What is a software ?

Software is a set of instructions, data or programs used
to operate computers and execute specific tasks.

What is software testing

Software Testing encompasses all activities that are aimed at discovering errors in a software product or its
components. It involves execution of software/system
components using manual or automated tools to evaluate
one or more properties of interest.

Typical Objectives of Testing

Delivering quality products is the ultimate objective of
testing. Also, whenever someone invests time or

money for an application, keeps expectations as well.Hence, testing refines the software and makes it
as per the user's expectations. Let's have look over the
various objectives of testing:

__(a) Identification of Bugs, and Errors:__
To prevent defects by evaluate work products such as
requirements, user stories, design, and Code.it reduce the level of risk of inadequate software quality.

__(b) Verification of requirements__
To verify whether all specified requirements have been fulfilled
__(c) Offers Confidence__:

To build confidence in the level of quality of the
test object.

<u>*(d) Quality Product*</u>
The main aim of testing is to maintain the quality of
the product. Also, testing has its own cycle and in each phase, all focus revolves around quality only.

<u>*(e) Enhances Growth*</u>
A quality delivery increases the potential of a business. And we all know quality comes
through testing only.

(4) Why is Testing Neccessary / Important ?
Software testing is very important because of the following reasons:
<u>*(a) Pointing out defects and errors*</u>

Software testing is really required to point out the defects and errors that were made during the
development phases.it can help reduce the risk of failures occurring during operation.

Eg: Programmers may make a mistake during the implementation of the software. There could be many reasons for this like lack of experience of the programmer, lack of knowledge of the programming language, insufficient experience
in the domain, incorrect implementation of the algorithm due to complex logic or simply human error.

***(b) Increase realiblity and customer satisfactions*.**
It's essential since it makes sure that the customer finds the organization reliable and their satisfaction
in the application is maintained.

(c) Prevent monetary losses for the testing organisation.

If the customer does not find the testing
organization
reliable or is not satisfied with the quality of the
deliverable, then they may switch to a competitor
organization.Sometimes contracts may also include
monetary penalties with respect to the timeline
and
quality of the product. In such cases, if proper
software
testing may also prevent monetary losses.

(d) Ensure the quality of the product.

It is very important to ensure the quality of the
product.
quality product delivered to the customers helps in
gaining their confidence.

<u>(e) Builds the customers confidence.</u>

As explained in the previous point, delivering good quality product on time builds the customers confidence
in the team and the organization.

<u>(f) Reduces maintenance cost.</u>

High quality product typically has fewer defects and
requires lesser maintenance effort, which in turn means
reduced costs.

<u>(g) Helps to improve the performance of application.</u>

Testing is required for an effective performance of software application or product.

<u>(h) Prevents production/future failures.</u>

It’s important to ensure that the application should not
result into any failures because it can be very expensive
in the future or in the later stages of the development.

(i) Helps to identify issues early.
Proper testing ensures that bugs and issues are detected
early in the life cycle of the product or application.
(j) Reduce development cost and saves time.
If defects related to requirements or design are detected
late in the life cyle, it can be very expensive to fix
them since this might require redesign, reimplementation
and retesting of the application.
(k) Required to increase user satisafaction.

Users are not inclined to use software that has
bugs.
They may not adopt a software if they are not
happy
with the stability of the application.
(l) Prevents business risks.
In case of a product organization or startup which
has
only one product, poor quality of software may
result in
lack of adoption of the product and this may
result in
losses which the business may not recover from.

Testing Principles

(a) Testing shows the presence of defects, not their absence.
It's important to note that not finding defects
does not
make your software bug-free. Because no software
is with

out its defects, testers iterate on their tests to find as
many defects as possible.

__(b) Exhaustive testing is impossible.__

Testing all possible combinations of inputs and preconditions is not typically possible. Rather, teams and
managers should prioritize testing based on risk analysis
to the product and business.

__(c) Early testing saves time and money.__

Testers don't need to wait until the software is deployed to
test it. Bugs discovered earlier in a product's lifecycle are
significantly cheaper and easier to address than if it were a customer-discovered bug.

__(d) Defects cluster together.__

Most software issues follow the Pareto Principle—80% of the issues stem from the same 20% of its modules. While there may be outliers to this, it's a helpful rule for focusing testing.

(e) Beware the Pesticide Paradox.

This borrows from the idea in agriculture that using the same pesticide over and over again will lead to a decline in its efficiency. In the software world, this means that the usual test cases will eventually stop finding new defects. Review and revise tests regularly.

(f) Testing is context-dependent.

