



Software testing tutorial provides basic and advanced concepts of software testing. Our software testing tutorial is designed for beginners and professionals.

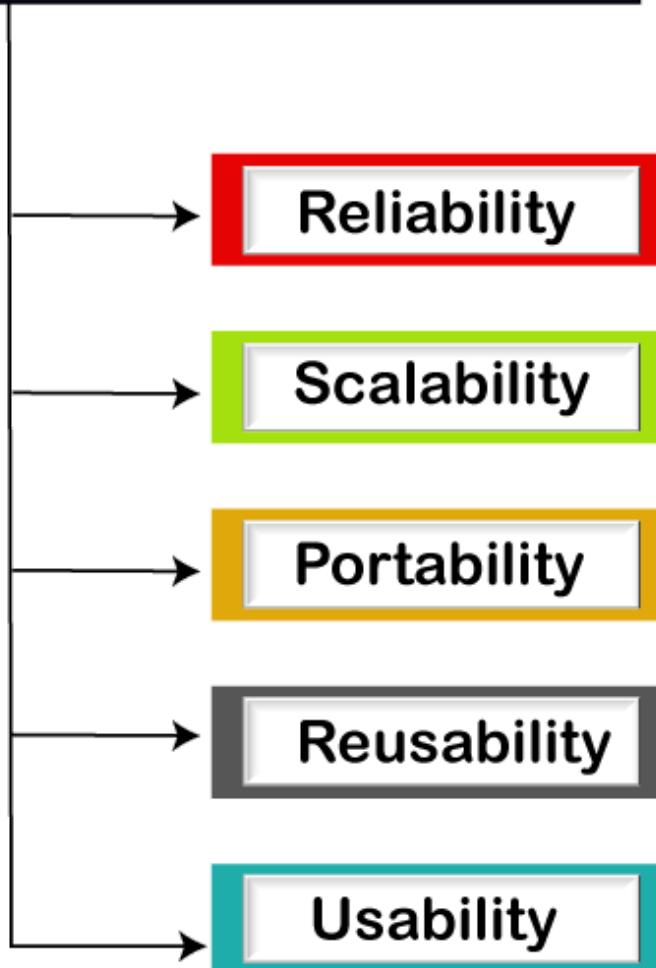
Software testing is widely used technology because it is compulsory to test each and every software before deployment.

Our Software testing tutorial includes all topics of Software testing such as Methods such as Black Box Testing, White Box Testing, Visual Box Testing and Gray Box Testing. Levels such as Unit Testing, Integration Testing, Regression Testing, Functional Testing. System Testing, Acceptance Testing, Alpha Testing, Beta Testing, Non-Functional testing, Security Testing, Portability Testing.

What is Software Testing

Software testing is a process of identifying the correctness of software by considering its all attributes (Reliability, Scalability, Portability, Re-usability, Usability) and evaluating the execution of software components to find the software bugs or errors or defects.

Attributes of Software



Software testing provides an independent view and objective of the software and gives surety of fitness of the software. It involves testing of all components under the required services to confirm that whether it is satisfying the specified requirements or not. The process is also providing the client with information about the quality of the software.

Testing is mandatory because it will be a dangerous situation if the software fails any of time due to lack of testing. So, without testing software cannot be deployed to the end user.

What is Testing

Testing is a group of techniques to determine the correctness of the application under the predefined script but, testing cannot find all the defect of application. The main intent of testing is to detect failures of the application so that failures can be discovered and corrected. It does not demonstrate that a product functions properly under all conditions but only that it is not working in some specific conditions.

Testing furnishes comparison that compares the behavior and state of software against mechanisms because the problem can be recognized by the mechanism. The mechanism may include past versions of the same specified product, comparable products, and interfaces of expected purpose, relevant standards, or other criteria but not limited up to these.

