

## **What is software testing?**

- Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not.

. In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

- According to ANSI/IEEE 1059 standard, Testing can be defined as - A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.

## **Why testing is important?**

[Software Testing](#) is necessary because we all make mistakes. Some of those mistakes are unimportant, but some of them are expensive or dangerous. We need to check everything and anything we produce because things can always go wrong – [humans make mistakes all the time](#).

There are several reasons which clearly tell us as why [Software Testing](#) is [important](#) and what are the major things that we should consider while testing of any product or application.

Software testing is very important because of the following reasons:


1. Software testing is really required to point out the [defects](#) and errors that were made during the [development phases](#).
2. It's essential since it makes sure of the Customer's reliability and their satisfaction in the application.
3. It is very [important](#) to ensure the Quality of the product. Quality product delivered to the customers helps in [gaining](#) their confidence.
4. Testing is necessary in [order](#) to provide the facilities to the customers like the delivery of high quality product or software application which requires lower maintenance cost and hence results into more accurate, consistent and reliable results.
5. Testing is required for an effective performance of [software application](#) or product.
6. 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.
7. It's required to stay in the business.

## **What is Software Quality?**

Quality software is reasonably [bug or defect](#) free, delivered on time and within budget, meets requirements and/or expectations, and is maintainable.

ISO 8402-1986 standard defines quality as “the totality of features and characteristics of a product or service that bears its ability to satisfy stated or implied needs.”


Key aspects of quality for the customer include:

- Good design – looks and style
- Good functionality – it does the job well
- Reliable – acceptable level of breakdowns or failure
- Consistency
- Durable – lasts as long as it should
- Good after [sales](#)  service
- Value for money


### **Good design – looks and style:**

It is very important to have a good design. The application or product should meet all the requirement specifications and at the same time it should be user friendly. The customers are basically attracted by the good looks and style of the application. The right color combinations, font size and the styling of the texts and buttons are very important.


### **Good functionality – it does the job [well](#) :**

Along with the good looks of the application or the product it's very [important](#)  that the functionality should be intact. All the features and their functionality should work as expected. There should not be any deviation in the actual result and the expected result.


### **Reliable – acceptable level of breakdowns or failure:**

After we have tested for all the features and their functionalities it also very [important](#)  that the application or product should be reliable. For example: There is an application of saving the students records. This application should save all the students records and should not fail after entering 100 records. This is called reliability.


### **Consistency:**

The software should have consistency across the application or product. Single software can be multi dimensional. It is very [important](#)  that all the different dimensions should behave in a consistent manner.


### **Durable – lasts as long as it should:**

The software should be durable. For example if software is being used for a year and the number of [data](#)  has exceed 5000 records then it should not fail if number of records increases. The software product or application should continue to behave in the same way without any functional breaks.

### **Good after sales service:**

Once the product is shipped to the customers then maintenance comes into the picture. It is very important to provide good [sales](#)  services to keep the customers happy and satisfied. For example if after using the product for six months the customer realizes to make some changes to the application then those changes should be done as fast as possible and should be delivered to the customers on time with quality.

### **Value for [money](#)** :

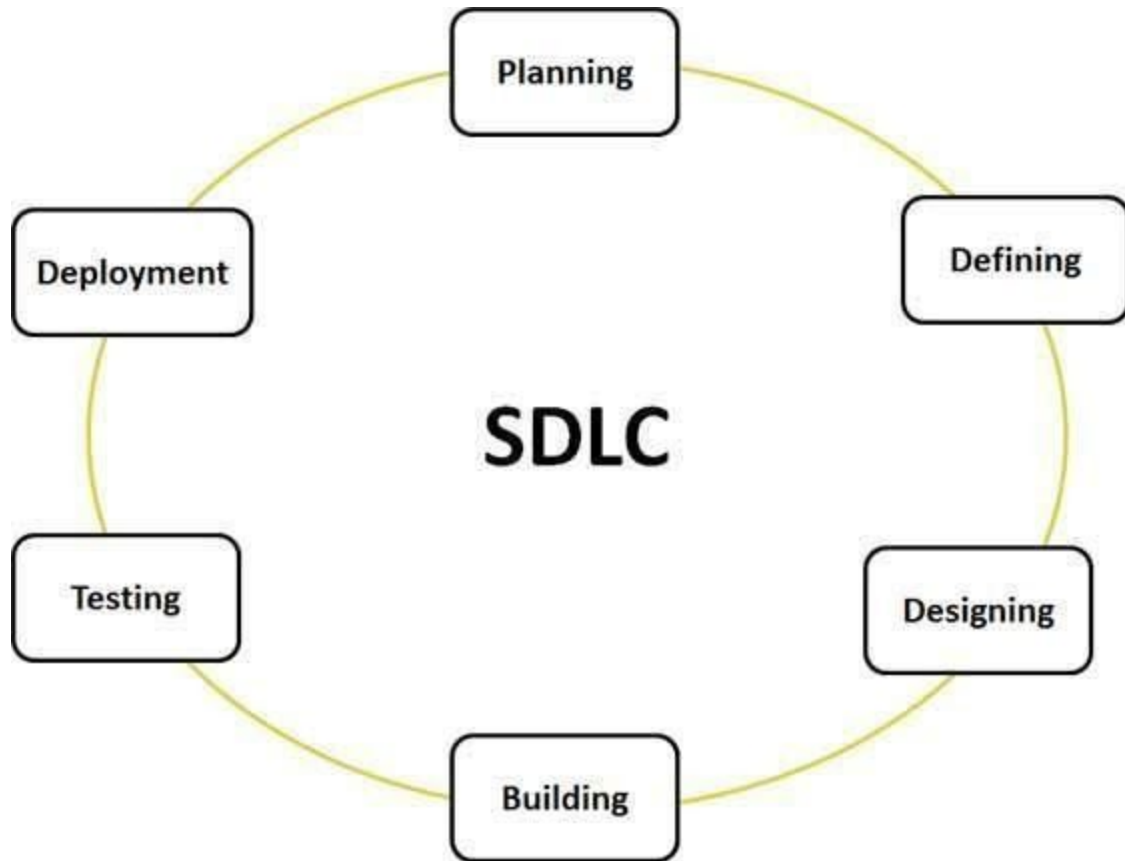
It's always important to deliver the product to the customers which have value for [money](#) . The product should meet the requirement specifications. It should work as expected, should be user friendly. We should provide good services to the customers. Other than the features mentioned in the requirement specifications some additional functionality could be given to the customers which they might not have thought of. These additional functionalities should make their product more user friendly and easy to use. This also adds value for money.

### **SDLC:**

SDLC, Software Development Life Cycle is a process used by software industry to design, develop and test high quality softwares. The SDLC aims to produce a high quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

- SDLC is the acronym of Software Development Life Cycle.
- It is also called as Software development process.
- The software development life cycle (SDLC) is a framework defining tasks performed at each step in the software development process.
- ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.
- SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

- The following figure is a graphical representation of the various stages of a typical SDLC.



- 
- A typical Software Development life cycle consists of the following stages:
- analysis is done the next step is to clearly define and document the product requirements

### **Stage 1: Requirement Analysis**

- Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational, and technical areas.

### **Stage 2: Defining Requirements**

Once the requirement and get them approved from the customer or the market analysts. This is done through .SRS. . Software Requirement Specification document which consists of all the product requirements to be designed and developed during the project life cycle.

### **Stage 3: Designing the product architecture**

- SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

- This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity , budget and time constraints , the best design approach is selected for the product.
- A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.
- **Stage 4: Building or Developing the Product**  
**/Development phase:**
  - In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.
  - Developers have to follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers etc are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java, and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.
- **Stage 5: Testing the Product**
  - This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However this stage refers to the testing only stage of the product where products defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.
- **Stage 6: Deployment in the Market and Maintenance**
  - Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometime product deployment happens in stages as per the organizations. business strategy. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).
  - Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

## SDLC Models

There are various software development life cycle models defined and designed which are followed during software development process. These models are also referred as "Software

Development Process Models". Each process model follows a Series of steps unique to its type, in order to ensure success in process of software development.

Following are the most important and popular SDLC models followed in the industry:

- Waterfall Model
- Iterative Model
- Spiral Model
- V-Model
- Big Bang Model

The other related methodologies are Agile Model, RAD Model, Rapid Application Development and Prototyping Models.

The Waterfall Model was first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

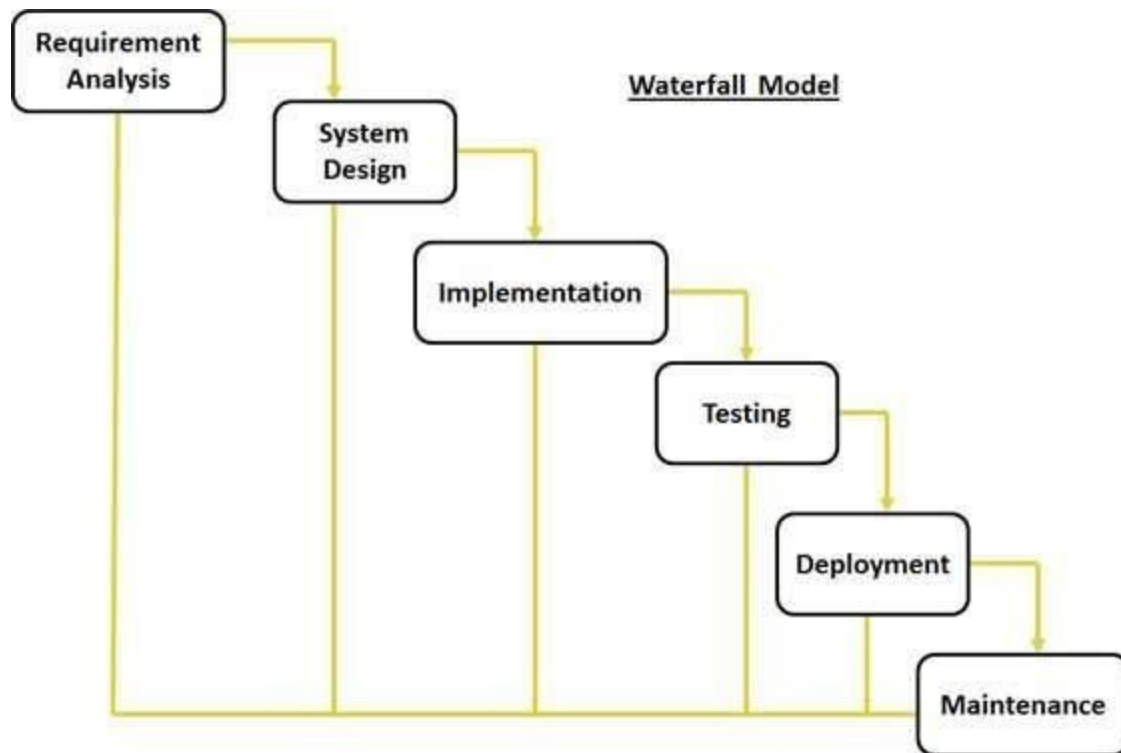
Waterfall model is the earliest SDLC approach that was used for software development .

