Good testing practices

When is something considered a bug?

For something to be a bug, it needs to "not work correctly". But this is a vague and imprecise definition. More accurately, a bug is something that doesn't behave as it's expected: something that doesn't follow the design and specifications determined by the people who decide how the application should work.
Of course, you can use your inner gut to decide if something is a bug or not without even knowing how the application is designed to work: if a button does not respond when clicked, it most certainly is a bug. But there are some other cases when you can't be sure and need to check the documentation or ask a project manager or designer.

What is a good bug report?

There are some main sections a bug report should have: detailed steps to reproduce (or acceptance test), actual result, expected result, screenshots, environment used to test, etc.. But, overall, a good report helps the developers determine what causes the bug and what needs to be fixed, along with what strategies they can use for that. To provide this information, a bug needs to be thoroughly investigated. This doesn't mean to spend a whole day trying to uncover every single little detail, but the description should be detailed enough and cover most cases that trigger the bug *(see "Now the bottom line: how to investigate a bug" below).*

Regression or not?

A "regression" is a bug recently introduced by the latest changes/features being developed. Basically, something stopped working at some point. "How long ago" it stopped working would help prioritize the fixing of the issue: if it worked 5 versions ago and then it broke, it was a regression 4 versions ago, but now there might be other fixes or features to develop that have a higher priority in the current development cycle (on the other hand, if a recent change just broke existing functionality, it would be wise to address is as soon as possible). Once an issue is found, in order to determine what kind of issue it is, testing should be performed in the application version/branch currently being developed as well as an older one to determine when it was introduced.

- If the issue is present in the current version and it was also present in a previous version, then it's not a regression (at least not in the current development cycle).
- If a previous version works as expected but the issue is present in a newer version, then it's a regression.
- If an issue is present in previous versions but it works as expected in newer versions, then chances are it was fixed just for the newer version. If this is the case, then there's probably a related bug report where more information can be found. It's also possible that the bug got "magically" fixed by some code change that wasn't intended to fix it but did (consider it a "bonus" :).

How to describe steps to reproduce / acceptance tests?

These steps should describe the exact scenarios that reproduce the bug, or the steps that should be followed to decide whether the bug has been fixed or not. This means that the cases should be

narrowed down until the exact behavior that triggers the bug is found. This will help developers determine what causes the issue and QE validate the fix.

How to describe the actual result?

It should be descriptive enough to help devs understand what exactly is happening without the need to look at screenshots. For example, saying that "a user can't create a post" in a blogging application is a vague and imprecise description that leads to more questions: Why can't they create content? Do they see an error message? Do they click on a button and nothing happens? Do they type and their text doesn't show up? A more appropriate description should be: "after clicking on the *Post* button nothing happens, but these errors are displayed in the browser console... [copy/paste error messages]".

Now, the bottom line: how to investigate a bug?

Let's consider a blogging application that allows users to create and comment on blogposts, and an hypothetical issue as an example:
"When trying to insert an image in a blogpost comment, the image doesn't show up."

The lazy way to (not) report a bug

The easy (and lazy) way would be to just write a report with the very little information we have:

Title: Image can't be inserted in blogpost comment

Steps to reproduce:
1. Find a blogpost,
2. Click on the "Add comment" button,
3. Click on the "Insert Image" button,
4. Choose an image and insert it.

Actual result:
The "insert image" dialog is dismissed,
No image is inserted in the text editor.

Expected result:
Image is inserted in the text editor.

But even when the basic guidelines have been followed (*descriptive title, detailed step by step reproduction, clear actual and expected behavior*), this is **not** the best report a QA engineer can deliver. Why? Because *it only covers a very specific use case* and doesn't provide information that let developers know if the bug just affects that very tiny portion of the application or it's affecting a wider variety of functionality. Of course, developers can take a look by themselves but that is actually a Quality Engineer's job.

Time to play detective and investigate

So, to achieve a good report, we must first do some *research* on the issue, to find out if other related functionality is affected. That is, we should come up with some ideas on what else to test

related to the bug we found. In the hypothetical issue above, we might want to investigate the following:

- Does this only happen on blogpost comments or it also happens on nested comments (comments to other comments)?
- Does it happen on comments only or also when creating a blogpost?
- Does it happen when the image is uploaded from local disk and also when it's inserted from an internet URL?
- Does it happen for different image file types and sizes?
- Does it happen if there is something else on the text editor (some text, for instance)? If so, does the position of the image matter (if it's inserted before or after text)?
- Does it change anything if you're the author of the blogpost or it's some other user?
- And so on...

