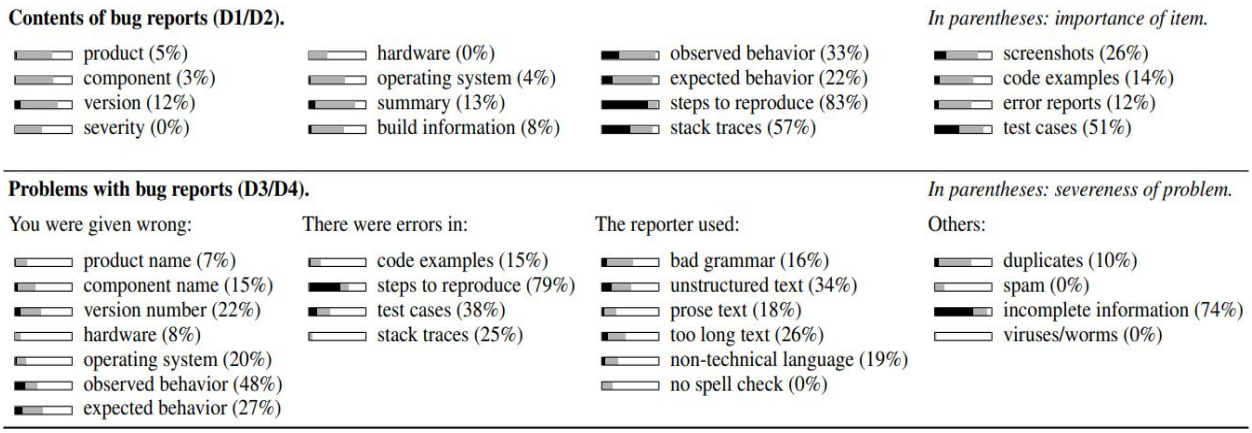


Figure 6: Results from the survey among developers

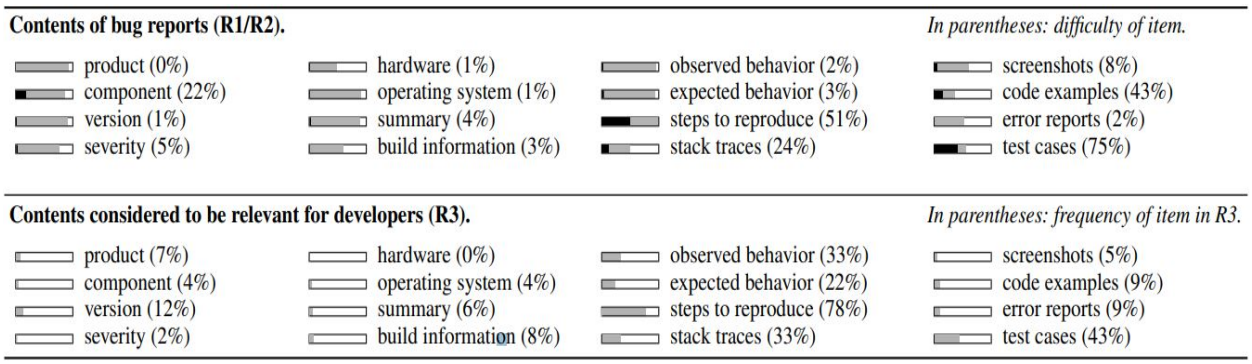


Figure 7: Results from the survey among reporters

Figure 7 represents the response from reporters where it shows most features provided by reporters are Steps to reproduce and observed and expected behavior. The number on the right, inside parentheses, represents difficulty in providing those items. For example, providing test cases is considered to be most difficult, and therefore even if the reporter thought it was important, they would be reluctant or unable to provide. In addition to that, the result also shows that features reporters consider being relevant for developers are different from what developers need (Bettenburg et al., 2008).

Along with the survey report, three different bug tracking tools in the market were also analyzed.

1. Jira
Pros:

1. Powerful Tool: The software is built for agile and Scrum project management that supports agile sprints and multiple project views.
2. Bug Tracking: The software was initially designed as a bug tracker. It is easy to report, locate and track bugs.
3. Reporting: Comes with great reporting tools like burnup charts, sprint reports, and many more.
4. Highly Customizable: Features and filters can be added to fit the organization's needs.

Cons:

1. Difficult to use: It is a powerful tool and can get very technical. Can be difficult to navigate for non-technical users.

2. Bug Tracking: Not all-important features are present. For example, Steps to produce is expected to be entered inside the description box.

2. Bugzilla

Pros:

1. Free Tool: It is open-source software.
2. Easy interface: It is easy to navigate around even for non-technical users.
3. Logging defects: Steps are clearly defined when logging defects making the process easy.
4. Application setup: Complicated and takes a long time to set up.

Cons:

1. Customizable: It needs more flexibility with customizing fields.
2. User Interface: It does not have that modern look and feels very outdated.
3. Accessibility: Cannot be associated with Agile methodologies, and plugins are limited. It needs more accessibility features.