The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In waterfall model phases do not overlap.

## **Waterfall Model design**

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

Following is a diagrammatic representation of different phases of waterfall model.



The sequential phases in Waterfall model are:

- **Requirement Gathering and analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.
- **System Design:** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.
- **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
- **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system:** Once the functional and non functional testing is done, the product is deployed in the customer environment or released into the market.
- **Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined

set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model phases do not overlap.

## Waterfall Model Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are:

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

## Waterfall Model Pros & Cons

### Advantage

The advantage of waterfall development is that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

### Disadvantage

The disadvantage of waterfall development is that it does not allow for much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The following table lists out the pros and cons of Waterfall model:

Pros	Cons
<ul style="list-style-type: none"><li>• Simple and easy to understand and use</li><li>• Easy to manage due to the rigidity of the model . each phase has specific deliverables and a review process.</li><li>• Phases are processed and completed one at a time.</li></ul>	<ul style="list-style-type: none"><li>• No working software is produced until late during the life cycle.</li><li>• High amounts of risk and uncertainty.</li><li>• Not a good model for complex and object-oriented projects.</li><li>• Poor model for long and ongoing projects.</li></ul>





- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Adjusting scope during the life cycle can end a project.
- Integration is done as a "big-bang" at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

## Agile Model & Methodology: Guide for Developers and Testers

To understand the concept of agile testing, first let's understand-


### What is Agile Methodology?

AGILE methodology is a practice that promotes [continuous](#)  **iteration** of development and testing throughout the [software](#)  development lifecycle of the project. Both development and testing activities are concurrent unlike the Waterfall model

I hope we got an idea of Agile!!! Now, we can step on to Agile Testing.



The agile software development emphasizes on four core values.

1. Individual and team interactions over processes and tools
2. Working software over comprehensive [documentation](#) 
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

## Agile Vs Waterfall Method

Agile and Waterfall model are two different methods for [software](#) development process. Though they are different in their approach, both methods are useful at times, depending on the [requirement](#) and the type of the project.

### Agile Model

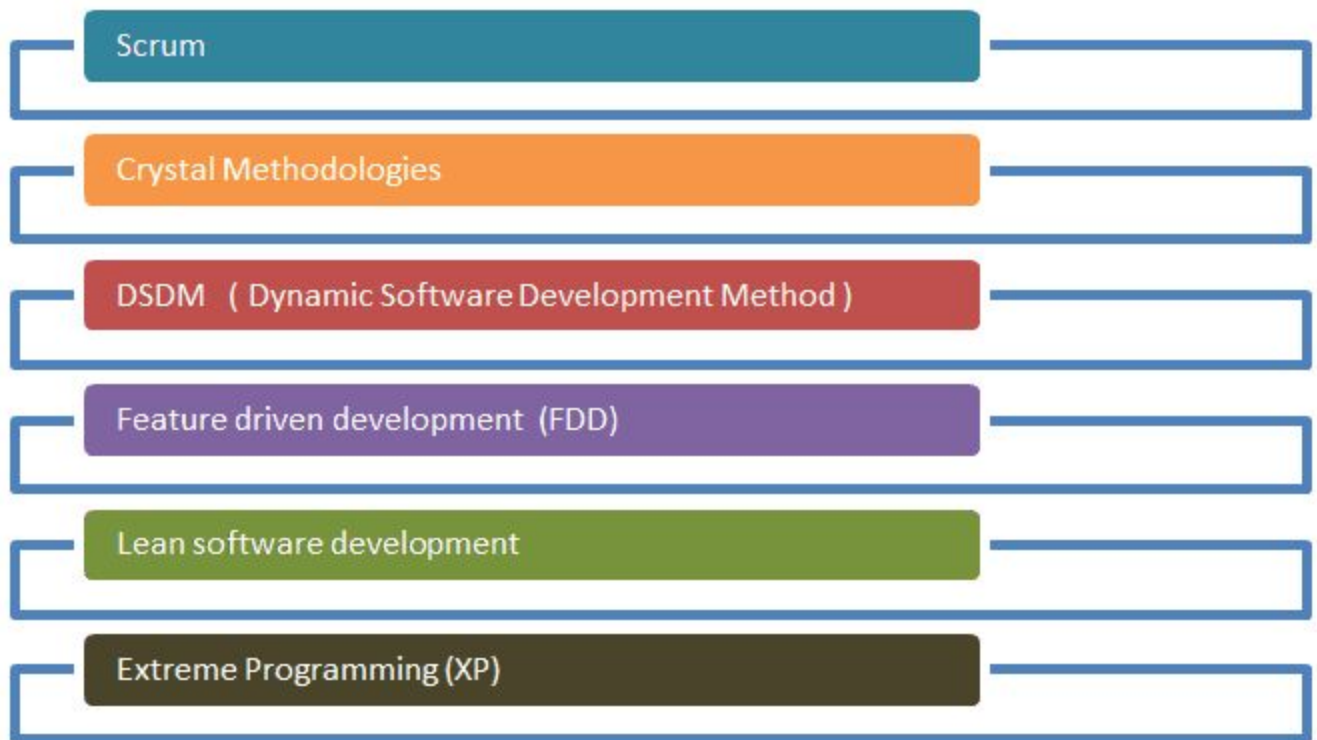
- Agile method proposes incremental and iterative approach to software design
- The **agile process** is broken into individual models that designers work on
- The customer has early and frequent [opportunities](#) to look at the [product](#) and make decision and changes to the project
- Agile model is considered unstructured compared to the waterfall model
- Small projects can be implemented very quickly. For large projects, it is difficult to estimate the development time.
- Error can be fixed in the middle of the project.
- Development process is iterative, and the project is executed in short (2-4) weeks iterations. [Planning](#) is very less.
- Documentation attends less priority than software development
- Every iteration has its own testing phase. It allows implementing regression testing every time new functions or logic are released.
- In agile testing when an iteration end, shippable features of the product is delivered to the customer. New features are usable right after shipment. It is useful when you have good contact with customers.

### Waterfall Model

- Development of the software flows sequentially from start point to end point.
- The design process is not broken into an individual models
- The customer can only see the product at the end of the project
- Waterfall model are more secure because they are so plan oriented
- All sorts of project can be estimated and completed.
- Only at the end, the whole product is tested. If the requirement error is found or any changes have to be made, the project has to [start](#) from the beginning
- The development process is phased, and the phase is much bigger than iteration. Every phase ends with the detailed description of the next phase.
- Documentation is a top priority and can even use for training staff and upgrade the software with another team
- Only after the development phase, the testing phase is executed because separate parts are not fully functional.
- All features developed are delivered at once after the long implementation phase.

- Testers and developers work together
- At the end of every sprint, user acceptance is performed
- It requires close communication with developers and together analyze requirements and planning
- Testers work separately from developers
- User acceptance is **performed** at the end of the project.
- Developer does not involve in requirement and planning process. Usually, time delays between tests and coding

### *Agile Testing Methodology*



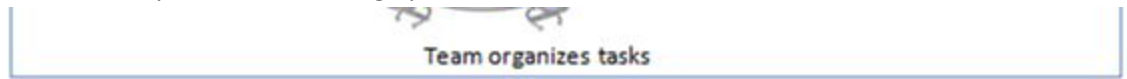
There are various **methods** present in agile testing, and those are listed below:

#### **Scrum**

SCRUM is an agile development method which concentrates specifically on how to manage tasks within a team-based development environment. Basically, Scrum is derived from activity that occurs during a rugby match. Scrum believes in empowering the development team and advocates working in small teams (say- 7 to 9 members). It consists of three roles, and their responsibilities are explained as follows:

- Scrum Master

- Master is responsible for setting up the team



- , sprint meeting and removes obstacles to progress
- Product owner
  - The Product Owner creates product backlog, prioritizes the backlog and is responsible for the delivery of the functionality at each iteration
- Scrum Team
  - Team manages its own work and organizes the work to complete the sprint or cycle

### Product Backlog

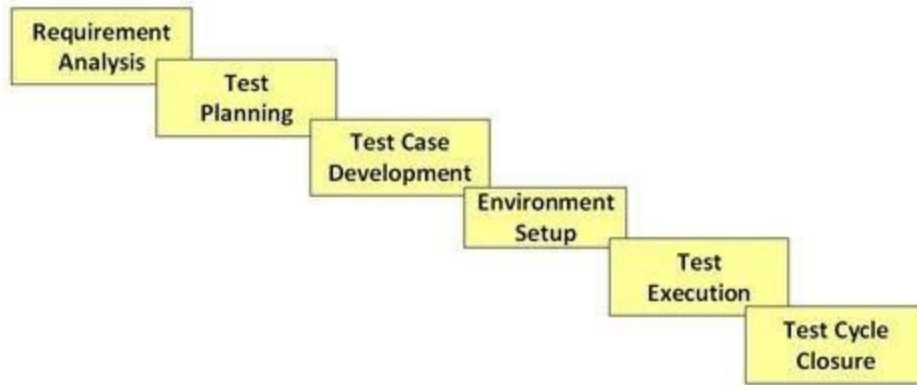
This is a repository where requirements are tracked with details on the no of requirements to be completed for each release. It should be maintained and prioritized by product owner and it should be distributed to the scrum team. Team can also request for a new requirement addition or modification or deletion

### *What is Software Testing Life Cycle (STLC)?*

[Software Testing](#) 🟢 Life Cycle (STLC) is defined as a sequence of activities conducted to perform Software Testing.

It consists of series of activities carried out methodologically to help certify your software product.

Diagram - Different stages in Software Test Life Cycle



Each of these stages have a definite Entry and Exit criteria; , Activities & Deliverables associated with it.

#### What is Entry and Exit Criteria?

**Entry Criteria:** Entry Criteria gives the prerequisite items that must be completed before testing can begin.

**Exit Criteria:** Exit Criteria defines the items that must be completed before testing can be concluded

You have Entry and Exit Criteria for all levels in the [Software Testing](#) Life Cycle (STLC)

In an Ideal world you will not enter the next stage until the exit criteria for the previous stage is met. But practically this is not always possible. So for this tutorial , we will focus of activities and deliverables for the different stages in STLC life cycle. Lets look into them in detail.

#### Requirement Analysis

During this phase, test team studies the [requirements](#) from a testing point of view to [identify](#) the testable requirements.

