BUG REPORTING

A software bug report contains specific information about what is wrong and what needs to be fixed in software or on a website. These comprehensive reports include requests or specifics for each software issue. It lets the developers understand what is wrong and how to fix it.

Now that we have established that bugs aren't good, and what bug reports are, let us discuss why we need them.

A clear and concise software bug report can help developers quickly understand what problems a web application or software is experiencing. The clear communication of issues with these reports will lead to fixing the bugs as soon as possible to provide users with a seamless experience.

Bug Reports: Why are they important?

How would you feel if you had to work with software that has many bugs and does that fulfill its intended purpose? Would you use such software or application? Surely, you wouldn't want to do anything with it.

Even if you are using such a tool, you wouldn't eventually give up and shift to a better software solution that offers a better experience. The same goes for end-users. If the software doesn't live up to their expectations, they will move to another platform.

Important information that every software bug report must have?

A good software bug report is all about how precise you are with the information.

Bug reporting best practices

Here are a few pointers to consider while writing a software bug report for developers:

Individual bug reports

Reporting multiple bugs in a single software bug report can lead to misunderstanding and confusion. If you have several issues to report, prepare individual bug reports with detailed information about each one.

It makes resolving bugs a breeze. The programmers quickly find and understand the problem. The more they know about the problem, the sooner they can start working on a solution.

Where is the bug?

Providing the location of the bug is important. It can guide the programmers in locating the bug and spare them from the hassle of hunting it down by themselves. For example: If the web application you are testing has a bug, include the URL of the page with the bug.

What were you doing when the bug appeared?

Providing details of the things that led to the bug can help developers recreate it. So, add information about what you were doing, which feature of the process you used, and the steps you took.

What did you expect to occur?

Adding details about the task you were trying to fulfill and the output you expected can further help programmers investigate the bug and locate the source of the issue.

Gather visual proof

While written details do help, adding visual information in the software bug reports can add more to it. If possible, try to record the bug with a screen recording tool. Also, take some screenshots of the same to give more details about the bug.

Add technical information

When describing your experience with the bug, include information such as your internet browser and operating system. Also, provide information about the desktop or mobile device on which you encountered the bug.

· Web browser

· Operating System of the device

· Display size

· Pixel ratio

Provide console logs

Console logs can help software developers identify and analyze all the errors that occur on any webpage. They can also check the user's actions that led to the bug. Using the logs, they can identify the root cause of the problem and speed up fixing the bug.

Furthermore, replicating a reported bug can be a difficult task, even after numerous attempts. The console logs can provide the added information to recreate the bug. Some tools like Disbug automatically captures all the technical logs.

Record error messages and codes

Another critical piece of information you can provide about the bug is the error message or code. Software developers can use the error message or code to locate and understand the bug.

With this information, developers will know what steps to take to resolve the issue in no time. However, keep in mind that error messages and codes on their own can be minor issues.

Replicating the bug

Does the bug occur every time? Perform the same task you were doing multiple times and check if the bug occurs every time. Include your findings in the software bug report as it will help developers identify the source of the bug.

Did you try to fix it?

With a website's functionality, there are a lot of variables that can affect its performance. Even the web browser you use can lead to errors. The point being, there are too many things to consider. But you can eliminate some of those by doing a few simple tests.

· Refresh the web page and try executing the task again

· Restart your system and try again

Also, don't forget to mention these tests in your bug reports.

How does it hinder your tasks?

The developers determine the order of fixing bugs by analyzing how much they obstruct the completion of a task. Developers prioritize fixing bugs by deciding how important they are. They evaluate the urgency and severity of the bugs and decide the order in which they will fix them.

The severity of the condition can range from severe (preventing you from working at all) to minor (which allows you to function normally). This data assists software engineers in determining the most efficient path to bug resolution.

These are some of the bug reporting best practices that can help you clearly communicate the issues. Follow them and you will have the perfect bug report that your developers will appreciate.

Software bug status in the defect life cycle

The Bug Life Cycle in software testing refers to the exact sequence of states that a bug has gone through since it was discovered. It aims to improve communication and coordination.

Fixing bugs involves many assignees. A lack of related information can lead to misunderstandings and delays in fixing the bug. The bug life cycle seeks to improve transparency and efficiency.

New: When a bug is first identified and reported, the status assigned to is new. It is the first stage in the bug life cycle. Validating the big and fixing it comes in the later stages.

Assigned: Once you report the bug, it is then assigned to the development team for fixing. The development team manager, tester, or project lead usually assigns it to the developer.

Open: In this stage, the developer validates the bug and starts fixing it. Although, if the developer deems the reported bug invalid, he/she transfers it to any of the following stages.

- Duplicate
- Deferred
- Rejected
- Not a bug

Fixed: As the name suggests, the stage of the bug life cycle shows that the assigned developer has finished fixing the bug.