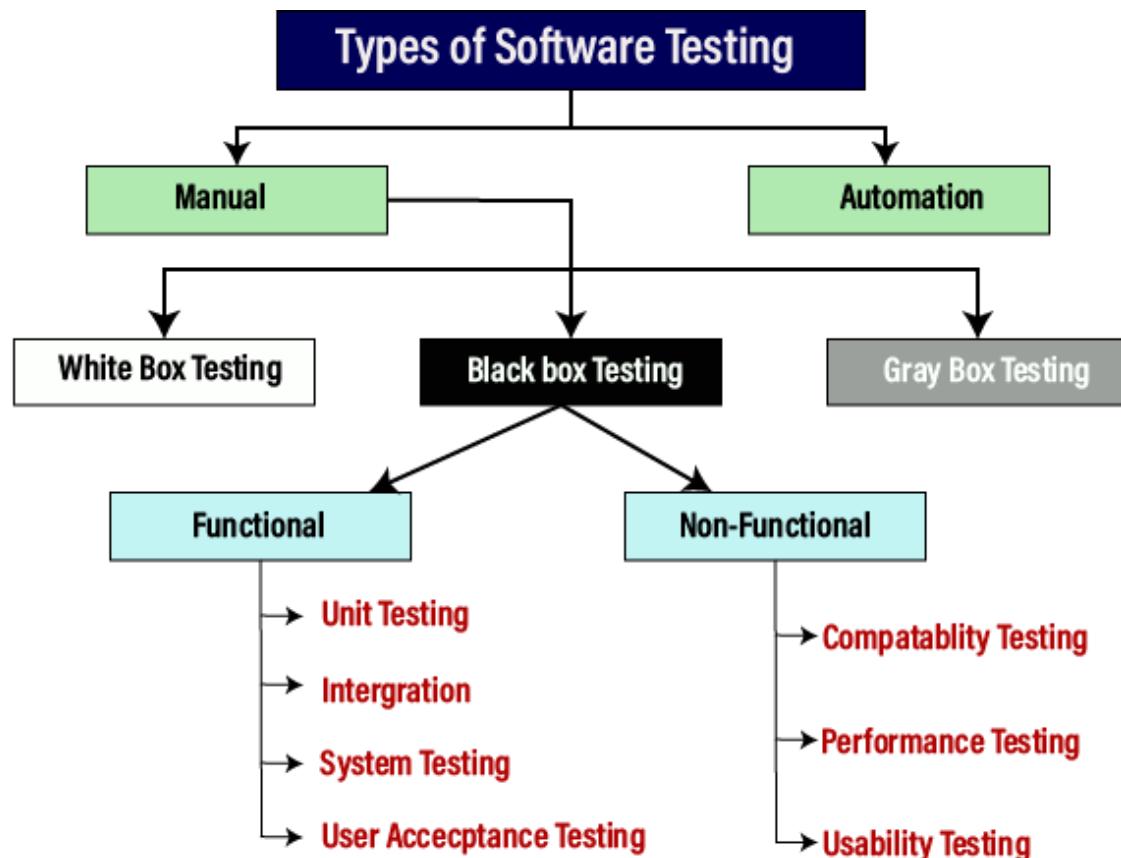
Testing includes an examination of code and also the execution of code in various environments, conditions as well as all the examining aspects of the code. In the current scenario of software development, a testing team may be separate from the development team so that Information derived from testing can be used to correct the process of software development.

The success of software depends upon acceptance of its targeted audience, easy graphical user interface, strong functionality load test, etc. For example, the audience of banking is totally different from the audience of a video game. Therefore, when an organization develops a software product, it can assess whether the software product will be beneficial to its purchasers and other audience.

Type of Software testing

We have various types of testing available in the market, which are used to test the application or the software.

With the help of below image, we can easily understand the type of software testing:



Manual testing

The process of checking the functionality of an application as per the customer needs without taking any help of automation tools is known as manual testing. While performing the manual testing on any application, we do not need any specific knowledge of any testing tool, rather than have a proper understanding of the product so we can easily prepare the test document.

Manual testing can be further divided into three types of testing, which are as follows:

- **White box testing**
- **Black box testing**
- **Gray box testing**

For more information about manual testing, refers to the below link:

<https://www.javatpoint.com/manual-testing>

Automation testing

Automation testing is a process of converting any manual test cases into the test scripts with the help of automation tools, or any programming language is known as automation testing. With the help of automation testing, we can enhance the speed of our test execution because here, we do not require any human efforts. We need to write a test script and execute those scripts.

