**What do software testers do on a daily basis? What tools do they use? What is the typical work day of a software quality analyst?**

Come to office by 10:00 AM.

Start reading Emails and have a general chat with your colleagues/friends.

Go for a coffee and with your cup of coffee attend the daily standup and update your team about your plan for today.

After this meeting, start doing your work. Test some modules, do smoke testing/retesting/regression. Do Deployment. Address high priority issues. Run any automated tests if you have.

Get more details from your developer/lead for testing the modules if you don't know much about the changes that they did.

If any bugs found, raise that in your defect tracking tool.

Go for lunch. Comeback and discuss some politics with your colleagues.

By this time, the developer who got assigned for that defect will come to your place. Give more details about that defect to your developer and discuss with them if they need any help in fixing that. Sometimes it leads to a severe arguments/quarrel.

Once one of you got convinced, go and resume your testing activities.

By evening go for a cup of coffee and then you might be have a call with your onsite peers/stakeholders.

Put yourself on mute for a hour and finally update your status within 1 or 2 minutes.

# What is the daily work of a automation tester like?

1. Get the requirements from Manual QA/Product teams.
2. Write code/scripts to automate the flows/scenarios.
3. Go and take a cup of coffee.
4. Run them and while it is executing in the second monitor, take a deep nap.
5. Obviously it will fail and then debug.
6. Add waits wherever necessary.
7. Still failing.
8. Go and play ping pong or carom board.
9. Put a Thread.Sleep for a maximum value.
10. Now it gets passed.
11. Check in the code and raise the pull request for code/test review.
12. Close the ticket or move it to "Ready for Review" status.
13. Go for lunch.
14. Start automate the next flow.
15. Go and take a cup of coffee.
16. In between, during code review, they will find out that you have used a constant Thread.Sleep and it will be in your review comments.
17. Work on that and use some dynamic waits.
18. Again send it for review and resume the next workflow.
19. If needed, do some changes/maintenance in the automation framework.
20. Once you are done with development of bunch of workflows, integrate it with any Continuous integration system and run it from there.
21. Update the status and produce the test report it in your daily status meeting/mail/scrum.

The above mentioned work is a daily routine for an automation tester who use UI automation tools like Selenium.

## A Day in the Life of a Software Tester

Your software is acting weird. It’s crashing and the front end isn’t working as expected. Well, I guess it’s time to bring a software tester on board.

Software testers are responsible for finding and reporting bugs to developers. That’s their primary job. But what does the day of a software tester look like? Where does a tester spend their time on? In this post today, I would like to share how a typical day of a tester looks like.

## Bugs all day long

No matter how much expertise your developers have, and no matter how much time they spend with quality assurance. There will always be bugs.

And as a software tester, it’s your responsibility to deal with them.

**A software tester’s main job is to locate and report bugs.**

Software testers need a specific set of skills in order to perform their daily tasks of finding and reporting bugs. Once the bugs are fixed, they need to re-visit these bugs to verify that these bugs are fixed.

What makes their day-to-day tasks challenging is the fact that many bugs are not getting fixed right away. Software testers sometimes have to go back and forth with the development team to clarify the bug, to add more information to the bug to help developers understand the bug better so that they can fix the bug thoroughly.

However, finding bugs or coordinating with developers to clarify bugs is not the hardest part of testing. The toughest part is how to deal with bugs missed. By missed bugs, I mean bugs reported by end users, which are generally expected to be found by testers.

## Coordinating between departments

When a tester finds that the bug she reported is still unfixed, she usually finds out why and then again documents the bug.

A software tester deals in her day-to-day work with a lot of coordination tasks. Making sure that developers have the right amount of information in order to fix an error, making sure that designers make use of your collected user feedback and aligning customers’ needs with their company’s strategy. There is a lot of coordination needed.

## Project updates

Most days of a software tester start with reading emails and project updates. The testers look out for emails from the project manager asking them to run various test cases or tickets.

Testers also scan their mailbox or project management system for emails from their developers giving them an update of the bugs that they had raised the previous day.

After understanding the updates, testers then need to adjust their testing tasks and activities accordingly. Even though people hate change requests, adapting change is part of software testing and testers need to get familiar with that.

## Testing, testing, testing

After finishing the organizational and communication part of your work day, a software tester starts testing by picking up the test case with the highest priority.

Priority is generally decided by the chronology of events that occurs in the course of the final release of the product.

