WHAT IS SOFTWARE TESTING?

Software Testing means Verification of Application Under Test (AUT).

Testing is the process, which includes the set of activities to make sure software application/product, is working as per client's requirements.

WHY SOFTWARE FAILS and WHY IS SOFTWARE TETSING IMPORTANT?

An error (or mistake) leads to a defect, which can cause an observed failure.

Testing is the process of identifying defects, where a defect is any variance between actual and expected results. "A mistake in coding is called **Error**, error found by tester is called **Defect**, defect accepted by development team then it is called **Bug**, build does not meet the requirements then it Is **Failure**."

ERROR: An error is a mistake, misconception, or misunderstanding on the part of a software developer.
DEFFECT: Defect is the difference between expected and actual result in the context of testing.
BUG: A bug is the result of a coding error. Bug is terminology of Tester.
FAILURE: A failure is the inability of a software system or component to perform its required functions within specified performance requirements.
FAULT: It is an anomaly in the software that may cause it to behave incorrectly, and not according to its specification.

Testing is important because software bugs could be expensive or even dangerous. Software bugs can potentially cause monetary and human loss; history is full of such examples:

- In November 2005, information on the UK's top 10 wanted criminals was displayed on a website. The publication of this information was described in newspapers and on morning radio and television and, as a result, many people attempted to access the site. The performance of the website proved inadequate under this load and the website had to be taken offline.
- The publicity created performance peaks beyond the capacity of the website.
- A new smartphone mapping application (app) was introduced in September 2012. Among many other problems, a museum was

incorrectly located in the middle of a river, and Sweden's second city, Gothenburg, seemed to have disappeared from at least one map.
- A small, one-line, change in the billing system of an electrical provider blacked out the whole of a major US city.
- In April 2015, Bloomberg terminal in London crashed due to software glitch affected more than 300,000 traders on financial markets. It forced the government to postpone a 3bn pound debt sale.
- Nissan cars have to recall over 1 million cars from the market due to software failure in the airbag sensory detectors. There has been reported two accidents due to this software failure.
- Starbucks was forced to close about 60 percept of stores in the U.S and Canada due to software failure in its POS system. At one point store served coffee for free as they unable to process the transaction.
- Some of the Amazon's third party retailers saw their product price is reduced to 1p due to a software glitch. They were left with heavy losses.
- Vulnerability in window 10. This bug enables users to escape from security sandboxes through a flaw in the win32k system.
- In 2015 fighter plane F-35 fell victim to a software bug, making it unable to detect targets correctly.
- China Airlines Airbus A300 crashed due to a software bug on April 26, 1994, killing 264 innocent live
- In 1985, Canada's Therac-25 radiation therapy machine malfunctioned due to software bug and delivered lethal radiation doses to patients, leaving 3 people dead and critically injuring 3 others.
- In April of 1999, a software bug caused the failure of a $1.2 billion military satellite launch, the costliest accident in history
- In May of 1996, a software bug caused the bank accounts of 823 customers of a major U.S. bank to be credited with 920 million US dollars.
- More interesting cases at https://blog.bitsrc.io/software-is-not-perfect-cases-of-software-failure-and-their-consequences-f5fec39c038f

WHAT IS MANUAL TESTING?