For more information about manual testing, refers to the below link:

<https://www.javatpoint.com/automation-testing>

Prerequisite

Before learning software testing, you should have basic knowledge of basic computer functionality, basic mathematics, computer language, and logical operators.

Audience

Our software testing tutorial is designed for beginners and professionals.

Problems

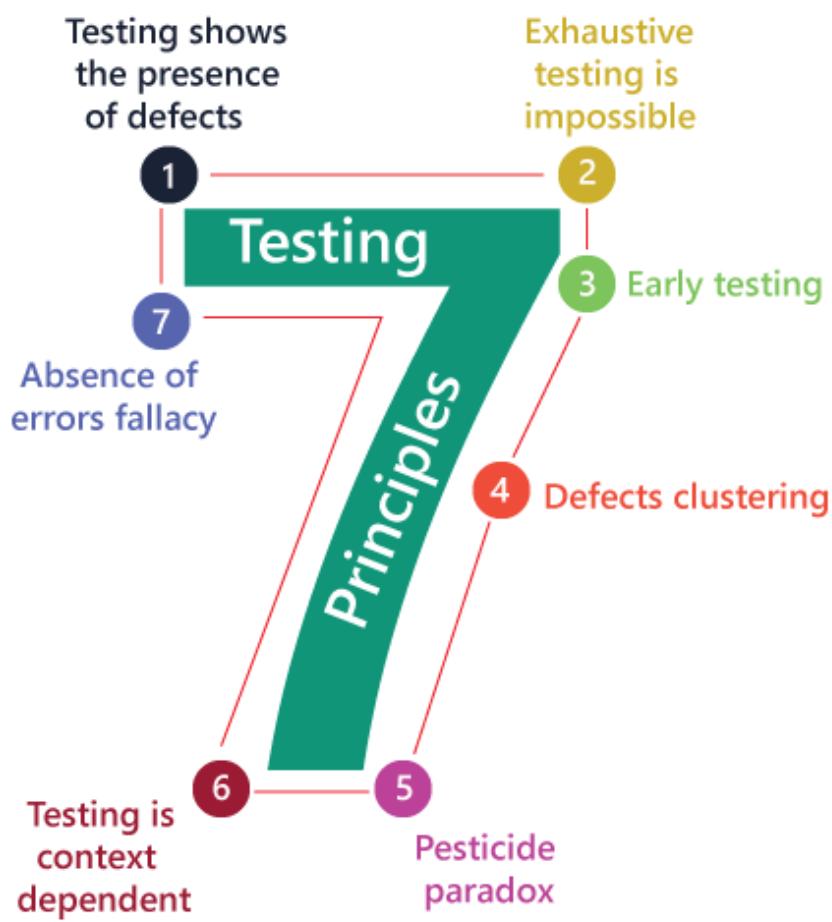
We assure that you will not find any problem in this Software Testing Tutorial. But if there is any mistake, please post the problem in contact form.

Software Testing Principles

Software testing is a procedure of implementing software or the application to identify the defects or bugs. For testing an application or software, we need to follow some principles to make our product defects free, and that also helps the test engineers to test the software with their effort and time. Here, in this section, we are going to learn about the seven essential principles of software testing.

Let us see the seven different testing principles, one by one:

- Testing shows the presence of defects
- Exhaustive Testing is not possible
- Early Testing
- Defect Clustering
- Pesticide Paradox
- Testing is context-dependent
- Absence of errors fallacy



Testing shows the presence of defects

The test engineer will test the application to make sure that the application is bug or defects free. While doing testing, we can only identify that the application or software has any errors. The primary purpose of doing testing is to identify the numbers of unknown bugs with the help of various methods and testing techniques because the entire test should be traceable to the customer requirement, which means that to find any defects that might cause the product failure to meet the client's needs.

By doing testing on any application, we can decrease the number of bugs, which does not mean that the application is defect-free because sometimes the software seems to be bug-free while performing multiple types of testing on it. But at the time of deployment in the production server, if the end-user encounters those bugs which are not found in the testing process.

Exhaustive Testing is not possible

Sometimes it seems to be very hard to test all the modules and their features with effective and non- effective combinations of the inputs data throughout the actual testing process.

Hence, instead of performing the exhaustive testing as it takes boundless determinations and most of the hard work is unsuccessful. So we can complete this type of variations according to the importance of the modules because the product timelines will not permit us to perform such type of testing scenarios.