The QA team may interact with various stakeholders (Client, Business Analyst, Technical Leads, System Architects etc) to understand the requirements in detail.

Requirements could be either Functional (defining what the software must do) or Non Functional (defining system performance /security availability )

.Automation feasibility for the given testing project is also done in this stage. [Activities](#)

- Identify types of tests to be [performed](#) .
- Gather details about testing priorities and focus.
- Prepare [Requirement Traceability Matrix \(RTM\)](#).
- Identify test environment details where testing is supposed to be carried out.
- Automation feasibility analysis (if required).

#### Deliverables

- RTM

- Automation feasibility report. (if [applicable](#) 🟢)

### Test Planning

This phase is also called **Test Strategy** phase. Typically , in this stage, a Senior QA manager will determine effort and cost estimates for the project and would prepare and [finalize](#) 🟢 the Test Plan. **Activities**

- Preparation of test [plan](#) 🟢 /strategy document for various types of testing
- Test tool selection
- Test effort estimation
- Resource planning and determining roles and responsibilities.
- Training requirement

### Deliverables

- [Test plan](#) /strategy document.
- [Effort estimation](#) document.

### Test Case Development

This phase involves creation, verification and rework of test cases & test scripts. [Test data](#) , is identified/created and is reviewed and then reworked as well.

### [Activities](#) 🟢

- Create test cases, automation scripts (if [applicable](#) 🟢)
- Review and baseline test cases and scripts
- Create test data (If Test Environment is available)

### Deliverables

- Test cases/scripts
- Test data

### Test Environment Setup


Test environment decides the [software](#) 🟢 and hardware conditions under which a work product is tested. Test environment set-up is one of the critical aspects of testing process and ***can be done in parallel with Test Case Development Stage. Test team may not be involved in this [activity](#) 🟢*** if the customer/development team provides the test environment in which case the test team is [required](#) 🟢 to do a readiness check (smoke testing) of the given environment. [Activities](#) 🟢


- Understand the required architecture, environment set-up and prepare hardware and software [requirement](#) 🟢 list for the Test Environment.
- Setup test Environment and test data
- Perform smoke test on the build

### Deliverables


- Environment ready with test data set up
- Smoke Test Results.

### Test Execution


During this phase test team will carry out the testing based on the test plans and the test cases prepared. Bugs will be reported back to the development team for correction and retesting will be [performed](#) . **Activities**



- Execute tests as per [plan](#) 
- Document test results, and log defects for failed cases
- Map defects to test cases in RTM
- Retest the defect fixes
- Track the defects to closure

### Deliverables

- [Completed](#)  RTM with execution status
- Test cases updated with results
- Defect reports

### Test Cycle Closure



Testing team will meet , discuss and analyze testing artifacts to identify strategies that have to be implemented in future, taking lessons from the current test cycle. The idea is to remove the process bottlenecks for future test cycles and share best practices for any similar projects in future. [Activities](#) 

- Evaluate cycle completion criteria based on Time,Test coverage,Cost,Software,Critical [Business](#)  Objectives , Quality
- Prepare test metrics based on the above parameters.
- Document the learning out of the project
- Prepare Test closure report
- Qualitative and quantitative reporting of quality of the work [product](#)  to the customer.
- Test result analysis to find out the defect distribution by type and severity.

### Deliverables

- Test Closure report
- Test metrics

[Finally](#) , *summary* of STLC Phases along with Entry and Exit Criteria

STLC Stage	Entry Criteria	Activity	Exit Criteria	Deliverables
Requirement Analysis	<a href="#">Requirements</a> 	Analyse <a href="#">business</a>  functionality	Signed off RTM	RTM
	Document available (both functional and	to know the business modules and module specific	Test automation	Automation

	<p>non functional)</p> <p>Acceptance criteria defined.</p> <p>Application architectural document available.</p>	<p>functionalities.</p> <p>Identify all transactions in the modules.</p> <p>Identify all the user profiles.</p> <p>Gather user interface/authentication, geographic spread requirements.</p> <p><a href="#">Identify</a> types of tests to be performed.</p> <p>Gather details about testing priorities and focus.</p> <p>Prepare Requirement Traceability Matrix (RTM).</p> <p>Identify test environment details where testing is supposed to be carried out.</p> <p>Automation feasibility analysis (if <a href="#">required</a>).</p>	<p>feasibility report signed off by the client</p>	<p>feasibility report (if <a href="#">applicable</a>)</p>
<p><a href="#">Test Planning</a></p>	<p>Requirements Documents</p> <p>Requirement Traceability matrix.</p> <p>Test automation feasibility <a href="#">document</a>.</p>	<p>Analyze various testing approaches available</p> <p>Finalize on the best suited approach</p> <p>Preparation of test plan/strategy document for various types of testing</p> <p>Test tool selection</p> <p>Test effort estimation</p> <p>Resource planning and determining roles and responsibilities.</p>	<p>Approved test plan/strategy document.</p> <p>Effort estimation document signed off.</p>	<p>Test plan/strategy document.</p> <p>Effort estimation document.</p>



Test case development	<p>Requirements Documents</p> <p>RTM and test plan</p> <p>Automation analysis <a href="#">report</a> </p>	<p>Create test cases, automation scripts (where <a href="#">applicable</a> )</p> <p>Review and baseline test cases and scripts</p> <p>Create test data</p>	<p>Reviewed and signed test Cases/scripts</p> <p>Reviewed and signed test <a href="#">data</a> </p>	<p>Test cases/scripts</p> <p>Test data</p>
Test Environment setup	<p>System Design and architecture documents are available</p> <p>Environment set-up plan is available</p>	<p>Understand the required architecture, environment set-up</p> <p>Prepare hardware and software requirement list</p> <p><a href="#">Finalize</a>  connectivity requirements</p> <p>Prepare environment setup checklist</p> <p>Setup test Environment and test data</p> <p>Perform smoke test on the build</p> <p>Accept/reject the build depending on smoke test result</p>	<p>Environment setup is working as per the plan and checklist</p> <p>Test data setup is complete</p> <p>Smoke test is successful</p>	<p>Environment ready with test data set up</p> <p>Smoke Test Results.</p>
Test Execution	<p>Baselined RTM, Test Plan , Test case/scripts are available</p> <p>Test <a href="#">environment</a>  is ready</p> <p>Test data set up is done</p> <p>Unit/Integration test report for the build to be tested is available</p>	<p>Execute tests as per <a href="#">plan</a> </p> <p>Document test results, and log defects for failed cases</p> <p>Update test plans/test cases, if necessary</p> <p>Map defects to test cases in RTM</p> <p>Retest the defect fixes</p> <p>Regression testing of <a href="#">application</a> </p> <p>Track the defects to closure</p>	<p>All tests planned are executed</p> <p>Defects logged and tracked to closure</p>	<p><a href="#">Completed</a> </p> <p>RTM with execution status</p> <p>Test cases updated with results</p> <p>Defect reports</p>

Test Cycle closure	<p>Testing has been completed</p> <p>Test results are available</p> <p>Defect logs are available</p>	<p>Evaluate cycle completion criteria based on - Time, Test coverage , Cost , Software Quality , Critical Business Objectives</p> <p>Prepare test metrics based on the above parameters.</p> <p><a href="#">Document</a> the learning out of the project</p> <p>Prepare Test closure report</p> <p>Qualitative and quantitative reporting of quality of the work product to the customer.</p> <p>Test result analysis to find out the defect distribution by type and severity</p>	<p>Test Closure <a href="#">report</a> signed off by client</p>	<p>Test Closure report</p> <p>Test metrics</p>
--------------------	--	--	---	--

## What is a Test Plan?

Test planning, the most important activity to ensure that there is initially a list of tasks and milestones in a baseline plan to track the progress of the project. It also defines the size of the test effort.

It is the main document often called as master test plan or a project test plan and usually developed during the early phase of the project.

## Test Plan Identifiers:

S.No.	Parameter	Description
1.	Test plan identifier	<p>Unique identifying reference.</p> <p>A brief introduction about the project and to the document.</p>
2.	Introduction	<p>Amazon is web based application which allows users to buy things online as well to sell the things. User can search the product, buy the product and sell the things</p>

	<p>A test item is a software item that is the application under test.</p> <p>Following features have to test:</p> <p>Login function</p> <p>Signup function</p> <p>Search function</p> <p>Add to cart function</p> <p>Checkout function</p>
3. Test items	
4. Features to be tested	<p>A feature that needs to tested on the testware.</p> <p>Identify the features and the reasons for not including as part of testing.</p>
5. Features not to be tested	<p>Following feature are not ready for test in this release:</p> <p>Checkout function</p> <p>Details about the overall approach to testing.</p>
6. Approach	<p>First we have to complete manual testing then after that when major issues will fix then we will do automation testing.</p>
Item pass/fail criteria	
Pass criteria:	
7. When all the major issues are fixed and all the test cases have been executed then we can say our testing passed successfully	<p>Documented whether a software item has passed or failed its test.</p>
Fail criteria:	
When the major issue is not fixed we will consider our testing is failed.	
8. Test deliverables	<p>The deliverables that are delivered as part of the testing process,such as test plans, test specifications and test summary reports.</p>

	Test plan
	Test cases
	Test execution Report
	All tasks for planning and executing the testing.
	Manual testing
9. Testing tasks	Automation testing
	Bugs report
10. Environmental needs	Defining the environmental requirements such as hardware, software, OS, network configurations, tools required.
	For testing purpose we need separate environment.
11. Responsibilities	Lists the roles and responsibilities of the team members.
12. Staffing and training needs	Captures the actual staffing requirements and any specific skills and training requirements.
	Currently we don't need any training
13. Schedule	States the important project delivery dates and key milestones.
14. Risks and Mitigation	High-level project risks and assumptions and a mitigating plan for each identified risk.
15. Approvals	Captures all approvers of the document, their titles and the sign off date.

A Test Strategy document is a high level document and normally developed by project manager. This document defines “Software Testing Approach” to achieve testing objectives. The Test Strategy is normally derived from the Business Requirement Specification document.

The Test Strategy document is a static document meaning that it is not updated too often. It sets the standards for testing processes and activities and other documents such as the Test Plan draws its contents from those standards set in the Test Strategy Document.

Some companies include the “Test Approach” or “Strategy” inside the Test Plan, which is fine and it is usually the case for small projects. However, for larger projects, there is one Test Strategy document and different number of Test Plans for each phase or level of testing.

### Components of the Test Strategy document

- Scope and Objectives
- Business issues
- Roles and responsibilities
- Communication and status reporting
- Test deliverables
- Industry standards to follow
- Test automation and tools
- Testing measurements and metrics
- Risks and mitigation
- Defect reporting and tracking
- Change and configuration management
- Training plan

Here is an example of an [Agile Test Strategy Document Template](#)

## Test Plan

The Test Plan document on the other hand, is derived from the Product Description, Software Requirement Specification SRS, or Use Case Documents.