Let’s say that there’s a test case that requires checking the functionality of a search bar on a website. The tester has to ensure that the search is functioning as expected.

This case would come in the highest priority if the ticket has to go live (which means that the users at the production end can see the search bar) within the next few hours of sending the email. The test case or the ticket is exchanged many times among the team (usually the tester and the developer).

The first time a tester works on a ticket, he sends it to the developer. If the developer is not sure about the issue, he’ll send the ticket back to the tester with appropriate comments and a screenshot. Therefore, it is important for the tester to understand each comment that the developer leaves on the ticket so that proper testing can be done.

Often each test ticket will run through a series of smaller, individual tests.

## Test planning

Many software testers plan their testing activities ahead. As a software tester, you’re responsible for planning future test and QA projects, making sure that there are enough resources available for the projects ahead. Specifically, test plan may include the following:

* Test goals or test objectives
* Features to be tested or not tested
* What test approaches to be used
* What test levels are needed
* Exit criteria (E.g.: When to stop testing)
* Risks
* Roles and responsibilities of team members
* Test milestones
* Test deliverables

This gives testing departments a productive start ahead when they start with their next QA efforts.

## Test Reporting

The whole idea of software testing to identify problems or potential problems so that the management can make the informed decision. Finding defects or executing test cases is one part; the other (yet crucial) part is to communicate the test result to management.

As a tester, you’ll need to report as soon as possible (often on a daily basis):

* What you tested and how the test result is
* There’s any critical issue that the management team should know
* How many more tests you need to test
* There’s any roadblock issue blocking your testing

## Wrapping it up

When it comes to software testing, a lot of skills are required to achieve the daily tasks. As a software tester, no day is the same and you’ll face new challenges every day. However, these are the basic and most important activities that you often spend your time on. A day in the life of test will become much more meaningful if you do your best job around these activities.

## 3 Simple Reasons Why Your Bug Report Sucks (and How to Fix It)

First, let me ask you a few questions

Are your bugs often rejected?

Are you bugs often assigned back to you and discussed back and forth to clarify information?

Do your leaders or managers often complain about your bugs?

If the answer is ***YES***, chances that your bug report sucks

In this post today I will point out 3 simple reasons causing bad bug report and how to fix it.

Sounds cool right? Let’s start with. . .

## Reason #1: You under-estimate the importance of reporting a bug

I’ve been in testing for several years and I’ve seen one problem:

***As a tester, you pay too much attention to bug finding. . .***

“Hey, what’s wrong with bug finding? I’m a tester and my job is to find the bug? “ You might ask

No, nothing wrong with that, it’s just this:

***. . . and you pay less attention to bug reporting***

*Why I have to report bug??*

What’s the point if you do testing and find bugs, but you don’t want to report or report them poorly?

It doesn’t sound right, right?

Now, let’s take a few seconds to recall us about the goal of software testing.

***“Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test”***

Also, from Michael Bolton:

***“Testing is not quality assurance, but acts in service to it; we supply information to programmers and managers who have the authority to make decisions about the project***

Who are my customers? In other words, who are my bug report readers?”

It depends on the project or the shop you are in. However, more often, the following people need and read your reports:

* **Your managers:** They need your report to have an overview of the status of the system under test so that they can plan the work or make a Go/ No Go decision.
* **Your developers:** Well, developers may not love you guys testers, but they surely need your bug report so that they can fix the problem accordingly.
* **Your peers:** Your peers need your bug information so that they can refer to, search to know more about the system and to avoid duplicate before submitting a bug.

You see? There are many people relying on your bug report to do their job properly.

If you fail to identify who your customers are and why they need your service, you fail to be a tester.

## Reason #2: You report the invalid bugs

Tell me if this sounds familiar to you?

You spent hours to find a bug.

You were quite sure that your bug was great.

You opened a bug tracking system and reported it right away.

And then. . .

You bug was coldly rejected as Invalid

Your bug is invalid because:

* It’s how the system is designed and you assume that’s a bug.
* It’s a duplicate. Your peer has reported it before.

Needless to say, reporting an invalid bug is totally a waste.

By reporting an invalid bug, you are not only wasting your time but also your manager’s time to review it.

## Reason #3: Your bug is unable to be reproduced

I see testers making this mistake a lot.

Your bugs sound great and valid, but the only problem is that:

**“IT’S UNREPRODUCIBLE”**

Why?

Because you might have missed one important step to reproduce the problem.

How can this happen?