Pending retest: When the assigned developer finishes fixing the bug, he/she marks it as fixed and assigns it to a tester to validate the fix. This stage of the bug cycle is called pending retest.

Retest: In this stage, the tester starts verifying the fix. If the bug is fixed and the software works as intended. Although, if it doesn't, the tester then changes the status to 'Reopened'.

Verified: This stage of the bug cycle clarifies that the tester has found no issues. The bug is they moved from retest to verified.

Reopened: If the tester finds any issues after the bug fix, he/she will change the status to reopened. The bug will be assigned to a developer again for a fix.

Closed: After the tester verifies the bug fix, he/she will mark it as closed.

Duplicate: A bug becomes a duplicate if it occurs twice or if it is identical to a previously reported bug concept.

Rejected: If the developer has doubts about the validity of the bug, he/she will mark it as rejected.

Deferred: The status "Deferred" is assigned to bugs that are not top priorities and are likely to be resolved in the upcoming release.

Not a bug: If a bug does not affect the software's functionality, it is classified as "Not a bug."

What are bug reporting tools and tracking systems?

A bug reporting tool, also known as a defect or bug tracking tool, is a software solution that records, reports, and manages the data on bugs that happen in any software service. Bug reporting and tracking is an essential part of software development. It helps development teams in preparing precise software bug reports and track the bug fix process.

Using bug reporting tools, developers can automate monitoring and tracking bugs. This helps them ensure the efficient functioning of any software, website, or service.

The five important functions of bug reporting in software testing:

- **Detecting the bug:** The development teams detect the bugs from the software in the testing phase.
- **Reporting:** The developer logs in the bug with all specific details related to the bug, such as time, URL, process, visual proof, console logs, and more.
- **Fixing:** Assign the bug to the developer and track the process of the fix.
- **Testing:** Validate the fix by testing the software and analyze its functionality.

- **Accumulating data:** Maintain records of the bug in the software bug report to avoid the same bug in future development.

Disbug: The best bug tracking tool

Disbug is a web-based bug reporting tool and project management platform. We specifically designed it for developers and project managers. Disbug can help you create comprehensive software bug reports with ease. You can also monitor the bug fix status and bring transparency to the process.

· Visually narrate software bugs with screen-cast and screenshot annotations

· Capture console and network logs, local storage, user clicks, and much more with ease.

· Get technical logs straight to your preferred tools with a link

· Integrations for Jira, GitLab, Trello, Slack, and more to streamline the process.

Still hesitant about the functioning of the tool? Disbug offers a free plan that you can use to test all the features of the tool.

METRICS AND STATISTICS.

Software Testing metrics are quantitative steps taken to evaluate the software testing process's quality, performance, and progress. This helps us to accumulate reliable data about the software testing process and enhance its efficiency. This will allow developers to make proactive and precise decisions for upcoming testing procedures.

What is a metric in software testing metrics?

A Metric is a degree to which a system or its components retains a given attribute. Testers don't define a metric just for the sake of documentation. It serves greater purposes in software testing. For example, developers can apply a metric to assume the time it

takes to develop software. It can also be assigned to determine the numbers of new features and modifications, etc., added to the software.

Importance of Software Testing Metrics

As mentioned, test metrics are crucial to measuring the quality and performance of the software. With proper software testing metrics, developers can−

- Determine what types of improvements are required to deliver a defect-free, quality software
- Make sound decisions about the subsequent testing phases, such as scheduling upcoming projects as well as estimating the overall cost of those projects
- Evaluate the current technology or process and check whether it needs further

  modifications

Types of software testing metrics

There are three types of software testing metrics−

- **Process Metrics:** Process metrics define the characteristics and execution of a project. These characteristics are essential to the improvement and
- maintenanceof the process in the SDLC (Software Development Life Cycle). **Product Metrics:** Product metrics define the size, design, performance, quality,and complexity of a product. By using these characteristics, developers can enhance their software development quality.
- **Project Metrics:** Project Metrics determine the overall quality of a project. It is used to calculate costs,

productivity, defects and estimate the resource and deliverables of a project.

It is incredibly vital to identify the correct testing metrics for the process. Few factors to consider−

- Choose your target audiences wisely before preparing the metrics
- Define the goal behind designing the metrics
- Prepare metrics by considering the specific requirements of the project Evaluate the financial gain behind each metrics
- Pair the metrics with the project lifestyle phase that delivers

optimum output .Software testing can be further divided into manual and automated testing.

In manual testing, the test is performed by QA analysts in a step-by-step process. Meanwhile, in automated testing, tests are executed with the help of test automation frameworks, tools, and software.

Both manual and automated testing has its strength and weakness.

Manual testing is a slow process, but it allows testers to handles complex scenarios.

The most significant advantage of automated testing is that it enables testers to run more testing in less time, covering a substantial level of permutations, which is nearly impossible to calculate manually.