Early Testing

Here early testing means that all the testing activities should start in the early stages of the software development life cycle's **requirement analysis stage** to identify the defects because if we find the bugs at an early stage, it will be fixed in the initial stage itself, which may cost us very less as compared to those which are identified in the future phase of the testing process.

To perform testing, we will require the requirement specification documents; therefore, if the requirements are defined incorrectly, then it can be fixed directly rather than fixing them in another stage, which could be the development phase.

Defect clustering

The defect clustering defined that throughout the testing process, we can detect the numbers of bugs which are correlated to a small number of modules. We have various reasons for this, such as the modules could be complicated; the coding part may be complex, and so on.

These types of software or the application will follow the **Pareto Principle**, which states that we can identify that approx. Eighty percent of the complication is present in 20 percent of the modules. With the help of this, we can find the uncertain modules, but this method has its difficulties if the same tests are performing regularly, hence the same test will not able to identify the new defects.

Pesticide paradox

This principle defined that if we are executing the same set of test cases again and again over a particular time, then these kinds of the test will not be able to find the new bugs in the software or the application. To get over these pesticide paradoxes, it is very significant to review all the test cases frequently. And the new and different tests are necessary to be written for the implementation of multiple parts of the application or the software, which helps us to find more bugs.

Testing is context-dependent

Testing is a context-dependent principle states that we have multiple fields such as e-commerce websites, commercial websites, and so on are available in the market. There is a definite way to test the commercial site as well as the e-commerce websites because every application has its own needs, features, and functionality. To check this type of application, we will take the help of various kinds of testing, different technique, approaches, and multiple methods. Therefore, the testing depends on the context of the application.

Absence of errors fallacy

Once the application is completely tested and there are no bugs identified before the release, so we can say that the application is 99 percent bug-free. But there is the chance when the application is tested beside the incorrect requirements, identified the flaws, and fixed them on a given period would not help as testing is done on the wrong specification, which does not apply to the client's requirements. The absence of error fallacy means identifying and fixing the bugs would not help if the application is impractical and not able to accomplish the client's requirements and needs.

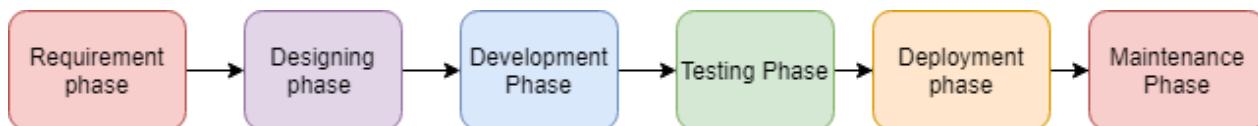
Software Development Life Cycle (SDLC)

SDLC is a process that creates a structure of development of software. There are different phases within SDLC, and each phase has its various activities. It makes the development team able to design, create, and deliver a high-quality product.

SDLC describes various phases of software development and the order of execution of phases. Each phase requires deliverable from the previous phase in a life cycle of software development. Requirements are translated into design, design into development and development into testing; after testing, it is given to the client.

Let's see all the phases in detail:

Different phases of the software development cycle



- Requirement Phase
- Design Phase
- Build /Development Phase
- Testing Phase
- Deployment/ Deliver Phase
- Maintenance

1. Requirement Phase

This is the most crucial phase of the software development life cycle for the developing team as well as for the project manager. During this phase, the client states requirements, specifications, expectations, and any other special requirement related to the product or software. All these are gathered by the business manager or project manager or analyst of the service providing company.

The requirement includes how the product will be used and who will use the product to determine the load of operations. All information gathered from this phase is critical to developing the product as per the customer requirements.

2. Design Phase

The design phase includes a detailed analysis of new software according to the requirement phase. This is the high priority phase in the development life cycle of a system because the logical designing of the system is converted into physical designing. The output of the requirement phase is a collection of things that are required, and the design phase gives the way to accomplish these requirements. The decision of all required essential tools such as **programming language** like Java, .NET, PHP, a **database** like Oracle, MySQL, a combination of hardware and software to provide a platform on which software can run without any problem is taken in this phase.