Chances that you didn’t double check yours steps to reproduce before hitting submit button

Chances that you did double check your steps, but you didn’t follow EXACTLY the steps you wrote. You double checked the steps based on your understanding of the bug and you assumed people having the same understanding as yours.

## 4 tips to have a quality bug report

“Okay, now I know why my bug report sucks, how can I fix this?”

To fix this, I would suggest focusing on these 4 key things in your bug reporting

### Tip #1: Your summary

Most of customers include your managers, developers, and peers will read the summary the first thing when they review a bug. This is especially true when they have more bugs to review.

The simple reason is that they don’t have enough time to go into details of every bug, so having a short and concise summary will surely help to grab an idea of what the problem is about and how important it is.

You can have a short and concise summary by telling exactly what problem you found and in what condition.

Don’t write your summary like this:

**“System crashes”**

Write this instead:

**“System crashes when users login and logout”**

Don’t write like this:

**“The Camera does not work properly”**

Write this instead:

**“The camera preview is blurred when taking picture in dark room”**

I can give more examples, but you got the idea right?

Smoke Testing: We do smoke test whenever we receive a new build that is relative unstable to ensure that the critical functionalities of the AUT working fine. The idea of this test is to find critical problems as soon as possible and reject the build so that we don’t waste time to test further (the broken build).

Sanity Testing: We do sanity test whenever we receive a new build that is relative stable to conduct to identify the dependent missing functionalities. In other words, you will validate the major functionalities of the system in a more detailed way.

Both sanity tests and smoke tests are ways to avoid wasting time and effort by quickly determining whether an application is too flawed to merit any rigorous testing.

Retesting: We do retesting for a feature/function which had defects (bugs) & those bugs have been fixed recently.

Regression Testing: We do regression testing for AUT when there are changes including new features / bug fixed to make sure no impact to the current existing functionalities.

To have better understanding, let me introduce you the table of comparison as below:

|  |  |  |  |
| --- | --- | --- | --- |
| Smoke Testing | Sanity Testing | Regression Testing | Re-testing |
| Smoke testing is executed to determine if critical functionalities of AUT are working fine | Sanity testing is executed to determine if the section of AUT is still working as design after some minor changes or bug fixes | Regression testing is executed to confirm whether a recent program or code change has not adversely affected existing features | Retesting is executed to confirm the test cases that failed in the final execution are passing after the defects are fixed |
| The purpose of smoke testing is to verify the "stability" of the system in order to proceed with more rigorous testing | The purpose of sanity testing is to verify the "rationality" of the system in order to proceed with more rigorous testing | The purpose of regression testing is that new changes should not have any side effects to existing functionalities | Re-testing is done on the basis of the defect fixes |
| Defect verification is not the part of smoke testing | Defect verification is not the part of sanity testing | Defect verification is not the part of regression testing | Defect verification is the part of re-testing |
| Smoke testing is executed before regression test | Sanity testing is executed before regression testing, after smoke testing | Based on the project and availability of resources, regression testing can be carried out parallel with Re-testing | - Re-testing is done before we start sanity testing - Priority of re-testing is also higher than regression testing, so it is carried out before regression testing |
| Smoke test can be excuted manually or automatically | Sanity test often is executed manually | You can do automation for regression testing, manual testing could be expensive and time consuming | You cannot automate the test cases for re-testing |
| Smoke testing is subset of Regression testing | Sanity testing is subset of Acceptance testing | Regression testing is only done when there is any modification or changes become mandatory in existing project | Re-testing executes a defect with same data and the same environment with different inputs with new build |
| Test cases of smoke test are part of regression testing, only cover critical functionalities | Sanity Test can be performed without test cases but domain knowledge is required | Test cases for regression testing can be obtained from the functional specification, user tutorials and manuals, and defect reports in regards to corrected problems | Test cases for re-testing can be re-used from previous executed test cases |

Tell me about your daily activities as a test engineer.

1)understanding the frs/brs doc and use cases  
2)if any ambiguity is there in requirement report to team lead /Business analyst  
3)after getting the clarification identify the test scenario and prepare the test scenarios document  
4)prepare the GUI test case document  
5)write the GUI test cases in excel sheel  
6)Build is ready for testing we need to do the sanity testing whether forther testing build is stable or not  
7)perform the functional testing with the positive and negative inputs  
8)if any defect are identified the report/assign to developer  
9)after fixing the defect retest the same component/build  
10)perform the regression testing  
11)perform the compatability testing as per the requirement  
12)generate the test report and send to the Lead/PM