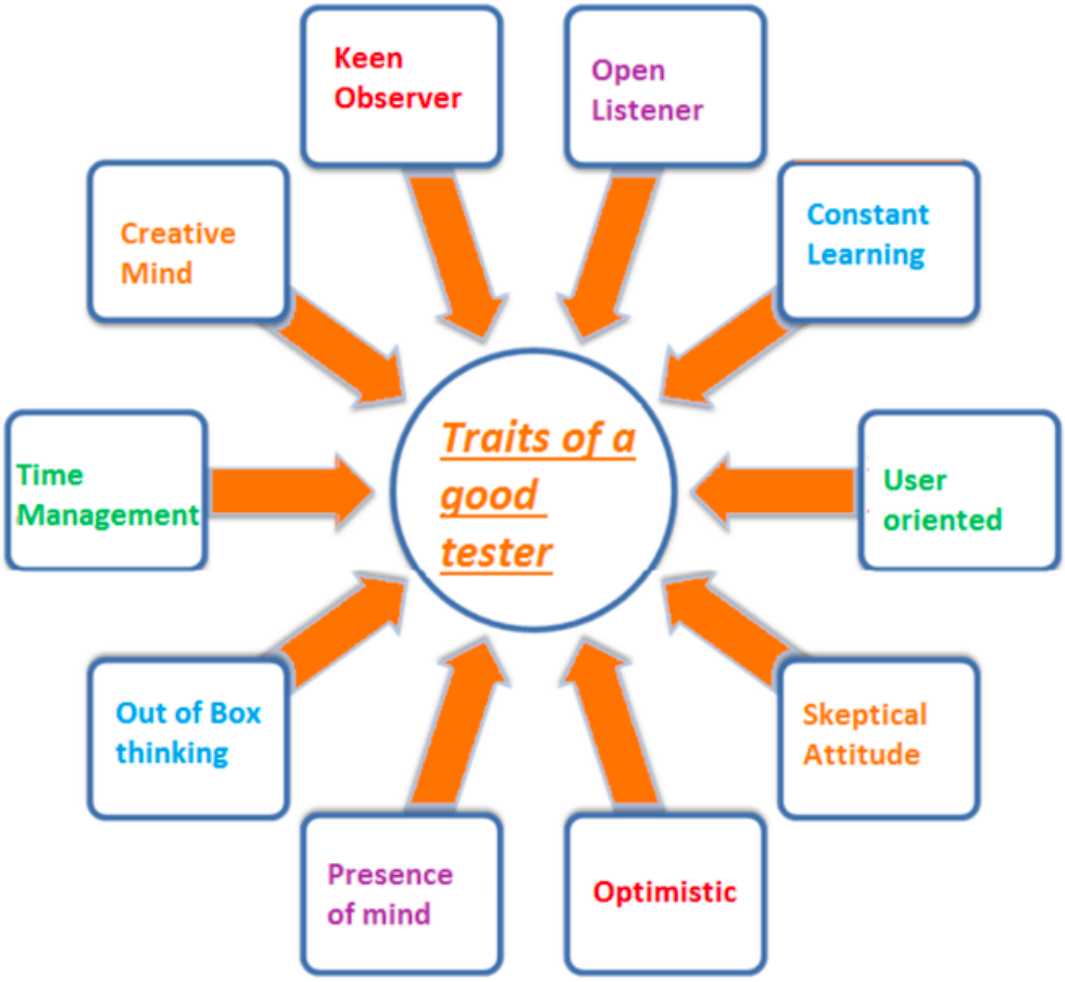
Manual Testing is a type of Software Testing where **Testers manually execute test cases** without using any automation tools.

Manual testing is the most primitive of all testing types and helps find bugs in the software system.

Any new application must be manually tested before it's testing can be automated. Manual testing requires more effort, but is necessary to check automation feasibility.

Manual Testing does not require knowledge of any automated testing tool.

QUALIFICATION OF TESTER

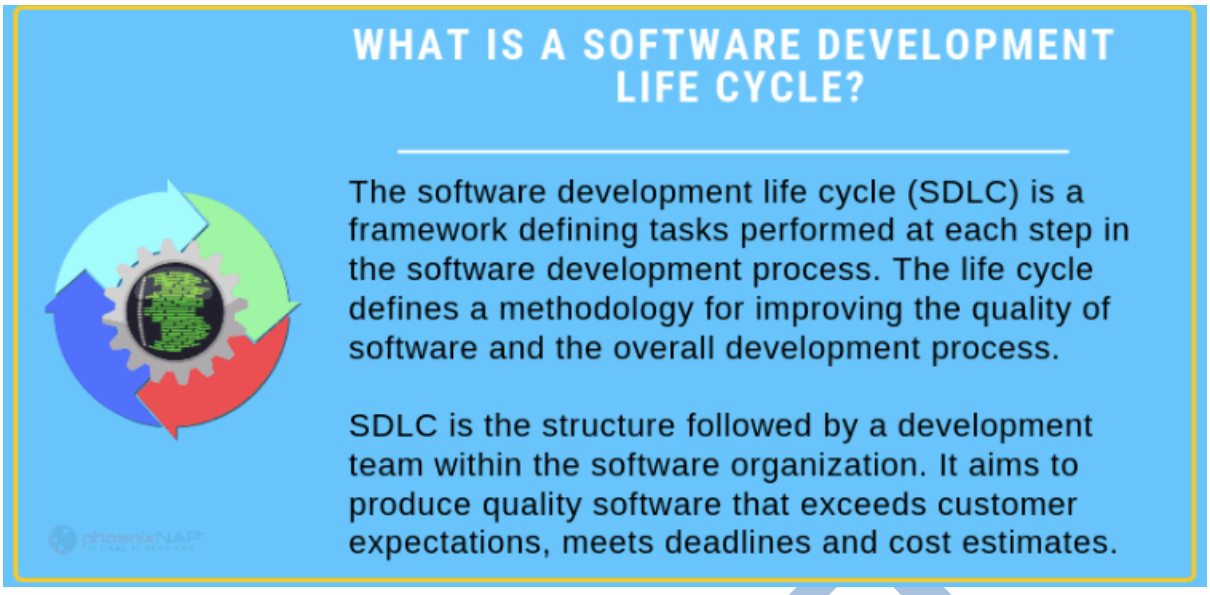


Tester should have an attitude to break the Application (**Finding bugs**)
Attention to details
Ability to research-Ability to search quickly
Quick learner
Strong customer point of view

CHALLENGES OF TESTER

Relationship with developer
Developing automation framework from scratch
Update the test packs when the requirements key point changes
Time constraint: Quality vs Time limit