A rinse-and-repeat testing model won't work for all scenarios.

For instance, a high-traffic ecommerce website must have
different test cases than an inventory app used by warehouse staff.

(g) Absence of errors is a fallacy.

Software without known issues doesn't equal error-free

software. Finding and fixing some defects won't guarantee
the software's overall success.

Why software have bugs ?

(a) Miscommunication or No Communication.

The success of any software application depends on the
communication between stakeholders, development, and
testing teams. Unclear requirements and misinterpretation
of requirements are the two major factors that cause
defects in software.defects are introduced if exact
requirements are not communicated properly.

(b)Software Complexity

The complexity of the current software applications can be

difficult for anyone with no experience in modern-day
software development.Windows-type interfaces, Client-
Server, and Distributed Applications, Data
Communications, enormous relational databases,
and sheer size of applications have all contributed
to the exponential growth in software/system
complexity.Using object-oriented techniques can
complicate, instead of simplifying, a project unless
it is well-engineered.

(c) Programming Errors

Programmers, like anyone else, can make common
programming mistakes. Not all developers are domain
experts. Inexperienced programmers or
programmers without proper domain knowledge
can introduce simple mistakes while coding.Lack of
simple coding practices, unit testing, debugging

are some of the common reasons for these issues to get introduced at the development stage.

(d) Changing Requirements

The customer may not understand the effects of changes or may understand and anyway request them to redesign,

rescheduling of engineers, effects on the other projects, and the work already completed may have to be redone or thrown out, hardware requirements that may be affected, etc. If there are any minor changes or major changes, known and unknown dependencies, then the parts of the project are likely to interact and cause problems, and the complexity of keeping a track of changes may result in errors. The enthusiasm of engineering staff may be affected. In some fast-changing business environments, continuously changed requirements may be a fact of life.

In these cases, the management must understand the resulting risks, and QA & test engineers must adapt and plan for continuous extensive testing to keep the inevitable bugs from running out of control.

<u>(e)Time Pressures</u>

Scheduling software projects is difficult, often requiring a lot of guesswork. When deadlines loom and the crunch comes,mistakes will be made still. Unrealistic schedules, though not common, the major
concern in small-scale projects/companies results in software bugs. If there is not enough time for proper design, coding, and testing, then it's quite obvious for defects to be introduced.

<u>(f) Egotistical or Overconfident People</u>

People prefer to say things like:

'no problem'

'piece of cake'

'I can whip that out in a few hours'

'It should be easy to update that old code'

Instead of:
‘That adds a lot of complexity and we could end up making a lot of mistakes’.
‘We do not know if we can do that; we’ll wing it’.
‘I can’t estimate how long it will take until I take a closer look at it’.
‘We can’t figure out what that old spaghetti code did in the
first place’.
If there are too many unrealistic ‘no problem’s’, then it results in software bugs.

(g) Poorly Documented code

It’s tough to maintain and modify the code that is badly
written or poorly documented; the result is Software Bugs. In many organizations, management provides no incentive for programmers to document their code or write clear,

understandable code.In fact, it’s usually the opposite: they get points mostly for quickly turning out code, and there’s job security if nobody else can understand it .Any new programmer working on this code may get confused and need to spent more time because of the complexity of the project and the poorly documented code.

(h) Software Development Tools

Visual tools, class libraries, compilers, scripting tools, etc.

often introduce their own bugs or are poorly documented,

resulting in added bugs.

Continuously changing software tools are used by software programmers. Keeping pace with the different versions and their compatibility is a major ongoing issue.

(i) Obsolete Automation Scripts

Writing automation scripts takes a lot of time, especially for complex scenarios. If automation team's record/write any test script but forget to update it over a period, then that test could become obsolete.
If the automation test is not validating the results properly,
then it won't be able to catch the defects.

(j) Lack of Skilled Testers

Having skilled testers with domain knowledge is extremely
important for the success of any project. But appointing all
experienced testers is not possible for all companies.
Domain knowledge and the tester's ability to find defects can produce high-quality software.
Compromise on any of this can result in buggy software.

Testing Terminologies .

(a) What is a bug?

In software testing, a bug is the informal name of
defects, which means that software or application
is not
working as per the requirement. When we have
some
coding error, it leads a program to its breakdown,
which
is known as a bug. The test engineers use the
terminology Bug.