There are several techniques and tools, such as data flow diagrams, flowcharts, decision tables, and decision trees, Data dictionary, and the structured dictionary are used for describing the system design.

3. Build /Development Phase

After the successful completion of the requirement and design phase, the next step is to implement the design into the development of a software system. In this phase, work is divided into small units, and coding starts by the team of developers according to the design discussed in the previous phase and according to the requirements of the client discussed in requirement phase to produce the desired result.

Front-end developers develop easy and attractive GUI and necessary interfaces to interact with back-end operations and back-end developers do back-end coding according to the required operations. All is done according to the procedure and guidelines demonstrated by the project manager.

Since this is the coding phase, it takes the longest time and more focused approach for the developer in the software development life cycle.

4. Testing Phase

Testing is the last step of completing a software system. In this phase, after getting the developed GUI and back-end combination, it is tested against the requirements stated in the requirement phase. Testing determines whether the software is actually giving the result as per the requirements addressed in the requirement phase or not. The Development team makes a test plan to start the test. This test plan includes all types of essential testing such as integration testing, unit testing, acceptance testing, and system testing. Non-functional testing is also done in this phase.

If there are any defects in the software or it is not working as per expectations, then the testing team gives information to the development team in detail about the issue. If it is a valid defect or worth to sort out, it will be fixed, and the development team replaces it with the new one, and it also needs to be verified.

5. Deployment/ Deliver Phase

When software testing is completed with a satisfying result, and there are no remaining issues in the working of the software, it is delivered to the customer for their use.

As soon as customers receive the product, they are recommended first to do the beta testing. In beta testing, customer can require any changes which are not present in the software but mentioned in the requirement document or any other GUI changes to make it more user-friendly. Besides this, if any type of defect is encountered while a customer using the software; it will be informed to the development team of that particular software to sort out the problem. If it is a severe issue, then the development team solves it in a short time; otherwise, if it is less severe, then it will wait for the next version.

After the solution of all types of bugs and changes, the software finally deployed to the end-user.

6. Maintenance

The maintenance phase is the last and long-lasting phase of SDLC because it is the process which continues until the software's life cycle comes to an end. When a customer starts using software, then actual problems start to occur, and at that time there's a need to solve these problems. This phase also includes making changes in hardware and software to maintain its operational effectiveness like to improve its performance, enhance security features and according to customer's requirements with upcoming time. This process to take care of product time to time is called maintenance.

"So, all these are six phases of software development life cycle (SDLC) under which the process of development of software takes place. All are compulsory phases without any one of the development cannot be possible because development continues for the lifetime of software with maintenance phase".

Software Development Life Cycle (SDLC) Models

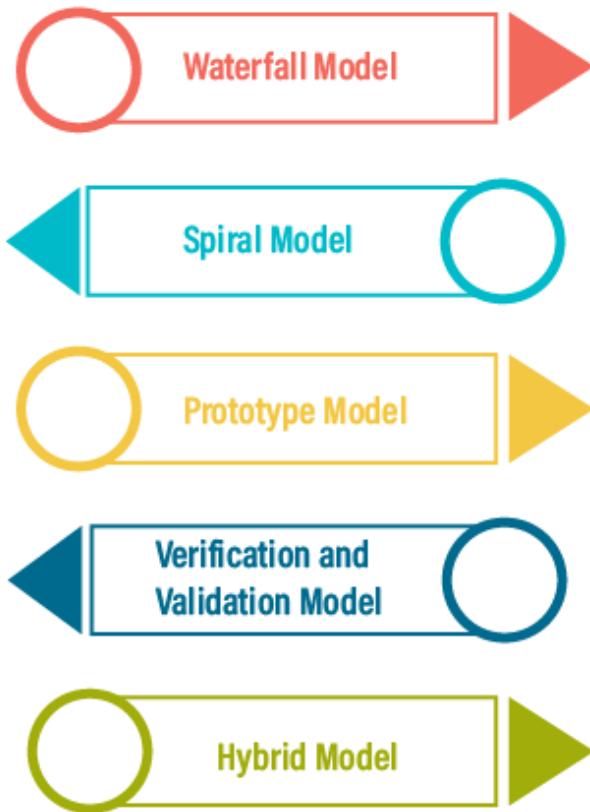
The software development models are those several process or approaches which are being selected for the development of project based on the project's objectives. To accomplish various purposes, we have many development life cycle models. And these models identify the multiple phases of the process. Picking up the correct model for developing the software application is very important because it will explain the what, where, and when of our planned testing.