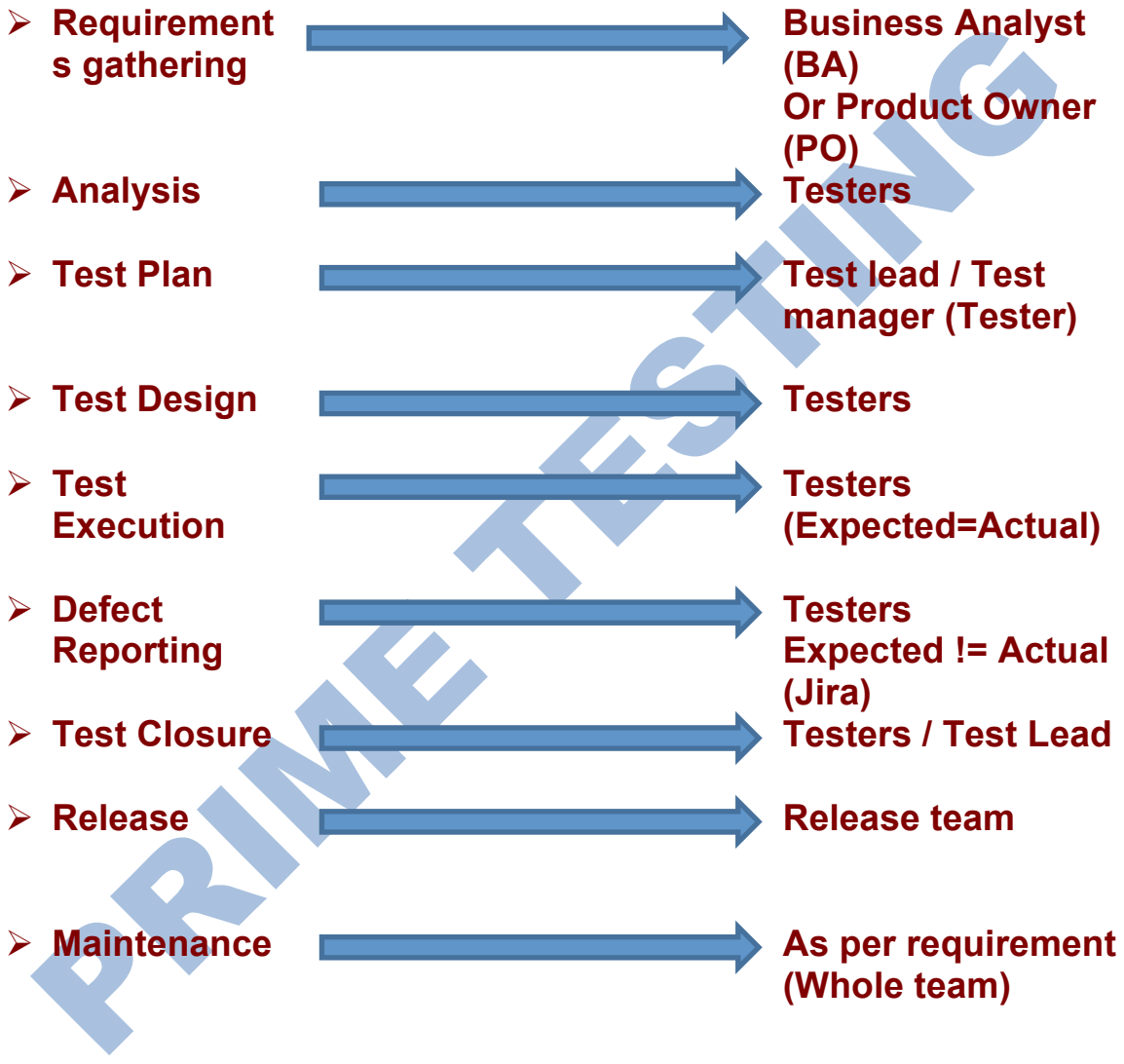
SDLC (SOFTWARE DEVELOPMENT LIFE CYCLE)