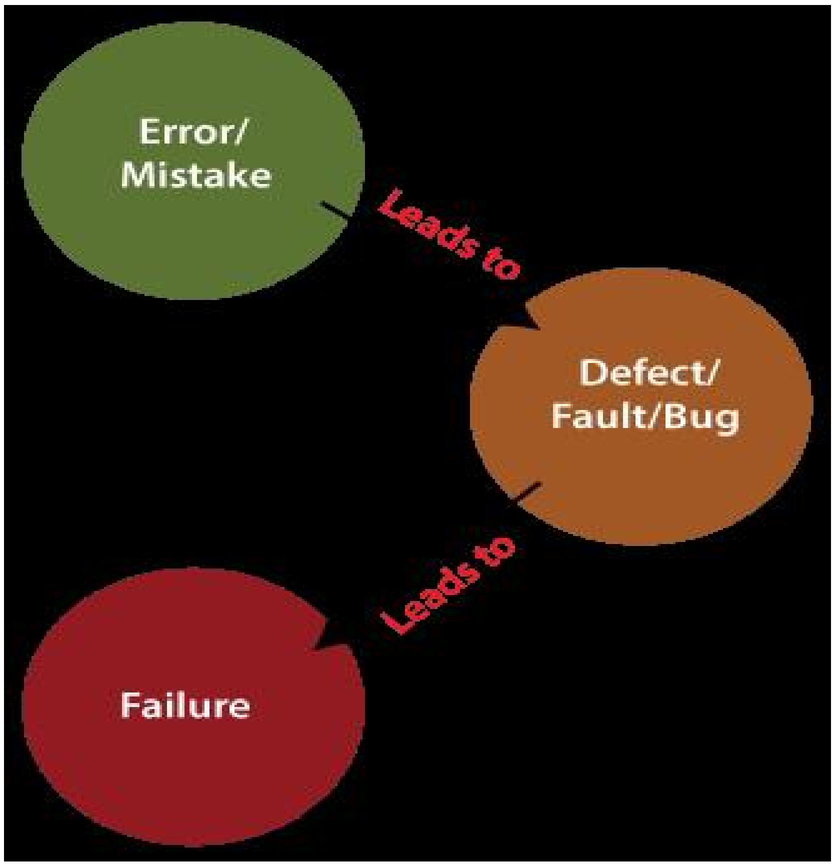
(b) What is a Defect?

When the application is not working as per the
requirement is knows as defects. It is specified as
the
difference from the actual and expected result.

(c) What is Error?

The Problem in code leads to errors, which means
that a

mistake can occur due to the developer's coding error as the developer misunderstood the requirement or the requirement was not defined correctly. The developers use the term error.



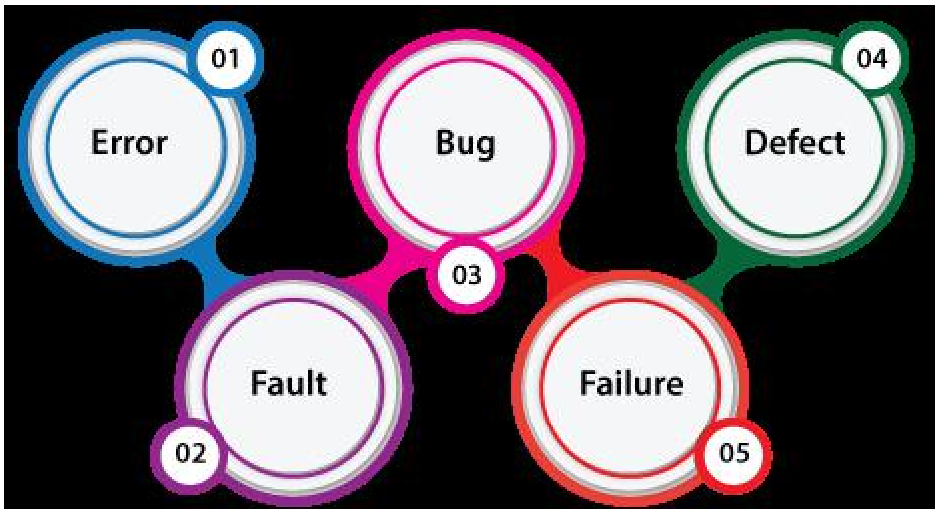
(d) What is Fault?

The fault may occur in software because it has not added the code for fault tolerance, making an application act up.
A fault may happen in a program because of the following
reasons:
- Lack of resources.
- An invalid step.
- Inappropriate data definition.

(e) What is Failure?

Many defects lead to the software's failure, which means that a loss specifies a fatal issue in software/ application or in its module, which makes the system unresponsive or broken.
In other words, we can say that if an end-user detects an
issue in the product, then that particular issue is called a
failure.