Here, are various software development models or methodologies:

- **Waterfall model**
- **Spiral model**

- **Verification and validation model**
- **Prototype model**
- **Hybrid model**

Software Development life Cycle



Waterfall Model

It is the first sequential-linear model because the output of the one stage is the input of the next stage. It is simple and easy to understand, which is used for a small project. The various phases of the waterfall model are as follows:

- **Requirement analysis**
- **Feasibility study**
- **Design**
- **Coding**
- **Testing**
- **Installation**
- **Maintenance**

For information about the waterfall model, refers to the below link:

Spiral Model

It is the best suites model for a medium level project. It is also called the **Cyclic and Iteration** model. Whenever the modules are dependent on each other, we go for this model. And here, we develop application model wise and then handed over to the customer. The different stages of the spiral model are as follows:

- **Requirement collection**
- **Design**
- **Coding**
- **Testing**

For information about the spiral model, refers to the below link:

Prototype Model

From the time when customer rejection was more in the earlier model, we go for this model as customer rejection is less. And also, it allows us to prepare a sample (prototype) in the early stage of the process, which we can show to the client and get their approval and start working on the original project. This model refers to the action of creating the prototype of the application.

For information about the prototype model, refers to the below link:

Verification & Validation Model

It is an extended version of the waterfall model. It will implement in two phases wherein the first phase, we will perform the verification process, and when the application is ready, we will perform the validation process. In this model, the implementation happens in the V shape, which means that the verification process done under downward flow and the validation process complete in the upward flow.

For information about the Verification and validation model, refers to the below link:

Hybrid Model

The hybrid model is used when we need to acquire the properties of two models in the single model. This model is suitable for small, medium, and large projects because it is easy to apply, understand.

The combination of the two models could be as follows:

- **V and prototype**
- **Spiral and Prototype**

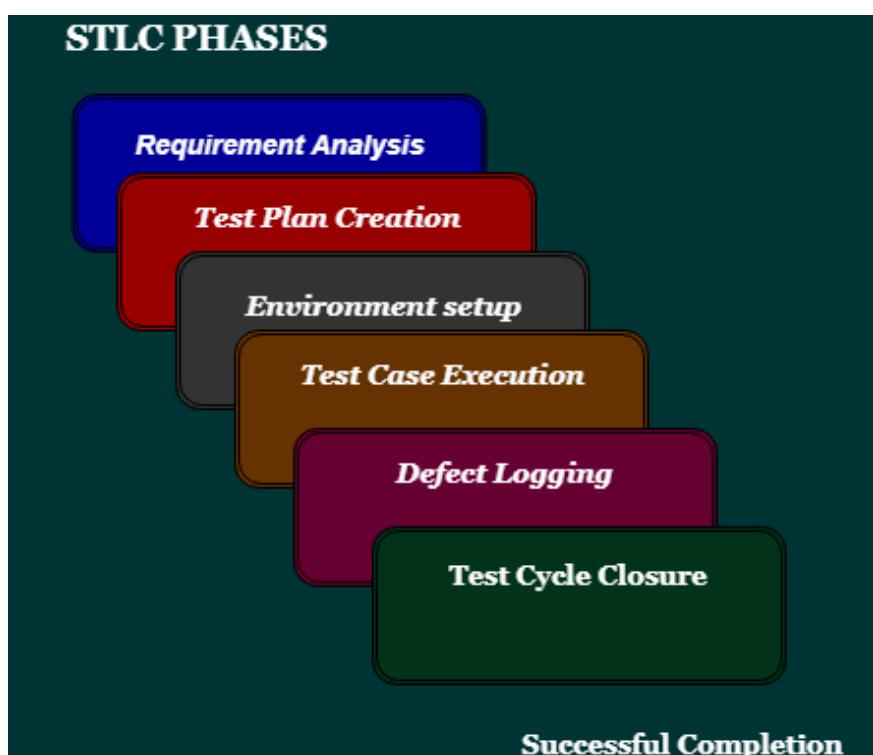
Software Testing Life Cycle (STLC)

The procedure of software testing is also known as STLC (Software Testing Life Cycle) which includes phases of the testing process. The testing process is executed in a well-planned and systematic manner. All activities are done to improve the quality of the software product.