Of course we can't investigate forever, so we want to cover the main use cases only, and exclude some others we feel might not be relevant (it all depends on how much time can be spent on the task).

Strategies

In order to design these ad-hoc test cases we are going to perform, there are some strategies we can follow (just to have a defined path and make things easier). The best strategy could be different for everyone, but this is what works for me:

1. Start by verifying the exact issue that was found to see if it can be reproduced at will and to come up with the exact reproduction steps, e.g.:
    a. Create a blogpost,
    b. Start adding a comment,
    c. Attempt to insert an image.
2. If the issue is reproduced, then start broadening the variables that you take into account. In this case of image insertion, see in what other ways you can insert an image:
    - Try other image types.
    - Try inserting from local disk and from a url.
    - Use drag & drop as well as the "insert image" button if available.
3. Continue broadening the cases you can try. What else can be inserted with an image?
    - Try inserting an image when there's some text in the editor.
    - Try inserting an image when there's another image in the editor.
4. Now go for the bigger picture:
    - Try inserting images on nested comments.
    - Try inserting images on blogpost creation.
    - Try with blogposts created by yourself and by other users.

Preconditions

Is there anything else, not related to the bug itself, that needs to be set up in order to reproduce the bug? For the example I've been using, maybe a certain non-default image type has to be enabled, or maybe some other setting needs to be changed to a non-default value. These are not specifically part of the steps to reproduce the issue but need to be specified. Remember that not all devs or QE have to be thoroughly familiar with the application or the feature involved in the bug, so we need to provide enough information to avoid wasting their time in researching information that should be present in the report.

A good bug report

If we investigated properly, we now know how widely spread the bug is (which also helps determine the severity) and can write a good, comprehensive report for our hypothetical bug:

Title: Local images can't be inserted in nested comments

Preconditions: Nested comments feature is enabled.

Steps to reproduce:
1. Create or find an existing blogpost,
2. Click on the "Add comment" button,
3. Click on the "Insert Image" button in the text editor,
4. Upload an image from your computer (tested file types: jpg, gif, png).

Actual result:
The "insert image" dialog is dismissed but no image is inserted in the editor.
The existence of text or other contents in the editor doesn't change the behavior.
Inserting an image through URL or drag & drop does work as expected, as well as inserting images in first-level comments.

Expected result:
Image is inserted in the text editor at the cursor position.

A few more tips to do a good bug investigation

- Test in different environments: browser, OS, test instances (to make sure it's not a configuration issue).
- With bugs of the "nothing happens" kind, it might be helpful to open the browser dev tools (or use a tool like Firebug) to catch any errors that might be happening in the background. Also looking at application logs might help.
- If there are different ways to do whatever causes a bug, try all of them (e.g.: if linking to content is broken, try copy-paste links as well as any "insert link" tool the text editor provides).
- If the bug was reported by someone else and you are trying to validate it, you can talk to the reporter or the developer to gather more information.
- Start by trying to mimic the exact situation that triggered the bug in the first place and then test similar cases (e.g.: if a specific blogpost name breaks the ability to link to it, begin by using the exact name and then go around trying other variants, like adding or removing spaces, special characters, numbers, longer/shorter words, etc.).
- Think of related functionality that might be affected: if you've found a bug that occurs, let's say, when posting a blogpost, then try other possible variants: if the application allows more than just blogposts, does the bug occur on other content types as well? Are comments also affected? Does the bug only happen to regular users or an admin user also sees it? What about anonymous users? With a text editor related bug, does it only happen in edit mode or it also affects the published view?

UX bugs

"UX" stands for "user experience" and these bugs are sometimes hard to pinpoint. Since UX bugs are actually design bugs (probably originated in the product design process and not during coding), defining them can be a subjective work. But there are some generally accepted heuristics that help determine whether something could cause UX trouble or not. But this is a whole different article that exceeds the information provided here.