The Test Plan document is usually prepared by the Test Lead or Test Manager and the focus of the document is to describe what to test, how to test, when to test and who will do what test.

### [How to create a powerful test strategy?](#)

It is not uncommon to have one Master Test Plan which is a common document for the test phases and each test phase have their own Test Plan documents.

There is much debate, as to whether the Test Plan document should also be a static document like the Test Strategy document mentioned above or should it be updated every often to reflect changes according to the direction of the project and activities.

My own personal view is that when a testing phase starts and the Test Manager is “controlling” the activities, the test plan should be updated to reflect any deviation from the original plan. After all, Planning and Control are continuous activities in the formal test process.

## Components of the Test Plan document

- Test Plan id
- Introduction
- Test items
- Features to be tested
- Features not to be tested
- Test techniques
- Testing tasks
- Suspension criteria
- Features pass or fail criteria
- Test environment (Entry criteria, Exit criteria)
- Test deliverables
- Staff and training needs
- Responsibilities
- Schedule

This is a standard approach to prepare test plan and test strategy documents, but things can vary company-to-company.

## What is a Test Scenario?

A Test Scenario is any functionality that can be tested. It is also called **Test Condition or Test Possibility**. As a tester, you may put yourself in the end user's shoes and figure out the real-world scenarios and use cases of the Application Under Test.

## What is Scenario Testing?

Scenario Testing is a variant of Software Testing where Scenarios are Used for Testing. Scenarios help in an Easier Way of Testing of the more complicated Systems

## Why create Test Scenarios?

Test Scenarios are created for following reasons,

- Creating Test Scenarios ensures complete Test Coverage
- Test Scenarios can be approved by various stakeholders like Business Analyst, Developers, Customers to ensure the Application Under Test is thoroughly tested. It ensures that the software is working for the most common use cases.
- They serve as a quick tool to determine the testing work effort and accordingly create a proposal for the client or organize the workforce.
- They help determine the most important end-to-end transactions or the real use of the software applications.

- For studying the end-to-end functioning of the program, Test Scenario is critical.

## When not create Test Scenario?

Test Scenarios may not be created when

- The Application Under Test is complicated, unstable and there is a time crunch in the project.
- Projects that follow Agile Methodology like Scrum, Kanban may not create Test Scenarios.
- Test Scenario may not be created for a new bug fix or [Regression Testing](#). In such cases, Test Scenarios must be already heavily documented in the previous test cycles. This is especially true for Maintenance projects.

## How to create a Test Scenario

As a tester, you can follow these five steps to create Test Scenarios-

- **Step 1:** Read the Requirement Documents like BRS, SRS, FRS, of the System Under Test (SUT). You could also refer uses cases, books, manual, etc. of the application to be tested.
- **Step 2:** For each requirement, figure out possible users actions and objectives. Determine the technical aspects of the requirement. Ascertain possible scenarios of system abuse and evaluate users with hacker's mindset.
- **Step 3:** After reading the Requirements Document and doing your due Analysis, list out different test scenarios that verify each feature of the software.
- **Step 4:** Once you have listed all possible Test Scenarios, a [Traceability Matrix](#) is created to verify that each & every requirement has a corresponding Test Scenario
- **Step 5:** The scenarios created are reviewed by your supervisor. Later, they are also reviewed by other Stakeholders in the project.

## Tips to Create Test Scenarios

- Each Test Scenario should be tied to a minimum of one Requirement or User Story as per the Project Methodology.
- Before creating a Test Scenario that verifies multiple Requirements at once, ensure you have a Test Scenario that checks that requirement in isolation.

- Avoid creating overly complicated Test Scenarios spanning multiple Requirements.
- The number of scenarios may be large, and it is expensive to run them all. Based on customer priorities only run selected Test Scenarios

## What is Traceability Matrix?(TM)

A Traceability Matrix is a document that co-relates any two-baseline documents that require a many-to-many relationship to check the completeness of the relationship.

It is used to track the requirements and to check the current project requirements are met.

## What is RTM (Requirement Traceability Matrix)?

Requirement Traceability Matrix or RTM captures all requirements proposed by the client or software development team and their traceability in a single document delivered at the conclusion of the life-cycle.

In other words, it is a document that maps and traces user requirement with test cases. The main purpose of Requirement Traceability Matrix is to see that all test cases are covered so that no functionality should miss while doing Software testing.

## Requirement Traceability Matrix – Parameters include

- Requirement ID
- Risks
- Requirement Type and Description
- Trace to design specification
- Unit test cases
- Integration test cases
- System test cases
- User acceptance test cases
- Trace to test script

## Types of Traceability Test Matrix

- **Forward traceability:** This matrix is used to check whether the project progresses in the desired direction and for the right product. It makes sure that each requirement is applied to the product and that each requirement is tested thoroughly. It maps requirements to test cases.



- **Backward or reverse traceability:** It is used to ensure whether the current product remains on the right track. The purpose behind this type of traceability is to verify that we are not expanding the scope of the project by adding code, design elements, test or other work that is not specified in the requirements. It maps test cases to requirements.
- **Bi-directional traceability ( Forward+Backward):** This traceability metrics ensures that all requirements are covered by test cases. It analyzes the impact of a change in requirements affected by the [Defect](#) in a work product and vice versa.

How to report defects?

ALM. Bugzilla, jira, rally

What kind information you have to put while reporting defect?

- i) Description of the bug.
- ii) Steps to reproduce:
- iii) Expected Result
- iv) Actual Result
- v) We have to attach screen shot
- vi) Special test data you used .

What is priority and severity?

Priority	severity	
H	H	login functionality not working
L	H	
L	L	
H	L	

Defect life cycle :

New

Open

Assigned

Question

Replied

Re-assign

Fix

Ready for test

Re-open

close

## What is a System Requirements Specification (SRS)?

A **System Requirements Specification (SRS)** (also known as a Software Requirements Specification) is a document or set of documentation that describes the features and behavior of a system or software application. It includes a variety of elements (see below) that attempts to define the intended functionality required by the customer to satisfy their different users. In addition to specifying how the system should behave, the specification also defines at a high-level the main business processes that will be supported, what simplifying assumptions have been made and what **key performance parameters** will need to be met by the system.

Types of testing:

Functional Testing

Non-functional testing

Smoke testing

Performance testing

Sanity testing

Stress testing

Regression testing

Load test

Adhoc testing

Endurance test

## What is Functional Testing?

Functional testing is a type of testing which verifies that each **function** of the software application operates in conformance with the requirement specification. This testing mainly involves black box testing and it is not concerned about the source code of the application.

Each and every functionality of the system is tested by providing appropriate input, verifying the output and comparing the actual results with the expected results. This testing involves checking of User Interface, APIs, Database, security, client/ server applications and functionality of the Application Under Test. The testing can be done either manually or using automation

## What do you test in Functional Testing?

The prime objective of Functional testing is checking the functionalities of the software system. It mainly concentrates on

-

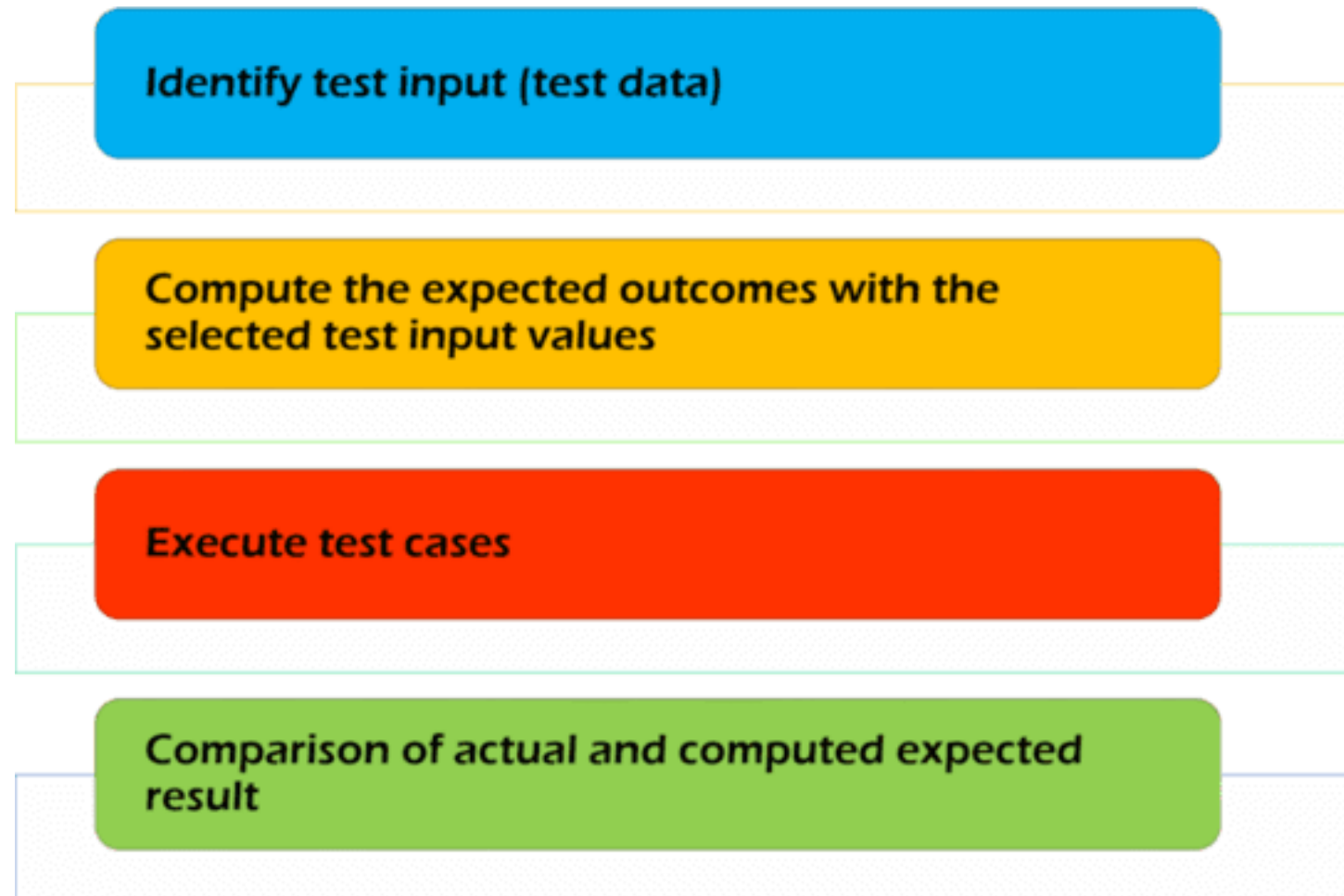
- **Mainline functions:** Testing the main functions of an application
- **Basic Usability:** It involves basic usability testing of the

system. It checks whether an user can freely navigate through the screens without any difficulties.