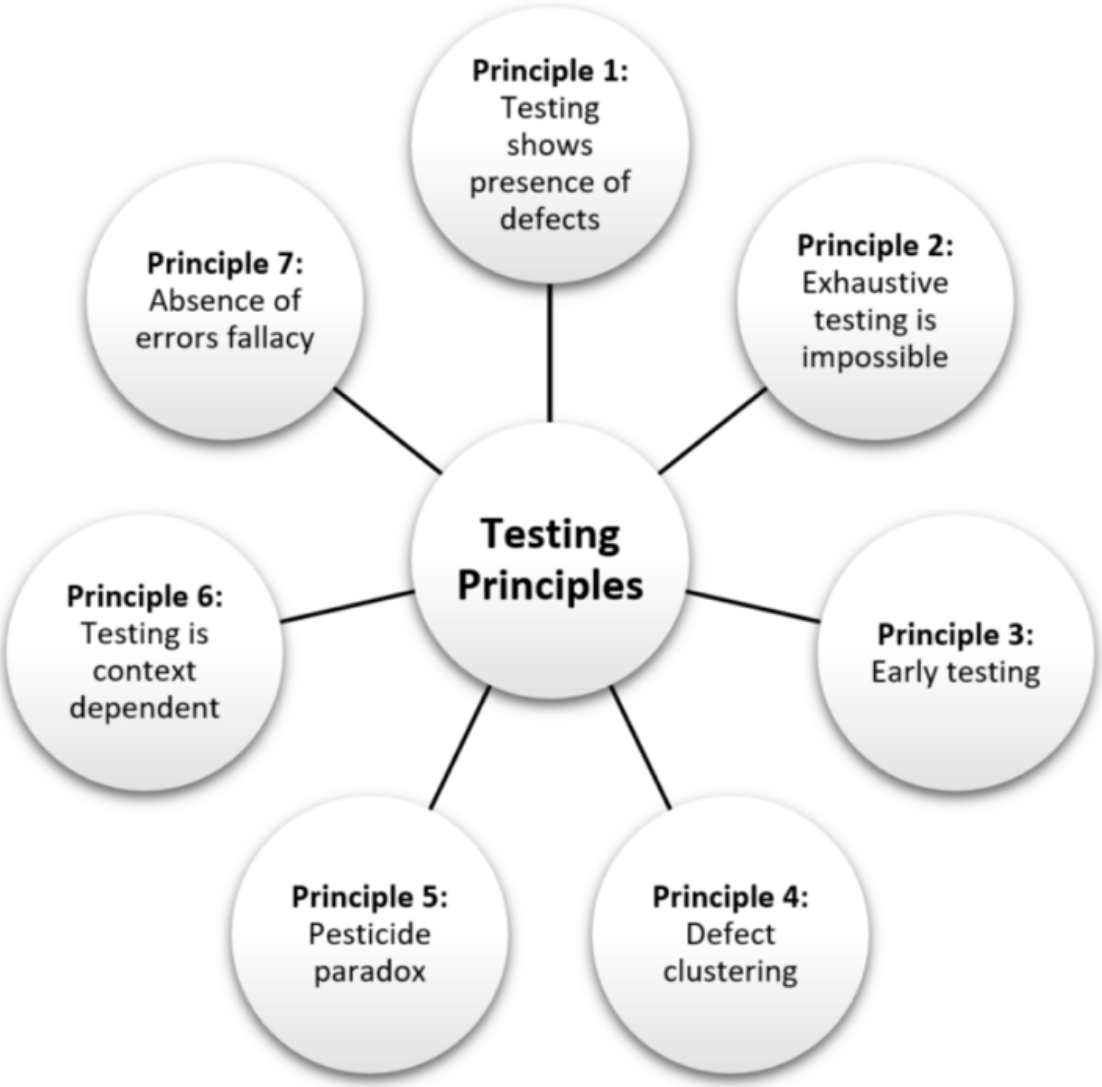
- **Requirements gathering** → **Business Analyst (BA) Or Product Owner (PO) BRS -> SRS/FRS**
- **Analysis** → **Developers**
- **Design** → **Technical Architecture (TA) HLD/LLD**
- **Coding** → **Programmer/Developer (Source code)**
- **Testing** → **Tester (Black Box) Test Execution**
- **Release** → **Release team (Production support team)**
- **Maintenance** → **As per requirement (Whole team)**

STLC (SOFTWARE TESTING LIFE CYCLE)

Software Testing Life Cycle refers to a testing process, which has specific steps to be executed in a definite sequence to ensure that the quality goals have been met. In STLC process, each activity is carried out in a planned and systematic way. Each phase has different goals and deliverables. Different organizations have different phases in STLC; however the basis remains the same.



SEVEN PRINCIPLES OF SOFTWARE TESTING



Testing shows presence of defects: The goal of software testing is to make the software fail. Software testing reduces the presence of defects. Software testing talks about the presence of defects and doesn't talk about the absence of defects. Software testing can ensure that defects are present but it cannot prove that software is defects free. Even multiple testing can never ensure that software is 100% bug-free. Testing can reduce the number of defects but not removes all defects.

Exhaustive testing is not possible: It is the process of testing the functionality of software in all possible inputs (valid or invalid) and pre-conditions are known as exhaustive testing. Exhaustive testing is

impossible means the software can never test at every test case. It can test only some test cases and assume that software is correct and it will produce the correct output in every test case. If the software will test every test cases then it will take more cost, effort, etc. and which is impractical.

