WHAT IS A BUG REPORT?

In the course of the QA process, when a bug has been identified, it has to be documented and sent to developers to be fixed. Given that software is exceptionally complex, layered, and feature-heavy in the current digital environment, most QA pipelines generate multiple bugs. Additionally, developers often work on multiple development projects simultaneously, which means they have a considerable number of bugs requiring attention. They have to operate under significant pressure and can be overwhelmed without the right resources.

Well-structured and adequately detailed bug reports are one of those resources. Good bug reports tell developers exactly what needs to be fixed and help them get it done faster. This prevents software releases from being delayed, offering faster time-to-market without compromising on quality.

Benefits of a good Bug Report:

A good bug report covers all the crucial information about the bug, which can be used in the debugging process:

1. It helps with a detailed bug analysis.
2. Gives better visibility about the bug and helps find the right direction and approach towards debugging.
3. Saves cost and time by helping debug at an earlier stage.
4. Prevents bugs from going into production and disrupting end-user experience.
5. Acts as a guide to help avoid the same bug in future releases.
6. Keeps all the stakeholders informed about the bug, helping them take corrective measures.

Elements of an Effective Bug Report:

When studying how to create a bug report, start with the question: What does a bug report need to tell the developer?

A bug report should be able to answer the following questions:

- What is the problem?
- How can the developer reproduce the problem (to see it for themselves)?

- Where in the software (which webpage or feature) has the problem appeared?
- What is the environment (browser, device, OS) in which the problem has occurred?

Elements of an Effective Bug Report:

When studying how to create a bug report, start with the question: What does a bug report need to tell the developer?

A bug report should be able to answer the following questions:

- What is the problem?
- How can the developer reproduce the problem (to see it for themselves)?
- Where in the software (which webpage or feature) has the problem appeared?
- What is the environment (browser, device, OS) in which the problem has occurred?

How to write an Effective Bug Report:

An effective bug report should contain the following:

1. Title/Bug ID
2. Environment
3. Steps to reproduce a Bug
4. Expected Result
5. Actual Result
6. Visual Proof (screenshots, videos, text) of Bug
7. Severity/Priority

1. Title/Bug ID

The title should provide a quick description of the bug. For example, "*Distorted Text in FAQ section on <name> homepage*".

Assigning an ID to the bug also helps to make identification easier.

2. Environment

A bug can appear in a particular environment and not others. For example, a bug appears when running the website on Firefox, or an app malfunctions only when running on an iPhone X. These bugs can only be identified with cross browser testing or cross device tests.
When reporting the bug, QAs must specify if the bug is observed in one or more specific environments. Use the template below for specificity:

- **Device Type**: Hardware and specific device model
- **OS**: OS name and version
- **Tester**: Name of the tester who identified the bug
- **Software version**: The version of the software which is being tested, and in which the bug has appeared.
- **Connection Strength**: If the bug is dependent on the internet connection (4G, 3G, WiFi, Ethernet) mention its strength at
  the time of testing.
- **Rate of Reproduction**: The number of times the bug has been reproduced, with the exact steps involved in each reproduction.

3. Steps to Reproduce a Bug

Number the steps clearly from first to last so that the developers can quickly and exactly follow them to see the bug for themselves. Here is an example of how one can reproduce a bug in steps:

1. Click on the "Add to Cart" button on the Homepage (this takes the user to the Cart).
2. Check if the same product is added to the cart.

4. Expected Result

This component of Bug Report describes how the software is supposed to function in the given scenario. The developer gets to know what the requirement is from the expected results. This helps them gauge the extent to which the bug is disrupting the user experience.

Describe the ideal end-user scenario, and try to offer as much detail as possible. For the above example, the expected result should be:

"The selected product should be visible in the cart."

5. Actual Result

Detail what the bug is actually doing and how it is a distortion of the expected result.

- Elaborate on the issue
- Is the software crashing?
- Is it simply pausing in action?
- Does an error appear?
- Or is it simply unresponsive?

6. Visual Proof of Bug

Screenshots, videos of log files must be attached to clearly depict the occurrence of the bug. Depending on the nature of the bug, the developer may need video, text, and images.

Testing using Browser Stack can leverage multiple debugging options such as text logs, visual logs (screenshots), video logs, console logs, and network logs. These make it easy for QAs and devs to detect exactly where the error has occurred, study the corresponding code and implement fixes.

Browser Stack's debugging toolkit makes it possible to easily verify, debug and fix different aspects of software quality, from UI functionality and usability to performance and network consumption.

The range of debugging tools offered by Browser Stack's mobile app and web testing products are as follows:

- Live: Pre-installed developer tools on all remote desktop browsers and Chrome developer tools on real mobile devices (exclusive on Browser Stack
- Automate: Screenshots, Video Recording, Video-Log Sync, Text Logs, Network Logs, Selenium Logs, Console Logs
- App Live: Real-time Device Logs from Logcat or Console
- App Automate: Screenshots, Video Recording, Video-Log Sync, Text Logs, Network Logs, Appium Logs, Device Logs, App Profiling

7. Bug Severity

Every bug must be assigned a level of severity and corresponding priority. This reveals the extent to which the bug affects the system, and in turn, how quickly it needs to be fixed.

Levels of Bug Severity:

- Low: Bug won't result in any noticeable breakdown of the system
- Minor: Results in some unexpected or undesired behavior, but not enough to disrupt system function
- Major: Bug capable of collapsing large parts of the system
- Critical: Bug capable of triggering complete system shutdown

Levels of Bug Priority:

- Low: Bug can be fixed at a later date. Other, more serious bugs take priority
- Medium: Bug can be fixed in the normal course of development and testing.
- High: Bug must be resolved at the earliest as it affects the system adversely and renders it unusable until it is resolved.