TYPES OF MANUAL TEST METRICS

Manual Test Metrics are of two types−

Base Metrics

Base metrics are data collected by analysts during test case

development and execution. These metrics are submitted to test leads and project managers by preparing a project status report. It is quantified by using calculated metrics −

- Number of test cases
- Number of test cases executed

Calculated **Metrics**

Calculated metrics are derived using data from base metrics. The test lead gathers these data and converts them to more meaningful information for tracking the progress of projects at the module level, tester level, etc.

It comprises a significant part of SDLC and empowers developers to make vital improvements in software.

Most used Metrics

Below are the types of metrics, popularly used by developers and testers

- **Defect metrics:** This metric allows developers to understand the various quality aspects of software, including functionality, performance, installation stability, usability, compatibility, etc.
- **Defects finding rate:** It is used to identify the pattern of defects during a specific timeframe
- **Defect severity:** It enables the developer to understand how the defect is going to impact the quality of the software.
- **Defect cause:** It is used to understand the root cause of the defect.
- **Test Coverage:** It defines how many test cases are assigned to the program. This metric ensures the testing is conducted to its full completion. It further aids in checking

the code flow and test functionalities.

- **Defect fixing time:** It determines the amount of time it takes to resolve a defect **Test case efficiency:** It tells the efficiency rate of test cases in finding defects **Schedule adherence:** Its primary motive is to figure out the time difference between the planned schedule and the actual time of executing a schedule.

TEST METRICS LIFE CYCLE

The life cycle of test metrics consists of four stages−

- **Analysis:** In this stage, developers identify the required metrics and define them.
- **Communicate:** Once metrics are identified, developers have to explain their importance to stakeholders and the testing team.
- **Evaluation:** This stage includes quantifying and verifying the data. Then testers have to use the data to calculate the value of the metric.

Report: Once the evaluation process is finished, the development team needs to create a report including a detailed summary of the conclusion. Then the report is distributed among stakeholders and relevant representatives. The stakeholders then give their feedback after reading the information carefully.

Metrics and statistics play a vital role in measuring and evaluating various aspects of performance, progress, and outcomes in different domains. They provide objective and quantifiable data that can be used to assess the effectiveness, efficiency, and quality of processes, systems, or initiatives. Here are some common metrics and statistics used in different contexts:

Key Performance Indicators (KPIs): KPIs are specific metrics that organizations use to

measure progress toward their goals and objectives. KPIs vary depending on the industry and the specific objectives being measured. Examples include sales revenue, customer satisfaction scores, website traffic, employee turnover rate, and production efficiency.

Quality Metrics: These metrics assess the quality of products or services. They can include defect rates, customer complaints, return rates, and customer satisfaction ratings. Quality metrics help organizations identify areas for improvement and track their progress in delivering high-quality offerings.

Financial Metrics: Financial metrics measure the financial health and performance of a business or organization. Examples include revenue growth, profit margins, return on investment (ROI), cash flow, and debt-to-equity ratio. Financial metrics help evaluate profitability, liquidity, and overall financial sustainability.

Customer Metrics: These metrics focus on understanding and evaluating the customer experience and satisfaction. Examples include Net Promoter Score (NPS), customer retention rate, customer lifetime value, and customer complaints. Customer metrics provide insights into customer loyalty, preferences, and perceptions.

Productivity Metrics: Productivity metrics assess the efficiency and output of individuals, teams, or processes. Examples include units produced per hour, average handling time for customer inquiries, response time to support tickets, and employee utilization rates. Productivity metrics help identify bottlenecks, optimize resource allocation, and improve overall efficiency.

User Engagement Metrics: These metrics are used in digital and online contexts to assess user engagement and interaction. Examples include website traffic, page views, bounce rate, click-through rate, time spent on a page, and social media engagement (likes, shares, comments). User engagement metrics help evaluate the effectiveness of digital marketing

efforts, user experience, and content performance.

Risk and Compliance Metrics: Risk and compliance metrics assess an organization's adherence to regulatory requirements, internal policies, and risk management practices. Examples include compliance violation incidents, risk exposure levels, audit findings, and cybersecurity incidents. These metrics help organizations monitor their risk posture and ensure compliance with legal and regulatory obligations.

Process Efficiency Metrics: These metrics evaluate the efficiency and effectiveness of specific processes within an organization. Examples include cycle time, throughput, defect rate, and rework rate. Process efficiency metrics help identify areas of improvement, streamline operations, and reduce waste.

Market and Industry Statistics: These statistics provide insights into market trends, customer demographics, industry benchmarks, and competitive landscape. Examples include market size, market share, growth rates, customer segmentation data, and industry-specific performance indicators. Market and industry statistics assist in strategic decision-making, market analysis, and competitive positioning.

It's important to note that the selection of metrics and statistics should align with the specific objectives, context, and requirements of the organization or domain being measured. They should be relevant, measurable, and provide actionable insights to drive improvement and informed decision-making.