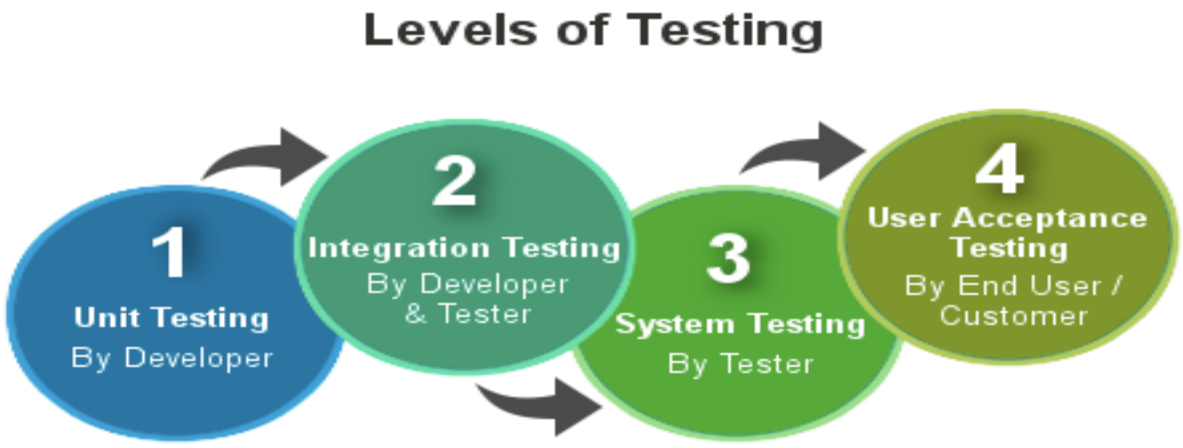
Early Testing: To find the defect in the software, early test activity shall be started. The defect detected in early phases of SDLC will very less expensive. For better performance of software, software testing will start at initial phase i.e. testing will perform at the requirement analysis phase.

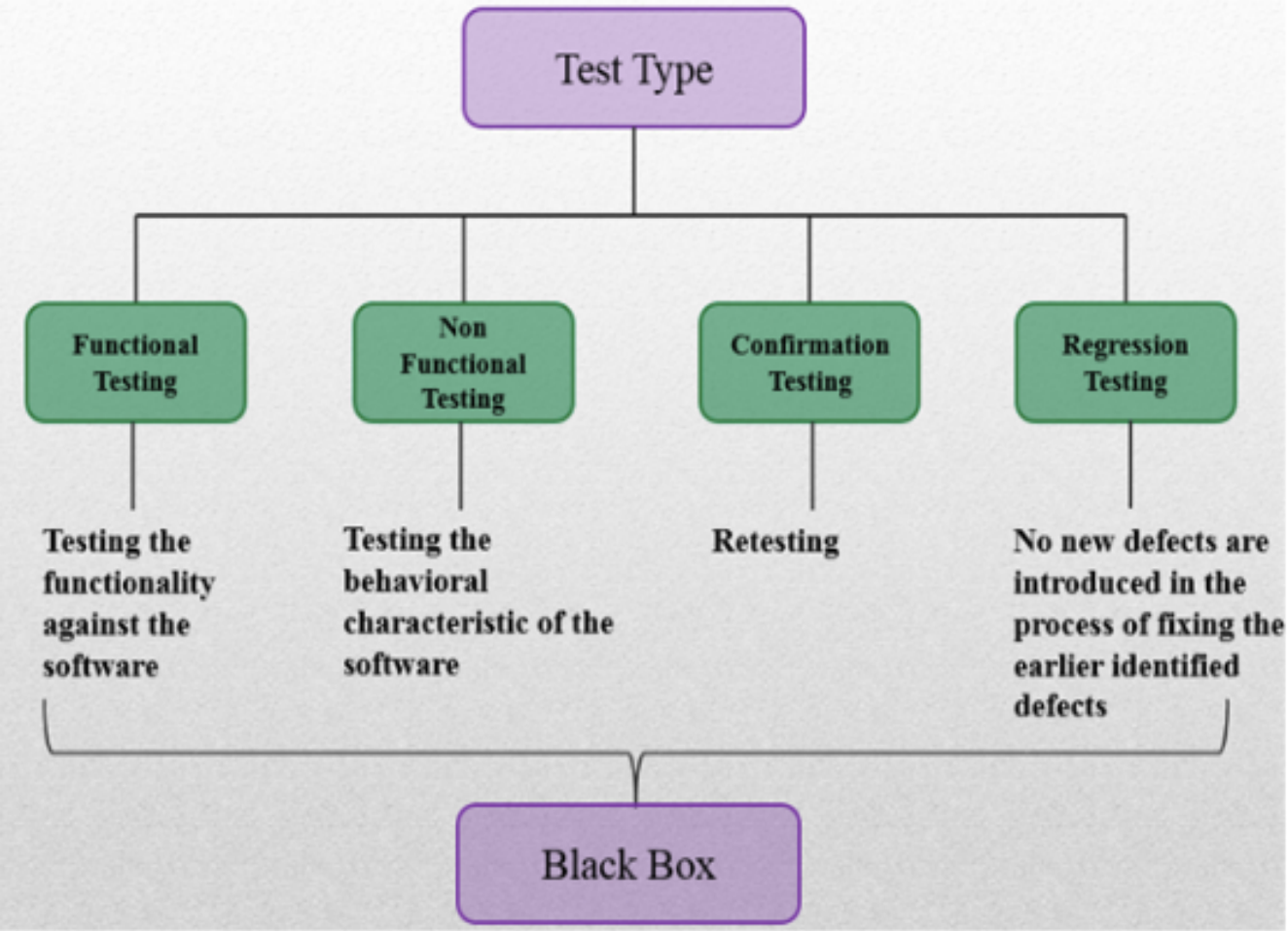
Defect clustering: In a project, a small number of the module can contain most of the defects. Pareto Principle to software testing state that 80% of software defects comes from 20% of modules.