3. Mantis

Pros:

1. Free software: It is a free, open-source, web-based bug tracking system.
2. User Interface: Like Bugzilla, the mantis interface is simple and easy to follow.
3. Customizable: It is a powerful tool that allows a high degree of customization.

Cons:

1. Interface: Very basic interface and does not always integrate with other tools. The looks need to be updated as well.
2. UX: You have to dig in to get some information causing pain and frustration.

## 7. FINDING

There are two parts to my research – documented survey by Bettenburg et al. (2008) and hands-on experience with three different bug tracking tools out in the market.

From the survey, it is clear that not all information is of equal importance when debugging a bug and therefore the kind of information provided matters. Information that helps reproduce the issue is important in knowing when the issue occurs. Stack trace exposes the path that leads to the exact point in the code where the error occurred. Observed and expected behavior helps the developer understand the requirement and make changes accordingly.

Additionally, the survey also shows more often than not, the information provided is incomplete or wrong. When that happens, according to Rahman et al. (2020), the developers either close those bugs or solicit further information. In such a situation, Hooimejer and Weimer (2007) stated that the higher number of comments posted in response to the bug can help resolve the issue quicker.

When experimenting with bug tracking tools, it was surprising to find that not all tools are of the same quality. To start with when setting up these applications, Bugzilla was the most inconvenient one. It had many individual components that needed to be installed separately. Jira was probably the easiest to install. Jira also has a modern and clean user interface making it easy to read the content. Bugzilla and Mantis on the other hand look very outdated.

Zira is a powerful tool with many features and therefore can get confusing for non-technical users. Additionally, in an effort to simplify the user interface, they only provide general fields instead of providing designated required fields for important information. The drawback to that is it can sometimes result in reporters forgetting to add all required information.

## 8. CONCLUSION

This research was about finding the answer to the question “what makes a good bug tracking system?” From my personal experience, I found that there are various bug tracking tools in the market, but they need improvement in one way or the other. Some are very technical, and some are very outdated and sometimes hard to navigate through.

Research suggests that for a bug tracking tool to be effective, it needs to be able to take all the important information needed for a developer to debug the code. It should require reporters to fill in important fields (features) so that reporters do not forget to include them in their reports. It also needs to be simple enough to navigate through.

## 9. FUTURE WORK

With limited time on hand for the project and limited knowledge of the frontend technology, only basic features were applied to the application. Moving forward, the goal is to add more features that can help users in better communication, track bug history, easily locate bug report and finally design it to be more user friendly. For example, adding comment capabilities would aid in better communication between reporters and developers. The bug log can show when the modification was made to the bug report and what kind of changes were made. Filters and search functionality are great features to have in the application. It allows users to look for specific bug reports based on the information provided like filter by developers, date, bug id, or unassigned bugs. It also helps save time and increase productivity. Another important feature is being able to generate reports from existing data. These data are valuable information for calculating estimation and progress, incident distribution, and ideas for process improvement (Mette, H., 2014). Finally, the application needs to be able to adapt and render pages according to different devices and screen sizes. This allows users to access the tool from any device with an optimal experience.

## 5. REFERENCE


Ajonit Tutorials (2021). What should be done after a bug is found? Retrieved 16 October 2021, from https://www.ajonit.com/software-testing/bug-is-found-tester-job/.

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).

Glover, A., & Archer, M. (2021). Ten Reasons Why You Fix Bugs As Soon As You Find Them. MoT. Retrieved 16 October 2021, from https://www.ministryoftesting.com/dojo/lessons/ten-reasons-why-you-fix-bugs-as-soon-as-you-find-them.

Hamilton, T. (2021). 20 Best Bug Tracking Tools (Defect Tracking Tools) in 2021. Guru99. Retrieved 16 October 2021, from https://www.guru99.com/top-20-bug-tracking-tools.html.

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).

Ibm.com (2021). What is bug tracking? | IBM. Retrieved 4 October 2021, from https://www.ibm.com/topics/bug-tracking

Matteson, S. (2018). Report: Software failure caused $1.7 trillion in financial losses in 2017. TechRepublic. Retrieved 30 October 2021, from https://www.techrepublic.com/article/report-software-failure-caused-1-7-trillion-in-financial-losses-in-2017/.

Mette, H. (2014). Guide to Advanced Software Testing, Second Edition (2nd ed.). Artech House.

Rahman, M. M., Khomh, F., & Castelluccio, M. (2020, September). Why are Some Bugs Non-Reproducible?:–An Empirical Investigation using Data Fusion–. In 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 605-616). IEEE.

softwaretestinghelp.com (2021). What Is Defect/Bug Life Cycle In Software Testing? Defect Life Cycle Tutorial. Retrieved 16 October 2021, from https://www.softwaretestinghelp.com/bug-life-cycle/.

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.

360logica (2021). Difference between Defect, Error, Bug, Failure and Fault! Retrieved 16 October 2021, from

https://www.360logica.com/blog/difference-between-defect-error-bug-failure-and-fault/.