- **Accessibility:** Checks the accessibility of the system for the user
- **Error Conditions:** Usage of testing techniques to check for error conditions. It checks whether suitable error messages are displayed.

## Functional Testing Process:

In order to functionally test an application, following steps must be observed.



Understand the Requirements  
Identify test input (test data)

A versatile 4-in-1 design.

Compute the expected outcomes with the selected test input values

Execute test cases

Comparison of actual and computed expected result

## Functional Vs Non-Functional Testing:

Functional Testing	Non-Functional Testing
Functional testing is performed using the functional specification provided by the client and verifies the system against the functional requirements.	Non-Functional testing checks Performance, reliability, security, and other non-functional aspects of the software system.
Functional testing is executed first	Non functional testing should be executed after functional testing

	after functional testing
Manual Testing or automation tools can be used for functional testing	Using tools will be effective
Business requirements are the inputs to functional testing	Performance parameters scalability are inputs to non functional testing.
Functional testing describes what the product does	Nonfunctional testing describes how the product works
Easy to do Manual Testing	Tough to do Manual Testing
Types of Functional testing are <ul style="list-style-type: none"> <li>• Unit Testing</li> <li>• Smoke Testing</li> <li>• Sanity Testing</li> <li>• Integration Testing</li> <li>• White box testing</li> <li>• Black Box testing</li> <li>• User Acceptance testing</li> <li>• Regression Testing</li> </ul>	Types of Non functional testing are <ul style="list-style-type: none"> <li>• Performance Testing</li> <li>• Load Testing</li> <li>• Volume Testing</li> <li>• Stress Testing</li> <li>• Security Testing</li> <li>• Installation Testing</li> <li>• Penetration Testing</li> <li>• Compatibility Testing</li> <li>• Migration Testing</li> </ul>

Non-functional testing is a type of testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.

An excellent example of non-functional test would be to check how many people can simultaneously login into a software.

Non-functional testing is equally important as functional testing and affects client satisfaction.

In this tutorial, we will learn

- Objectives of Non-functional testing

- [Characteristics of Non-functional testing](#)
- [Non-functional testing Parameters](#)
- [Type of Testing](#)
- [Non-functional Testing Types](#)

## **Objectives of Non-functional testing**

- Non-functional testing should increase usability, efficiency, maintainability, and portability of the product.
- Helps to reduce production risk and cost associated with non-functional aspects of the product.
- Optimize the way product is installed, setup, executes, managed and monitored.
- Collect and produce measurements, and metrics for internal research and development.
- Improve and enhance knowledge of the product behavior and technologies in use.

## **Characteristics of Non-functional testing**

- Non-functional testing should be measurable, so there is no place for subjective characterization like good, better, best, etc.
- Exact numbers are unlikely to be known at the start of the requirement process
- Important to prioritize the requirements
- Ensure that quality attributes are identified correctly

## **Non-functional testing Parameters**

Security	Availability	Efficiency	Integr
Reliability	Survivability	Usability	Flexibi
Scalability	Reusability	Interoperability	Portab

## Non Functional Testing Parameters

- **Security:**

The parameter defines how system is safeguarded against deliberate and sudden attacks from internal and external sources. This is tested via Security Testing.

- **Reliability:**

The extent to which any software system continuously performs the specified functions without failure. This is tested by [Reliability Testing](#)

- **Survivability:**

The parameter checks that the software system continues to function and recovers itself in case of system failure. This is checked by [Recovery Testing](#)

- **Availability:**

The parameter determines the degree to which user can depend on the system during its operation. This is checked by [Stability Testing](#).



- **Usability:**

The ease with which the user can learn, operate, prepare inputs and outputs through interaction with a system. This is checked by [Usability Testing](#)

- **Scalability:**

The term refers to the degree in which any software application can expand its processing capacity to meet an increase in demand. This is tested by [Scalability Testing](#)

- **Interoperability:**

This non-functional parameter checks a software system interfaces with other software systems. This is checked by [Interoperability Testing](#)

- **Efficiency:**

The extent to which any software system can handles capacity, quantity and response time.

- **Flexibility:**

The term refers to the ease with which the application can work in different hardware and software configurations. Like minimum RAM, CPU requirements.

- **Portability:**

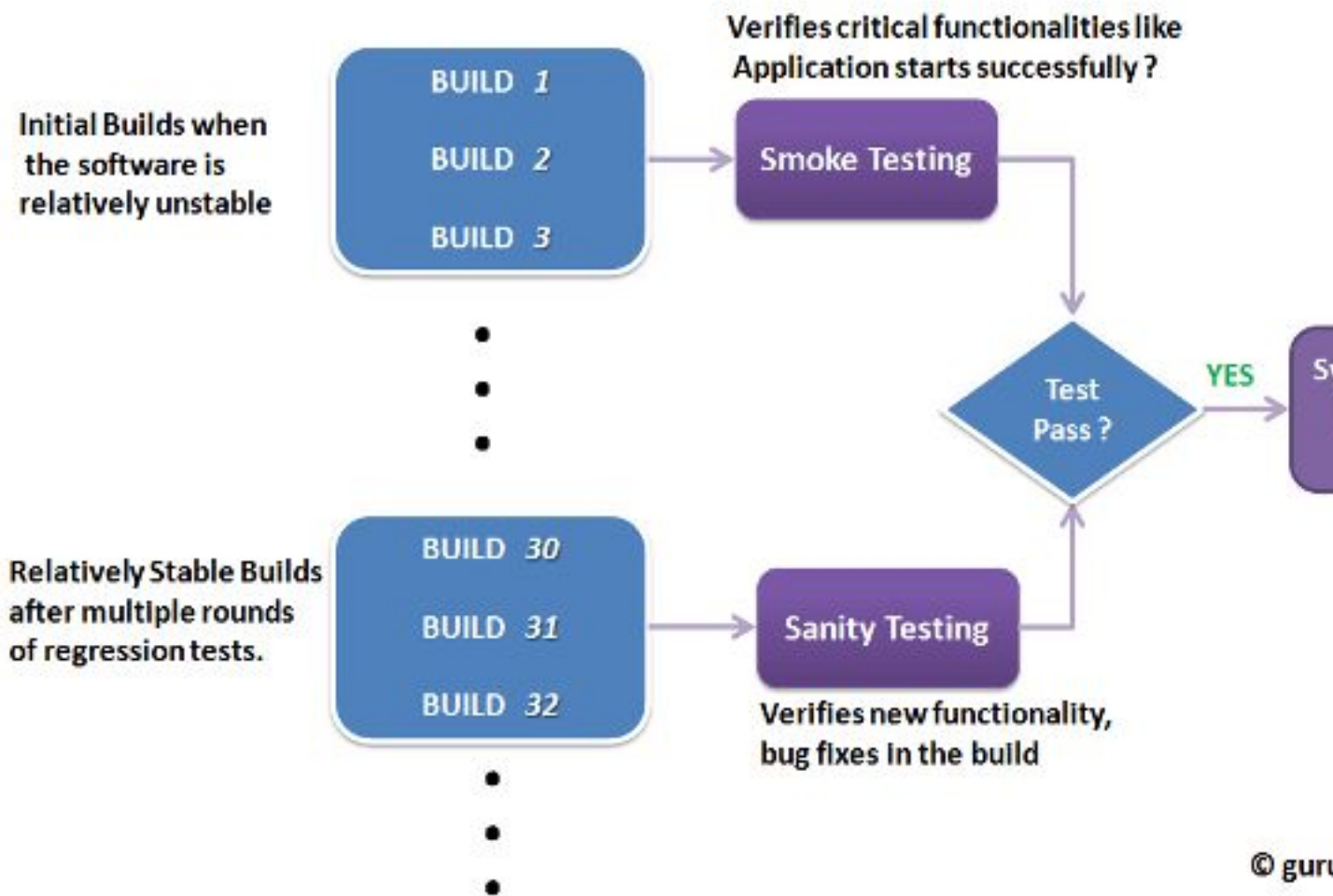
The flexibility of software to transfer from its current hardware or software environment.

- **Reusability:**

It refers to a portion of the software system that can be converted for use in another application.

Smoke and Sanity testing are the most misunderstood topics in Software Testing. There is enormous amount of literature on the subject, but most of them are confusing. The following article makes an attempt to address the confusion.

The key differences between Smoke and Sanity Testing can be learned with the help of following diagram -



To appreciate the above diagram, let's first understand -

```
.projectFill{fill:#26BFD4;} .copy02MaskFill{fill:#ffffff;}  
.copyPath01{fill:none;stroke:#ffffff;stroke-width:6;stroke-miterlimit:10;  
}  
.copyPath02{fill:none;stroke:#ffffff;stroke-width:7;stroke-miterlimit:10;  
}  
.copyPath03{fill:none;stroke:#ffffff;stroke-width:4;stroke-miterlimit:10;  
}  
.copyPath04{fill:none;stroke:#ffffff;stroke-width:5;stroke-miterlimit:10;  
}
```

## What is a Software Build?

If you are developing a simple computer program which consists of only one source code file, you merely need to compile and link this one file, to produce an executable file. This process is very simple.

Usually this is not the case. A typical Software Project consists of hundreds or even thousands of source code files. Creating an executable program from these source files is a complicated and time-consuming task.

You need to use "build" software to create an executable

program and the process is called " *Software Build*"

## **What is Smoke Testing?**

**Smoke Testing** is a kind of Software Testing performed after software build to ascertain that the critical functionalities of the program is working fine. It is executed "before" any detailed functional or regression tests are executed on the software build. The purpose is to reject a badly broken application, so that the QA team does not waste time installing and testing the software application.

In Smoke Testing, the test cases chosen cover the most important functionality or component of the system. The objective is not to perform exhaustive testing, but to verify that the critical functionalities of the system is working fine.

For Example a typical smoke test would be - Verify that the application launches successfully, Check that the GUI is responsive ... etc.

## **What is Sanity Testing?**

Sanity testing is a kind of Software Testing performed after receiving a software build, with minor changes in code, or functionality, to ascertain that the bugs have been fixed and no further issues are introduced due to these changes. The goal is to determine that the proposed functionality works roughly as expected. If sanity test fails, the build is rejected to save the time and costs involved in a more rigorous testing.

Today's Mortgage Rate

Reasons:

Minor changes have done by developers

Tester found defects and developers fixed that defect by changing code

The objective is "not" to verify thoroughly the new functionality, but to determine that the developer has applied some rationality (sanity) while producing the software. For instance, if your scientific calculator gives the result of  $2 + 2 = 5$ ! Then, there is no point testing the advanced functionalities like  $\sin 30 + \cos 50$ .