Pesticide paradox: Repeating the same test cases again and again will not find new bugs. So it is necessary to review the test cases and add or update test cases to find new bugs.

Testing is context dependent: Testing approach depends on context of software developed. Different types of software need to perform different types of testing. For example, the testing of the e-commerce site is different from the testing of the Android application.

Absence of errors fallacy: If built software is 99% bug-free but it does not follow the user requirement then it is unusable. It is not only necessary that software is 99% bug-free but it also mandatory to fulfil all the customer requirements.





TEST REQUIREMENTS

As we don't have the official Facebook requirements for the login screen. We'll come up with our own requirement set for the login screen. We'll write the scenario based on these requirements.

- User should be logged in with valid credential.
- User should NOT be logged in with INVALID credential.
- Username should contain letter, number and period.
- Username should not be left blank.
- Username should not be more than 40 characters.
- Username should not start with or contain any symbols.
- Password should be at least 6 characters.
- Password should contain combination of letter, numbers and symbols.
- Password should not contain spaces and period.
- Password should not be more than 40 characters.

Test case template:
A test case template is a document comes under one of the test artefacts, which allows testers to develop the test cases for a particular test scenario in order to verify whether the features of an application are working as intended or not.
Most of the companies are using test case management tools such as JIRA and some of the companies still using excel sheets to write test cases.
Assume we need to write test cases for a scenario (Verify the login of Facebook account).
Here are some test cases.
1. Enter valid User Name and valid Password
2. Enter valid User Name and invalid Password
3. Enter invalid User Name and valid Password
4. Enter invalid User Name and invalid Password

Find the test case template screenshot below:

Project Name	Facebook
Module Name	Login
Reference Document	If any
Create By	Prime Tester
Date of Creation	DD/MM/YY
Date of Review	DD/MM/YY

Test Case Id	Test Scenario	Test Case	Pre-condition	Test Data	Test Steps	Expected Result	Post Condition	Actual Result	Status (Pass/Fail)	Bug id?
TCFB_Login_0001	Verify Login of Facebook	Enter valid user name and valid password	1. Need a valid Facebook account to login	<URL> <Valid user name> <Valid password>	1. Enter user name 2. Enter password 3. Click "Login" button	Successful Login	Facebook home page displayed.			Bug Link

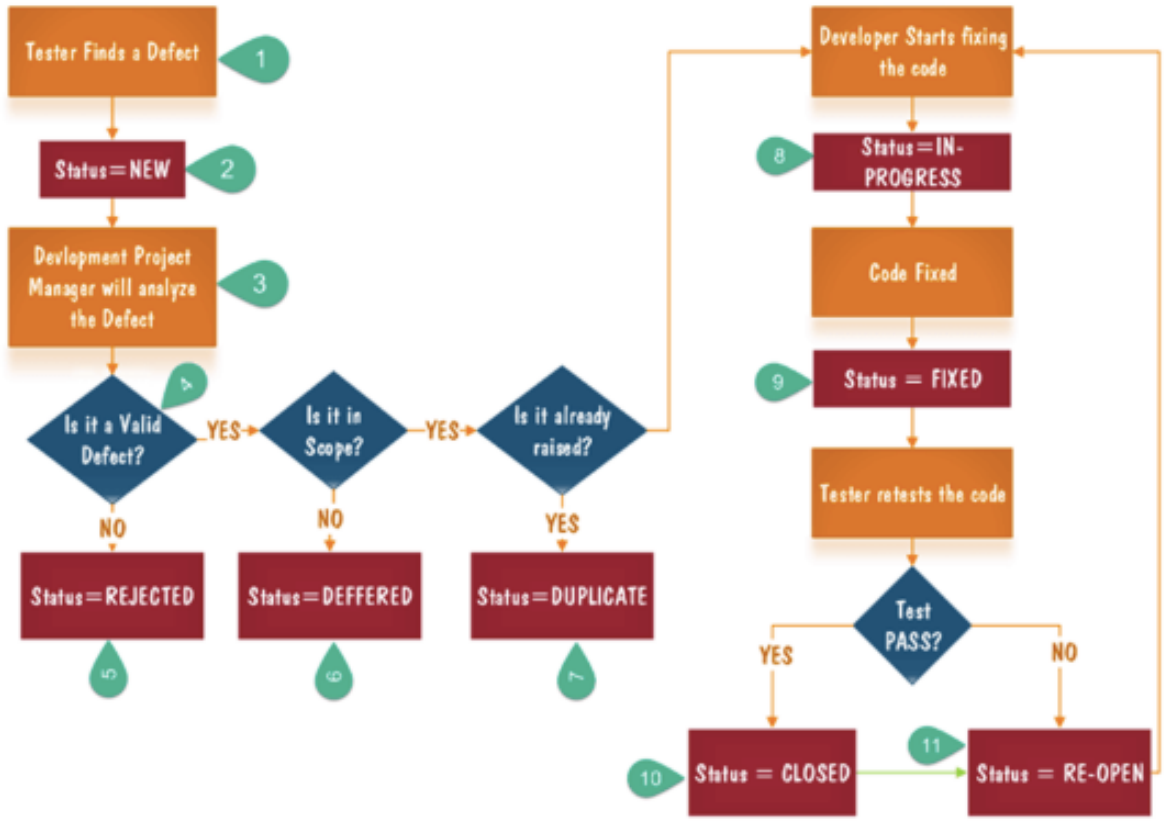
Main fields of a test case:

PROJECT NAME: Name of the project the test cases belong to
MODULE NAME: Name of the module the test cases belong to
REFERENCE DOCUMENT: Mention the path of the reference documents (if any such as Requirement Document, Test Plan, Test Scenarios etc.,)
CREATED BY: Name of the Tester who created the test cases
DATE OF CREATION: When the test cases were created
REVIEWED BY: Name of the Tester who created the test cases
DATE OF REVIEW: When the test cases were reviewed
EXECUTED BY: Name of the Tester who executed the test case
DATE OF EXECUTION: When the test case was executed
TEST CASE ID: Each test case should be represented by a unique ID. It's good practice to follow some naming convention for better understanding and discrimination purpose.
TEST SCENARIO: Test Scenario ID or title of the test scenario.

TEST CASE: Title of the test case
PRE-CONDITION: Conditions, which needs to meet before executing the test case.
TEST STEPS: Mention all the test steps in detail and in the order how it could be executed.
TEST DATA: The data, which could be, used an input for the test cases.
EXPECTED RESULT: The result, which we expect once the test cases were executed. It might be anything such as Home Page, Relevant screen, Error message etc.,
POST-CONDITION: Conditions which needs to achieve when the test case was successfully executed.
ACTUAL RESULT: The result which system shows once the test case was executed.
STATUS: If the actual and expected results are same, mention it as Passed. Else make it as Failed. If a test fails, it has to go through the bug life cycle to be fixed.

Bug life cycle is also known as Defect life cycle. In Software Development process, the bug has a life cycle. The bug should go through the life cycle to be closed. Bug life cycle varies depends upon the tools (Jira) used and the process followed in the organization.