Possibilities are there one defect that might lead
to one
failure or several failures.
For example, in a bank application if the Amount
Transfer
module is not working for end-users when the
end-user tries to transfer money, submit button is
not working. Hence, this is a failure.
The flow of the above terminologies are shown in
the
following image:



(f) Verification

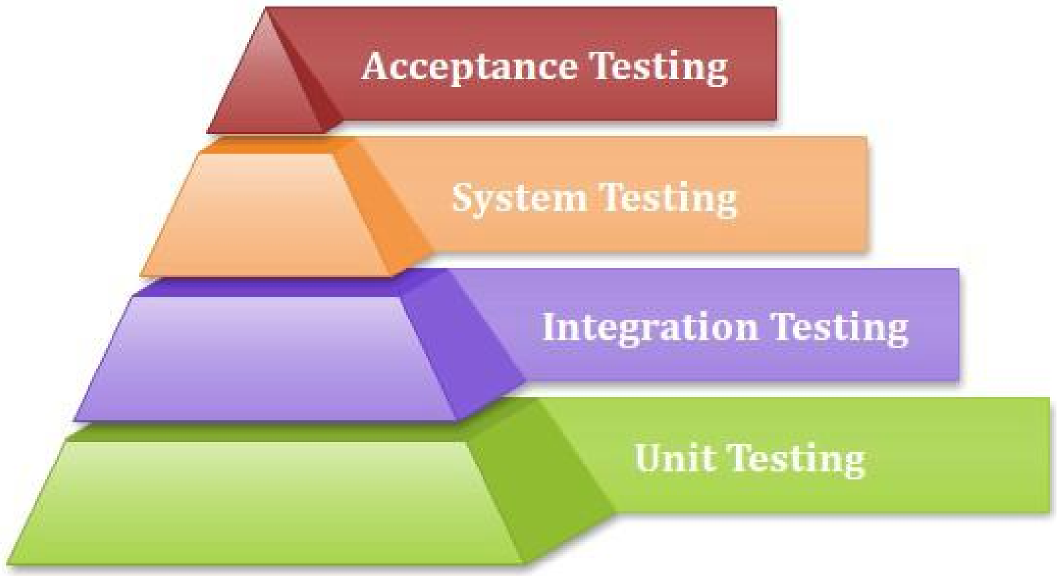
Verification is the process of checking that a software
achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfills the
requirements that we have. Verification is static testing.
Verification means Are we building the product right?

(g) Validation

Validation is the process of checking whether the software
product is up to the mark or in other words product has high level requirements. It is the process of checking the
validation of product i.e. it checks what we are developing is the right product. it is validation of actual and expected

product. Validation is the dynamic testing. Validation means Are we building the right product?

Levels of software Testing



(1) Unit testing

Unit testing involves the testing of each unit or an individual component of the software application. The aim behind unit testing is to validate unit components with its performance.

A unit is a single testable part of a software
system and
tested during the development phase of the
application
software.
The purpose of unit testing is to test the
correctness of
isolated code. A unit component is an individual
function or code of the application. White box
testing approach used for unit testing and usually
done by the developers.
(2) Integration Testing.
To test whether multiple software components
function well together as a group, you will need
integration testing.
Integration means combining. For Example, In this
testing
phase, different software modules are combined
and tested as a group to make sure that
integrated system is ready for system testing.

Integrating testing checks the data flow from one module to other modules. This kind of testing is performed by testers.

(3) System testing:

System testing is performed on a complete, integrated
system. It allows checking system's compliance as per the
requirements. It tests the overall interaction of components. It involves load, performance, reliability and security testing.

System testing most often the final test to verify that the
system meets the specification. It evaluates both functional and non-functional need for the testing.

(4) Acceptance testing:

Acceptance testing is a test conducted to find if the
requirements of a specification or contract are met as per its delivery. Acceptance testing is

basically done by the user or customer. However, other stockholders can be involved in this process.

Types of Testing

(a) Functional Testing

Functional testing verifies that the operational
execution
of a program or mobile app happens according to
the
technical and business requirements. Only if every
feature
of a software system works correctly, it can pass a
functional test.
Functional testing is usually conducted before
nonfunctional
testing and is done manually. The tester
provides specific inputs to the program and
compares the
result with the expected output. Functional testers
are not

concerned about the source code but focus on checking
the functionality.
You can perform functional testing either by following
the manual or automation testing approaches.
Eg: If you test whether a user able to login into a
system or not, after registration, you are doing
functional testing

(b) Non-Functional Testing
Apart from functional requirements that define the users
activities,these requirements specify aspects like
performance, response time,peak load, usability, security
etc.