## Smoke Testing Vs Sanity Testing - Key Differences

Smoke Testing	Sanity Testing
Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine	Sanity Testing is done to check functionality / bugs have been fixed
The objective of this testing is to verify the "stability" of the system in order to proceed with more rigorous testing	The objective of the testing is to verify the "rationality" of the system in order to proceed with more rigorous testing
This testing is performed by the developers or testers	Sanity testing is usually performed by testers
Smoke testing is usually documented or scripted	Sanity testing is usually not documented or scripted
Smoke testing is a subset of <a href="#">Regression Testing</a>	Sanity testing is a subset of Acceptance testing
Smoke testing exercises the entire system from end to end	Sanity testing exercises only one component of the entire system
Smoke testing is like General Health Check Up	Sanity Testing is like special health check up

## Points to note.

- 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.
- Sanity Testing is also called tester acceptance testing.
- Smoke testing performed on a particular build is also known as a build verification test.
- One of the best industry practice is to conduct a Daily build and smoke test in software projects.
- Both smoke and sanity tests can be executed manually or using an automation tool. When automated tools are used, the tests are often initiated by the same process that generates the build itself.
- As per the needs of testing, you may have to execute both Sanity and Smoke Tests on the software build. In such cases, you will first execute Smoke tests and then go ahead with Sanity Testing. In industry, test cases for Sanity Testing are commonly combined with that for smoke tests, to speed up test execution. Hence, it's a common that the terms are often confused and used interchangeably

## What is Regression Testing?

Regression Testing is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features.

Regression Testing is nothing but full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.

This testing is done to make sure that new code changes

should not have side effects on the existing functionalities. It ensures that old code still works once the new code changes are done.

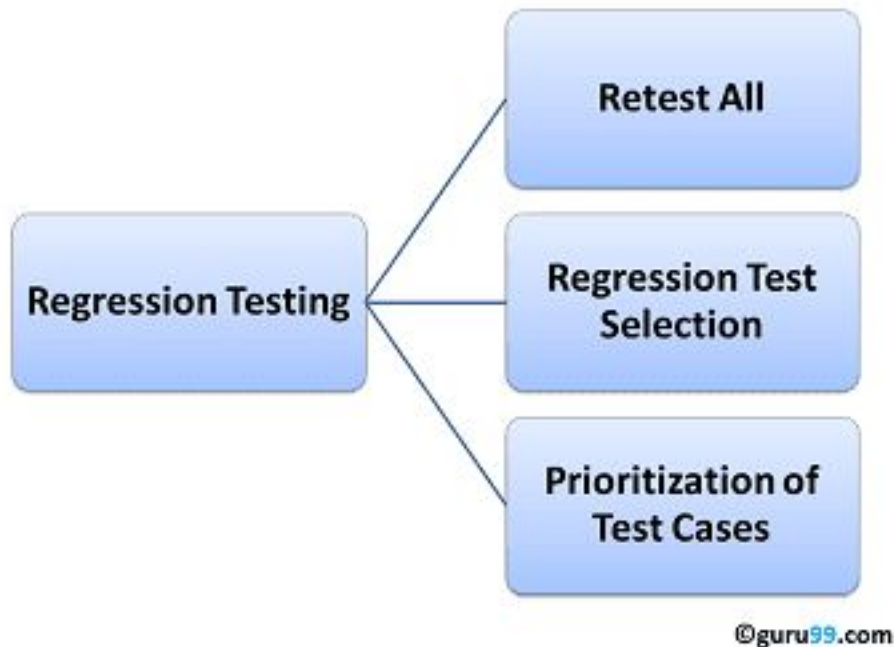
## **Need of Regression Testing**

Regression Testing is required when there is a

- Change in requirements and code is modified according to the requirement
- New feature is added to the software
- Defect fixing
- Performance issue fix

## **Regression Testing Techniques**

Software maintenance is an activity which includes enhancements, error corrections, optimization and deletion of existing features. These modifications may cause the system to work incorrectly. Therefore, Regression Testing becomes necessary. Regression Testing can be carried out using following techniques:



## Retest All

- This is one of the methods for Regression Testing in which all the tests in the existing test bucket or suite should be re-executed. This is very expensive as it requires huge time and resources.

## Regression Test Selection

- Instead of re-executing the entire test suite, it is better to select part of test suite to be run
- Test cases selected can be categorized as 1) Reusable Test Cases 2) Obsolete Test Cases.
- Re-usable Test cases can be used in succeeding regression cycles.
- Obsolete Test Cases can't be used in succeeding cycles.

## Prioritization of Test Cases

- Prioritize the test cases depending on business impact, critical & frequently used functionalities. Selection of test



cases based on priority will greatly reduce the regression test suite.

## Selecting test cases for regression testing

It was found from industry data that good number of the defects reported by customers were due to last minute bug fixes creating side effects and hence selecting the [Test Case](#) for regression testing is an art and not that easy. Effective Regression Tests can be done by selecting following test cases

-

- Test cases which have frequent defects
- Functionalities which are more visible to the users
- Test cases which verify core features of the product
- Test cases of Functionalities which has undergone more and recent changes
- All Integration Test Cases
- All Complex Test Cases
- Boundary value test cases
- Sample of Successful test cases
- Sample of Failure test cases

## Regression Testing Tools

If your software undergoes frequent changes, regression testing costs will escalate.

In such cases, Manual execution of test cases increases test execution time as well as costs.

Automation of regression test cases is the smart choice in such cases.

Extent of automation depends on the number of test cases that remain re-usable for successive regression cycles.

Following are most important tools used for both functional and regression testing:

**Selenium:** This is an open source tool used for automating web applications. Selenium can be used for browser based regression testing.

**Quick Test Professional (QTP):** HP Quick Test Professional is automated software designed to automate functional and regression test cases. It uses **VBScript** language for automation. It is a Data driven, Keyword based tool.

**Rational Functional Tester (RFT):** IBM's rational functional tester is a **Java** tool used to automate the test cases of software applications. This is primarily used for automating regression test cases and it also integrates with Rational Test Manager.

## Regression Testing and Configuration Management

Configuration Management during Regression Testing becomes imperative in Agile Environments where code is being continuously modified. To ensure effective regression tests, observe the following :

- Code being regression tested should be under a configuration management tool
- No changes must be allowed to code, during the regression test phase. Regression test code must be kept immune to developer changes.
- The database used for regression testing must be isolated. No database changes must be allowed

## Difference between Re-Testing and Regression Testing:

Retesting means testing the functionality or bug again to ensure

the code is fixed. If it is not fixed, [Defect](#) needs to be re-opened. If fixed, Defect is closed.

Regression testing means testing your software application when it undergoes a code change to ensure that the new code has not affected other parts of the software.

Also, Check out the complete list of differences over [here](#) .

## Challenges in Regression Testing:

Following are the major testing problems for doing regression testing:

- With successive regression runs, test suites become fairly large. Due to time and budget constraints, the entire regression test suite cannot be executed
- Minimizing test suite while achieving maximum [Test coverage](#) remains a challenge
- Determination of frequency of Regression Tests, i.e., after every modification or every build update or after a bunch of bug fixes, is a challenge.

### What is Adhoc Testing?

Adhoc testing is an informal testing type with an aim to break the system. This testing is usually an unplanned activity. It does not follow any test design techniques to create test cases. In fact it does not create test cases altogether! This testing is primarily performed if the knowledge of testers in the system under test is very high. Testers randomly test the application without any test cases or any business requirement document.

Ad hoc Testing does not follow any structured way of testing and it is randomly done on any part of application. Main aim of this testing is to find defects by random checking. Adhoc testing can be achieved with the testing technique called **Error Guessing**. Error guessing can be done by the people having enough experience on the system to "guess" the most likely source of errors.

This testing requires no documentation/ planning /process to be followed. Since this testing aims at finding defects through random approach, without any documentation, defects will not be mapped to test cases. Hence, sometimes, it is very difficult to reproduce the defects as there are no test steps or requirements mapped to it.

What is UAT?

User acceptance is a type of testing performed by the Client to certify the system with respect to the requirements that was agreed upon. This testing happens in the final phase of testing before moving the software application to Market or Production environment.

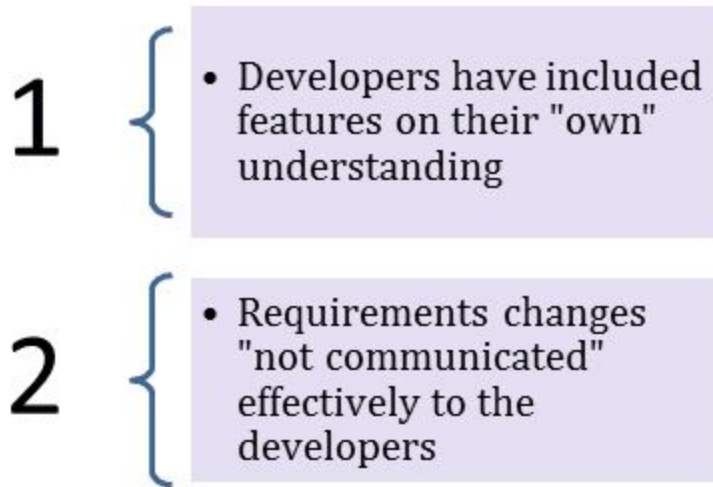
The main purpose of this testing is to validate the end to end business flow. It does NOT focus on the cosmetic errors, Spelling mistakes or System testing. This testing is carried out in separate testing environment with production like data setup. It is a kind of black box testing where two or more end users will be involved.

Who Performs UAT?

- Client
- End users

Need of User Acceptance Testing:

Once a software has undergone Unit, Integration and System testing the need of Acceptance Testing may seem redundant. **But Acceptance Testing is required because**



- Developers code software based on requirements document which is their "own" understanding of the requirements and **may not actually be what the client needs from the software.**
- Requirements changes during the course of the project may not be communicated effectively to the developers.

## Acceptance Testing and V-Model

In VModel, User acceptance testing corresponds to the requirement phase of the Software Development life cycle(SDLC).

software testing is the process of verifying and validating a software application to check whether it is working as expected. The intent is to find defects and improve the product quality. There are two ways to test a software, namely Positive Testing and Negative Testing.

## What is Positive Testing?

**Positive testing** is the type of testing that can be performed on the system by providing the **valid data as input**. It checks whether an application behaves as expected with positive inputs. This test is done to check the application that does what it is supposed to do.

For example -

Enter Only Numbers

### Positive Testing

There is a text box in an application which can accept only numbers. Entering values up to 99999 will be acceptable by the system and any other values apart from this should not be acceptable. To do positive testing, set the valid input values from 0 to 99999 and check whether the system is accepting the values.