BUG LIFE CYCLE

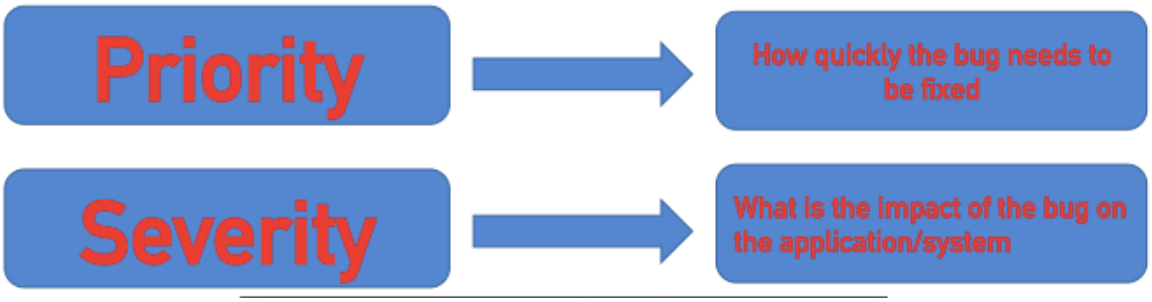


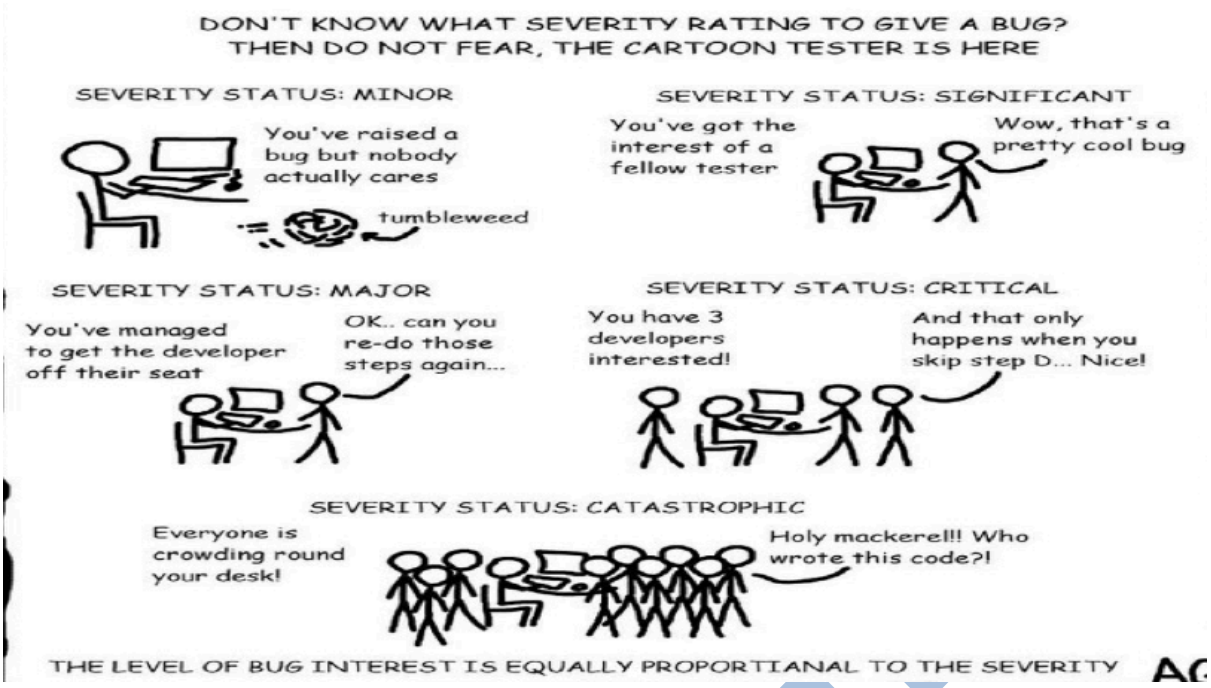
HOW TO REPORT A BUG?

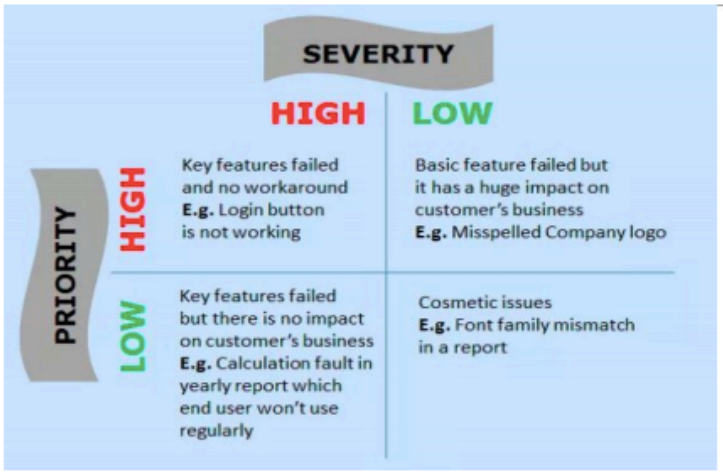
BUG/DEFECT REPORT TEMPLATE

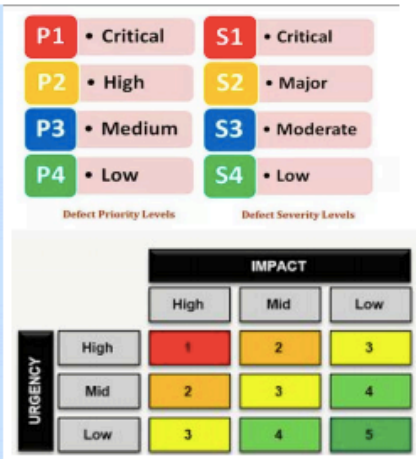
Defect ID – Every bug or defect has it's unique identification number
Defect Description – This includes the abstract of the issue.
Product Version – This includes the product version of the application in which the defect is found.
Detail Steps – This includes the detailed steps of the issue with the screenshots attached so that developers can recreate it.
Date Raised – This includes the Date when the bug is reported
Reported By – This includes the details of the tester who reported the bug like Name and ID
Status – This field includes the Status of the defect like New, Assigned, Open, Retest, Verification,
Closed, Failed, Deferred, etc.
Fixed by – This field includes the details of the developer who fixed it like Name and ID
Date Closed – This includes the Date when the bug is closed
Severity – Based on the severity (Critical, Major or Minor) it tells us about impact of the defect or bug in the software application
Priority – Based on the Priority set (High/Medium/Low) the order of fixing the defect can be made.
(Know more about Severity and Priority)

SEVERITY VS PRIORITY









Top 10 Bug/Issue Tracking Tools