Filing a Bug

You will be asked to file a bug for every bug you encounter in this course. We will be using the default/generic fields used in most bug tracking databases. Even though there are many ways to customize these fields for the specific needs of a company, and every database differs slightly, the specific fields you will be required to give will always be needed in some form no matter where you go.

Keep in mind that a bug tracking database is used throughout the entire life cycle of a bug. So fields/values and comments are often changed or added throughout the process for each bug separately. However, you will be specifically asked to file bugs at a particular point in time. **That is, when you "file a bug," you will be filing it as though the bug is new to "the company" (i.e., the class) and you have just "verified" the bug but have not yet found or fixed it.** You will NOT be asked to change these things as you go, but you should keep in mind that in the normal process, you should be constantly keeping these things current. Even if you have already found and fixed the bug before you filed it (which you shouldn't do very often in this course, and should never do on the job), you will file the bug towards the beginning of the life cycle process.

There are several fields that are very important to a company that we simply will not have the proper context to use in this course. But please know they are out there and they can be very important.

*The fields you will **NOT be required to provide** are below with a brief explanation of each:*

- *Version of the program - the specific version number in which the bug is occurring*
- *Resolution - this is always added to the bug report in the end, it is how the bug was resolved. In this course, it will always be assumed that you have eventually found and fixed the bug, so you will not be required to give any bug a "resolution." In real life, you would have to make sure that once you fix a bug, when you commit the code, you always mark the resolution. The resolution can be one of the following:*
    - *Invalid - not an actual issue (usually user error, not a bug)*
    - *Duplicate - the bug has already been filed and this entry is a duplicate of that bug*
    - *Fixed - the bug has been found and fixed*
    - *Wontfix - sometimes the issue can be "valid" but it is not an issue the company intends to fix ("it's a feature!")*
    - *Worksforme - cannot reproduce the bug given the information (can be reopened later if more information is presented)*
- *Assignee - in our case, you will always be the assignee of the bugs you file. But much of the time, companies will have their own unique process to determine who gets assigned what bugs.*
- *Environment - this includes several pieces of relevant information such as the OS, hardware running on, which version of the software, other needed configuration files, etc.*

Since we are all working on similar machines, these issues are not very relevant to us. But don't forget they can sometimes be important in the real world, so as you file bugs, it is good to remember just what you would need in other situations for this field.

The fields you will be required to provide in any bug report are below. Some below have several but specific options. You will typically be graded on how well you choose the most correct option. In truth, some bugs we file might be slightly ambiguous in this sense where there is not a precisely correct and/or a precisely incorrect answer. In most cases though, you will be asked to explain your responses, which we will use for grading purposes.

- **Name** - name of the software/program for which you are filing a bug
- A **one-line summary** that uniquely identifies the issue - this is very important to get precise! It is usually the part that gets searched the most, so it should uniquely identify the bug while being descriptive enough so that everyone reading the summary knows roughly what the bug is (or isn't) doing correctly. It certainly doesn't need to (nor should it!) give every detail. It is a fine line here... which is actually a LOT like naming your variables and functions properly! Each should contain the following (as appropriate):
    - What happened
    - Where it happened
    - Under what circumstances
- **Status** - the current status of the bug. The status can be one of several possibilities outlined below with short explanations for each:
    - *Unconfirmed* - the bug has just been filed by someone outside of the company and has not been verified to be an issue yet by anyone in the company
    - *New* - the bug has been (often quickly) verified by some trusted entity in the company to be, in fact, a bug. It is now needing to be assigned.
    - *Assigned* - some developer in the company is now responsible for finding and fixing the bug
    - *Resolved* - the bug has been "laid to rest" in some form or manner, but it can still be reopened if needed (see "Resolution" below for further details and what values this can take on)
    - *Verified* - the original assignee has found and fixed the bug and has labeled it as resolved. The bug gets labeled as verified once someone in the company (usually testers and/or QA personnel) has verified that the fix was in fact a fix for all possible cases of the bug occurrences and it did not cause other issues
    - *Closed* - the verified fix is in the software and has now been shipped in the product itself so that customers may benefit from the fix
    - *Reopened* - usually, when there is a "kink" in the process somewhere and the bug was mis-labeled or incorrectly resolved, the bug will be reopened. As an employee, you do NOT want it to occur very often (if ever!) that you have filed a bug as resolved but it had to be reopened because the bug still exists in some form. Reopened basically means the bug is still a bug and someone needs to find and fix it.

- **Severity** - how severe the issue is. One should be careful to select the appropriate value for this one because severity is often part of what is used in the triage process to determine which things should be addressed sooner than others. Severity can be one of several possibilities and is chosen in part by whom the bug is currently affecting. They are below with a brief explanation of each:
    - *Blocker* (AKA Showstopper) - blocks development (not necessarily the user, but the developers of the product) and nothing else can happen until this is fixed
    - *Critical* - the program crashes on the customer and/or there is loss of data or memory for the customer
    - *Major* - the customer is experiencing a major loss of functionality of the software
    - *Normal* - just a standard problem that needs to be addressed
    - *Minor* - not something that is causing major harm to the customer... a workaround usually exists so the customer isn't blocked from anything. But they are likely annoyed by something in the software since someone took the time to file a bug and they would like to eventually see this work. This is still actually a bug that needs to eventually be addressed.
    - *Trivial* - this is usually something that is not a big deal at all and is usually cosmetic. If a bug is labeled as trivial, that doesn't mean developers should never address it, but it is not causing any actual issues for anyone.
    - *Enhancement* - this isn't a bug at all but a request for a desired feature.
- **Priority** - how much this particular bug should be prioritized in relation to all the other bugs currently. The values for priority (that we will use in this course) will be on a scale from 1 to 10 with 10 being the highest possible priority and 1 being the lowest. Typically, management will set this one because most companies use the priority as a means for managers to control the flow of work being done. Just give the best answer you think applies and make sure to explain your answer. Remember that severity and priority values together give the most information.
- **Comments** - this is where those involved throughout the process may provide free-form comments. The one filing the bug will usually include all relevant bits of information to reproduce the bug. Others verifying the bug might add further needed or useful information. Developers can add several comments regarding the process of finding the bug. Developers can include things tried, helpful scripts and other files, any dead ends that proved to be fruitless in finding the bug, etc. Eventually, a description of how the bug was found and fixed can be included here. For our purposes, when you are filing the bug, since it is the beginning part of the process, you will file the bug as though it is new and you only have the information to reproduce the bug.