### What is Negative Testing?

**Negative Testing** is a variant of testing that can be performed on the system by providing **invalid data as input**. It checks whether an application behaves as expected with the negative inputs. This is to test the application does not do anything that it is not supposed to do so.

For example -

Enter Only Numbers

### Negative Testing

Negative testing can be performed by entering characters A to Z or from a to z. Either software system should not accept the values or else it should throw an error message for these invalid data inputs.

### What is Integration Testing?

In integration Testing, individual software modules are integrated logically and tested as a group.

A typical software project consists of multiple software modules, coded by different programmers.

integration Testing focuses on checking data communication amongst these modules.

Hence it is also termed as '**I & T**' (Integration and Testing), '**String Testing**' and sometimes 'Thread Testing'.

## Why do Integration Testing?:

Although each software module is unit tested, defects still exist for various reasons like

- A Module in general is designed by an individual software developer whose understanding and programming logic may differ from other programmers. integration Testing becomes necessary to verify the software modules work in unity
- At the time of module development, there are wide chances of change in requirements by the clients. These new requirements may not be unit tested and hence system integration Testing becomes necessary.
- Interfaces of the software modules with the database could be erroneous
- External Hardware interfaces, if any, could be erroneous
- Inadequate exception handling could cause issues.

## What is White Box Testing?

White Box Testing is the testing of a software solution's internal coding and infrastructure. It focuses primarily on strengthening security, the flow of inputs and outputs through the application, and improving design and usability. White box testing is also known as Clear Box testing, Open Box testing, Structural testing, Transparent Box testing, Code-Based testing, and Glass Box testing.

It is one of two parts of the "**box testing**" **approach** of software testing. Its counter-part, blackbox testing, involves testing from an external or end-user type perspective. On the other hand, Whitebox testing is based on the inner workings of an application and revolves around internal testing.

The term "whitebox" was used because of the see-through box concept. The clear box or whitebox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "[Black Box Testing](#)" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested

Unit testing:

## What is Performance Testing?

Performance Testing is a type of testing to ensure software applications will perform well under their expected workload.

Features and Functionality supported by a software system is not the only concern. A software application's performance like its response time, reliability, resource usage and scalability do matter. The goal of Performance Testing is not to find bugs but to eliminate performance bottlenecks.

The focus of Performance Testing is checking a software program's

- Speed - Determines whether the application responds quickly
- Scalability - Determines maximum user load the software application can handle.
- Stability - Determines if the application is stable under varying loads

Performance Testing is popularly called as “Perf Testing” and is a subset of performance engineering.

## Why do Performance Testing?



Performance Testing is done to provide stakeholders with information about their application regarding speed, stability and scalability. More importantly, Performance Testing uncovers what needs to be improved before the product goes to market. Without Performance Testing, software is likely to suffer from issues such as: running slow while several users use it simultaneously, inconsistencies across different operating systems and poor usability. Performance testing will determine whether or not their software meets speed, scalability and stability requirements under expected workloads. Applications sent to market with poor performance metrics due to non-existent or poor performance testing are likely to gain a bad reputation and fail to meet expected sales goals. Also, mission-critical applications like space launch programs or life-saving medical equipments should be performance tested to ensure that they run for a long period of time without deviations.



# Types of Performance Testing

- **Load testing** - checks the application's ability to perform under anticipated user loads. The objective is to identify performance bottlenecks before the software application goes live.
- **Stress testing** - involves testing an application under extreme workloads to see how it handles high traffic or data processing. The objective is to identify breaking point of an application.
- **Endurance testing** - is done to make sure the software can handle the expected load over a long period of time.
- **Spike testing** - tests the software's reaction to sudden large spikes in the load generated by users.
- **Volume testing** - Under Volume Testing large no. of. Data is populated in database and the overall software system's behavior is monitored. The objective is to check software application's performance under varying database volumes.
- **Scalability testing** - The objective of scalability testing is to determine the software application's effectiveness in "scaling up" to support an increase in user load. It helps plan capacity addition to your software system.

Manual tester:

Webservices testing

Soapui.

Function testing

backend testing

Qtp

sql quiries

katalon

Seleinium

Performane testing

Jmeter

Loadrunner

Server knowledge

A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service. For example, a client invokes a web service by sending an XML message, then waits for a corresponding XML response. As all communication is in XML, web services are not tied to any one operating system or programming language--Java can talk with Perl; Windows applications can talk with Unix applications.

- Web services are self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains. These applications can be local, distributed, or web-based. Web services are built on top of open standards such as TCP/IP, HTTP, Java, HTML, and XML.
- Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents.
- A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards.

SOAP is an acronym for Simple Object Access Protocol. It is an XML-based messaging protocol for exchanging information among computers. SOAP is an application of the XML specification.

## Points to Note

Below mentioned are some important point which the user should take note of. These points briefly describes the nature of SOAP –

- SOAP is a communication protocol designed to communicate via Internet.
- SOAP can extend HTTP for XML messaging.
- SOAP provides data transport for Web services.
- SOAP can exchange complete documents or call a remote procedure.
- SOAP can be used for broadcasting a message.
- SOAP is platform- and language-independent.
- SOAP is the XML way of defining what information is sent and how.
- SOAP enables client applications to easily connect to remote services and invoke remote methods.

Although SOAP can be used in a variety of messaging systems and can be delivered via a variety of transport protocols, the initial focus of SOAP is remote procedure calls transported via HTTP.

Other frameworks including CORBA, DCOM, and Java RMI provide similar functionality to SOAP, but SOAP messages are written entirely in XML and are therefore uniquely platform- and language-independent.

A SOAP message is an ordinary XML document containing the following elements –

- **Envelope** – Defines the start and the end of the message. It is a mandatory element.
- **Header** – Contains any optional attributes of the message used in processing the message, either at an intermediary point or at the ultimate end-point. It is an optional element.
- **Body** – Contains the XML data comprising the message being sent. It is a mandatory element.
- **Fault** – An optional Fault element that provides information about errors that occur while processing the message.

No.	SOAP	REST
1)	SOAP is a <b>protocol</b> .	REST is an <b>architectural style</b> .
2)	SOAP stands for <b>Simple Object Access Protocol</b> .	REST stands for <b>REpresentational State Transfer</b> .
3)	SOAP <b>can't use REST</b> because it is a protocol.	REST <b>can use SOAP</b> web services because it is a concept and can use any protocol like HTTP, SOAP.
4)	SOAP <b>uses services interfaces to expose the business logic</b> .	REST <b>uses URI to expose business logic</b> .
5)	<b>JAX-WS</b> is the java API for SOAP web services.	<b>JAX-RS</b> is the java API for RESTful web services.

- |     |   |   |
|-----|---|---|
| 6)  | SOAP <b>defines standards</b> to be strictly followed.      | REST does not define too much standards like SOAP.                                    |
| 7)  | SOAP <b>requires more bandwidth</b> and resource than REST. | REST <b>requires less bandwidth</b> and resource than SOAP.                           |
| 8)  | SOAP <b>defines its own security</b> .                      | RESTful web services <b>inherits security measures</b> from the underlying transport. |
| 9)  | SOAP <b>permits XML</b> data format only.                   | REST <b>permits different</b> data format such as Plain text, HTML, XML, JSON etc.    |
| 10) | SOAP is <b>less preferred</b> than REST.                    | REST <b>more preferred</b> than SOAP.   |

**RESTful** Web Services are basically REST Architecture based Web Services. In REST Architecture everything is a resource. RESTful web services are light weight, highly scalable and maintainable and are very commonly used to create APIs for web-based applications. This tutorial will teach you the basics of RESTful Web Services and contains chapters discussing all the basic components of RESTful Web Services with suitable examples.

## What is REST?

REST stands for **RE**presentational State **T**ransfer. REST is a web standards based architecture and uses HTTP Protocol for data communication. It revolves around resources where every component is a resource and a resource is accessed by a common interface using HTTP standard methods. REST was first introduced by Roy Fielding in year 2000.

In REST architecture, a REST Server simply provides access to resources and the REST client accesses and presents the resources. Here each resource is identified by URIs/ Global IDs. REST uses various representations to represent a resource like Text, JSON and XML. JSON is now the most popular format being used in Web Services.

### HTTP Methods

The following HTTP methods are most commonly used in a REST based architecture.

- **GET** – Provides a read only access to a resource.
- **PUT** – Used to create a new resource.
- **DELETE** – Used to remove a resource.
- **POST** – Used to update an existing resource or create a new resource.
- **OPTIONS** – Used to get the supported operations on a resource.

## RESTful Web Services

A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and

running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards.

Web services based on REST Architecture are known as RESTful Web Services. These web services use HTTP methods to implement the concept of REST architecture. A RESTful web service usually defines a URI (Uniform Resource Identifier), which is a service that provides resource representation such as JSON and a set of HTTP Methods.

How you report defect?

Testers will use some tool to report defect:

ALM, Jira, Rally,

i)Description:

When entering valid id and invalid password for login , system is allowing to see the inbox page.

Steps to re-produce:

i)Goto yahoo.com

ii)Enter valid email

iii)enter invalid password

iv)Click on sign in button

Expected Result:

System should display error message.

Actual Result:

System allowing to user to sign in successfully.

Test data:

- vii)      khan
- viii)     password:there(invalid passwd)

Screen shot:

NEW

Open

Assign

Question

Re-assign

Fixed

Ready of re- test

Re-open

close

Test Cases:

Scenario:

Verify that when user enter valid email id and click on next button system will display password page.

Step #	Description/Input	Expected Result	Actual Result
1	Open following url in google chrome : "www.yahoo.com	Yahoo home page should display	pass
2	Click on sign in button	Login page should display	pass
3.	Click on email id field and enter valid email id	Email should enter successfully and will display in the field	
5.	Click next button	System should display page to enter password.	

1. What is Bytecode? What is JDK, JVM, JRE?

2. What is the role of ClassLoader in Java?
3. Can you save a java file without name?
4. What are wrapper classes and why do we need? Difference between int and Integer?
5. What is immutable classes? Name 5 immutable classes in Java? How to make a class immutable?
6. What are OOPS concepts? Explain them. What all OOPS concepts have you used in your framework?
7. What is Object class in Java? Explain role and importance of toString, equal and hashCode method?
8. What is method overloading and overriding? What is upcasting and down casting?
9. Can we overload private and final methods? Can we override static methods?
10. Can we override private and final methods?
11. Should return type be same in method overloading and overriding?
12. Can a overriding method throw new or broader checked exception?
13. Explain autoboxing in Java.
14. What is abstract class and interface? Difference between abstract class and interface? Can we have private methods in interfaces? Can we have constructor in abstract classes? Can we make a class abstract without any abstract methods?
15. What is constructor and its uses? What is the difference between super and this?
16. Will this code compile?

```
public class Vehicle {  
  
String model;  
  
String color;
```



```
public Vehicle(String model, String color) {  
  
    this.model=model;  
  
    this.color=color;  
  
}  
  
public static void main(String[] args) {  
  
    Vehicle v = new Vehicle();  
  
}  
  
}
```

17. What will be the output of below code?

```
class Vehicle {  
  
    public Vehicle() {  
  
        System.out.println("I am the super vehicle");  
  
    }  
  
}  
  
class FourWheeler extends Vehicle {  
  
    public FourWheeler() {  
  
        System.out.println("I am a car or a truck or whatever 4 wheel has");  
  
    }  
  
}  
  
class Car extends FourWheeler{  
  
    public Car() {  
  
        System.out.println("I am a car");  
  
    }  
  
}
```

```
}
```

```
}
```

```
public class Demo{
```

```
public static void main(String[] args) {
```

```
Car c = new Car();
```

```
}
```

```
}
```

18. What is the use of instanceof operator in java?
19. What are the access modifiers in java? Explain visibility of public, private, default and protractor?
20. What is the difference between Parent P = new Child() and Child C = new Child() where Parent is a super class and Child is a base class?
21. What is the use of final keyword in a variable, method and class?
22. Difference between final, finally and finalize?
23. What is the difference between == and equal()?
24. What will be the output of below code?

```
public class StringExample {
```

```
public static void main(String[] args) {
```

```
String s1 = "true";
```

```
String s2 = "true";
```

```
//first sop
```

```
System.out.println(s1==s2);
```

```
String s3= new String("true");
```

```
//second sop
```

```
System.out.println(s1==s3);
```

```
String s4= "True";
```

```
//Third sop
```

```
System.out.println(s2==s4);
```

```
}
```

```
}
```

25. What will be the output of below code?

```
public class StringExample {
```

```
public static void main(String[] args) {
```

```
String str = " hello ";
```

```
str.trim();
```

```
System.out.println(str);
```

```
}
```

```
}
```

26. How many methods of String class have you used? Name them with example

27. What is the difference between String, StringBuffer and StringBuilder?

28. Where are String values stored in memory? Why String class is immutable?

29. What is static block and instance block?

30. What is the default value of instance variable and local variable?

31. What is the use of static variable? Can we use static variables inside non static methods? Can we use non static variables inside static methods? How to invoke static methods?

32. Is it a valid for loop?

```
int i=0;

for(;i<10; i++){

    System.out.println(i);

}
```

33. What is the difference between break and continue?

34. What is the difference between while and do-while?

35. What will happen if we don't have break statement inside matching catch? Should default be the last statement in a switch case block?

36. Can your switch statement accept long, double or float data type?

37. What is the hierarchy of throwable class? What are checked and unchecked exceptions?

38. What is the difference between Error and Exception?

39. What is the difference between throw and throws?

40. Is try block without finally and catch allowed?

41. How to handle multi level exceptions?

42. Will this code compile?

```
public class ExcetionExample {

public static void main(String[] args) {

try {
```

```

FileReader f = new FileReader(new File("D:\\myfile"));

}catch(IOException e) {

}catch(FileNotFoundException e) {

}

}

}

```

43. What are the default values of array?

44. What is garbage collection?

45. How to run garbage collection? When will it run?

46. What are the ways to make objects eligible for garbage collection?

47. How many objects available for GC?

```

public class GC {

public static void main(String[] args) {

GC gc1= new GC();

GC gc2 = new GC();

GC gc3 = new GC();

gc1 = null;

}

}

```

48. Difference between System.gc() and Runtime.getRuntime().gc()?

49. How do you read contents of a file? How do you write in a file?

50. What is functional interface? Give example of functional interfaces in Java?
51. What are marker interfaces in Java?
52. What is multithreading? Difference between process and thread?
53. What is the difference between thread class and runnable interface?
54. How to get today's date using date class? How to add days in today's date?
55. How to get current month and year using calendar class?
56. What is stream in Java 8?
57. What is default and static methods in interface in Java 8?
58. What is the difference between Collections and Collection?
59. What is the hierarchy of collection?
60. What are the sub interfaces and sub classes of List, Set and Map interface?
61. What is the difference between ArrayList and LinkedList? Which one is faster in insertion, deletion and random access memory?
62. What is the difference between singly linkedlist, doubly linkedlist and circular linkedlist?
63. What main interfaces LinkedList implements?
64. What is the difference between Stack and Queue?
65. What is the difference between List and Set?
66. What is HashMap?
67. How to convert Array into ArrayList and ArrayList into Array?
68. How to convert Set into List?

69. How to iterate HashSet and HashMap?
70. What is SortedSet and SortedMap?
71. What is LinkedHashSet and LinkedHashMap?
72. What is TreeSet and TreeMap?
73. What is the difference between Hashtable and HashMap?
74. What are difference methods of Collections class have you used?
75. Can you iterate list forward and backward? Can you iterate linkedlist forward and backward?
76. What is the between comparable and comparator?
77. Why can we compare Strings without implementing compare() or compareTo methods?
78. WAP to check if a given number in palindrome.
79. WAP to count duplicates in a given string.
80. WAP to reverse a string using inbuilt reverse method as well as programmatically.
81. WAP to sort an array.
82. WAP to find min and max in array.
83. WAP to find sum of array.
84. WAP to find missing number in array.
85. WAP to find largest and second largest in array? Smallest and second smallest in array?
86. Where have you used List, Set and Map in Selenium?
87. What is Singleton class in Java? How to create Singleton class? What is factory method?
88. Example of Selenium methods using method overloading?

89. Can you name 10 interfaces in Selenium?
90. Have you ever designed framework? Please explain your framework.
91. What is WebDriver and WebElement?
92. What is the super interface of WebDriver? What is the hierarchy of WebDriver?
93. What are methods of WebDriver and WebElement?
94. What is the difference between findElement and findElements?
95. What are waits in selenium? Difference between implicit wait and explicit wait?
96. What is the importance of / and // in xpath?
97. What is the importance of \*, \$ and ^ in Css selector?
98. What are the challenges you faced in automation? What are the limitation of Selenium? What are the challenges you faced while working with IE browser?
99. Explain Page factory model.
100. How to read data from excel? Difference between HSSFWorkbook and XSSSSFWorkbook?
101. How to read data from dynamic web table? How to fetch data from last row of dynamic web table?
102. How to handle multiple windows in Selenium? How to open a new window? How do you switch from one window to another window?
103. How do you handle Alert? Can you write a method to check if alert exists?
104. How do you handle frames in Selenium?
105. Can you write isElementPresent method?
106. What are the various example of locator strategies in Selenium?



- 107.What is the difference between Xpath and CSS locators?
- 108.How to select values from dropdown?
- 109.What is the difference between Action and Actions?
- 110.How to do double click, right click, drag and drop?
- 111.What is robot class? Where do you use in Selenium?
- 112.What is the difference between quit and close?
- 113.Can you explain about some common exceptions in Selenium?
- 114.What is JavaScriptExecutor in Selenium?
- 115.How do you do parallel execution? What is threadlocal?
- 116.What are the various ways of click and sendkeys in Selenium?
- 117.What is Selenium grid and what are its usages?
- 118.What are the difference in Selenium WebDriver 2 and WebDriver 3?
- 119.How can you find if an element is displayed?
- 120.How can you find if an element is selected?
- 121.How can you find if an element is enabled?
- 122.How to refresh a page? How to go forward and back?
- 123.How to fetch title of a page?
- 124.What is the difference between getText() and getAttribute() methods?
- 125.What are the different types of frameworks?
- 125.What is BDD framework? Have you ever worked in any BDD framework?
- 126.How can we get the font size, font colour, font text used for the particular

text on the web page?

127.What is the difference between driver.get() and driver.navigate().to()?

128.What are desired capabilities? How to define desired capabilities for Chrome Firefox and IE?

129.What are before and after annotations in TESTNG? Explain their sequence.

130.What are include and exlude tag in testng.xml ?

140.How to define priorities of test methods in TESTNG?

141.What will happen if there are 3 @BeforeClass methods in a class?

142.What are listeners in TESTNG? Give example of some listners?

144.How to re-run failed test cases in TESTNG?

145.What is the difference between soft assert and assert in TestNG?

146.How to pass parameters from testing.xml?

147.How to read data from external files using data provider?

148.What is the return type of data provider?

149.How to maximize browser in selenium?

150.How to delete cookies in Selenium?

151.How to wait for page load in Selenium?

153.What are different assertions in TestNG?

154.How to mouse hover on an element?

155.What is Cucumber BDD?

157.Explain Cucumber workflow?

- 158.Can we extend Cucumber hooks classes?
- 159.Can we extend Cucumber step definition classes?
- 160.What are Cucumber hooks?
- 161.What are feature files in Cucumber?
- 162.What are step definitions in Cucumber?
- 163.How to create runner in Cucumber?
- 164.What are tags in Cucumber?
- 165.Explain options in Cucumber runner?
- 166.What are tagged hooks in cucumber?
- 167.What is the priority of before and after hooks in Cucumber?
- 168.What is background in cucumber?
- 167.What is data table in cucumber?
- 168.How to read data from data table?
- 169.What is the difference between scenario and scenario outlines?
- 170.What is example in cucumber?
- 171.Explain different components of feature file?
- 172.How to initialize Chrome, firefox and IE driver in Selenium?
- 173.What are the different languages supported by Webdriver?
- 174.What is geckodriver?
- 175.How to read values of all dropdown values?
- 176.How to capture screenshots in webdriver?

177.What is the difference between OnTestStart() and OnStart() methods in ItestListener?

178.How to click using JavaScriptExecutor?

179.What is threadcount and invocation count in TestNG?

180.What are the different browsers supporter by Selenium?

181.How can we specify dependency of one Test case to another in TestNG?

182.Explain what is POM.xml and dependencies.

183.How to trigger testing.xml using Maven?

184.What is thread life cycle in java?

185.What is the storage size of short, int, long, float, double.

189.What is the default value of int, long, float, double, boolean.

190.Can we catch more than one exception in single catch block?

191.What are inner classes in java?

192.What is the meaning of call by reference and call by value?

193.What is type conversion or type promotion in Java?

194.What is the scope of instance variables?

195.What is the scope of static variables?

196.Explain about static imports in Java?

197.Can you explain what about varargs? For example, main(String...args)

198.Can a class be named as main? Can a main method be overloaded or Overridden?

199.What is stack and heap memory? Where variables are created?

200.How do you run test cases on remote machine? How to create object of RemoteWebDriver class?