Let's see, the different steps of STLC.

Software testing life cycle contains the following steps:

1. Requirement Analysis
2. Test Plan Creation
3. Environment setup
4. Test case Execution
5. Defect Logging
6. Test Cycle Closure



Requirement Analysis:

The first step of the manual testing procedure is requirement analysis. In this phase, tester analyses requirement document of SDLC (Software Development Life Cycle) to examine requirements stated by the client. After examining the requirements, the tester makes a test plan to check whether the software is meeting the requirements or not.

Entry Criteria	Activities	Deliverable
For the planning of test plan requirement specification, application architecture document and well-defined acceptance criteria should be available.	Prepare the list of all requirements and queries, and get resolved from Technical Manager/Lead, System Architecture, Business Analyst and Client. Make a list of all types of tests (Performance, Functional and security) to be performed. Make a list of test environment details, which should contain all the necessary tools to execute test cases.	List of all the necessary tests for the testable requirements and Test environment details

Test Plan Creation:

Test plan creation is the crucial phase of STLC where all the testing strategies are defined. Tester determines the estimated effort and cost of the entire project. This phase takes place after the successful completion of the **Requirement Analysis Phase**. Testing strategy and effort estimation documents provided by this phase. Test case execution can be started after the successful completion of Test Plan Creation.

Entry Criteria	Activities	Deliverable
Requirement Document	Define Objective as well as the scope of the software. List down methods involved in testing. Overview of the testing process. Settlement of testing environment. Preparation of the test schedules and control procedures. Determination of roles and responsibilities. List down testing deliverables, define risk if any.	Test strategy document. Testing Effort estimation documents are the deliverables of this phase.

Environment setup:

Setup of the test environment is an independent activity and can be started along with **Test Case Development**. This is an essential part of the manual testing procedure as without environment testing is not possible. Environment setup requires a group of essential software and hardware to create a test environment. The testing team is not involved in setting up the testing environment, its senior developers who create it.

Entry Criteria	Activities	Deliverable
Test strategy and test plan document. Test case document. Testing data.	Prepare the list of software and hardware by analyzing requirement specification. After the setup of the test environment, execute the smoke test cases to check the readiness of the test environment.	Execution report. Defect report.

Test case Execution:

Test case Execution takes place after the successful completion of test planning. In this phase, the testing team starts case development and execution activity. The testing team writes down the detailed test cases, also prepares the test data if required. The prepared test cases are reviewed by peer members of the team or Quality Assurance leader.

RTM (Requirement Traceability Matrix) is also prepared in this phase. Requirement Traceability Matrix is industry level format, used for tracking requirements. Each test case is mapped with the requirement specification. Backward & forward traceability can be done via RTM.

Entry Criteria	Activities	Deliverable
Requirement Document	Creation of test cases. Execution of test cases. Mapping of test cases according to requirements.	Test execution result. List of functions with the detailed explanation of defects.

Defect Logging:

Testers and developers evaluate the completion criteria of the software based on test coverage, quality, time consumption, cost, and critical business objectives. This phase determines the characteristics and drawbacks of the software. Test cases and bug reports are analyzed in depth to detect the type of defect and its severity.

Defect logging analysis mainly works to find out defect distribution depending upon severity and types. If any defect is detected, then the software is returned to the development team to fix the defect, then the software is re-tested on all aspects of the testing.

Once the test cycle is fully completed then test closure report, and test metrics are prepared.

Entry Criteria	Activities	Deliverable
Test case execution report. Defect report	It evaluates the completion criteria of the software based on test coverage, quality, time consumption, cost, and critical business objectives. Defect logging analysis finds out defect distribution by categorizing in types and severity.	Closure report Test metrics

Test Cycle Closure:

The test cycle closure report includes all the documentation related to software design, development, testing results, and defect reports.

This phase evaluates the strategy of development, testing procedure, possible defects in order to use these practices in the future if there is a software with the same specification.

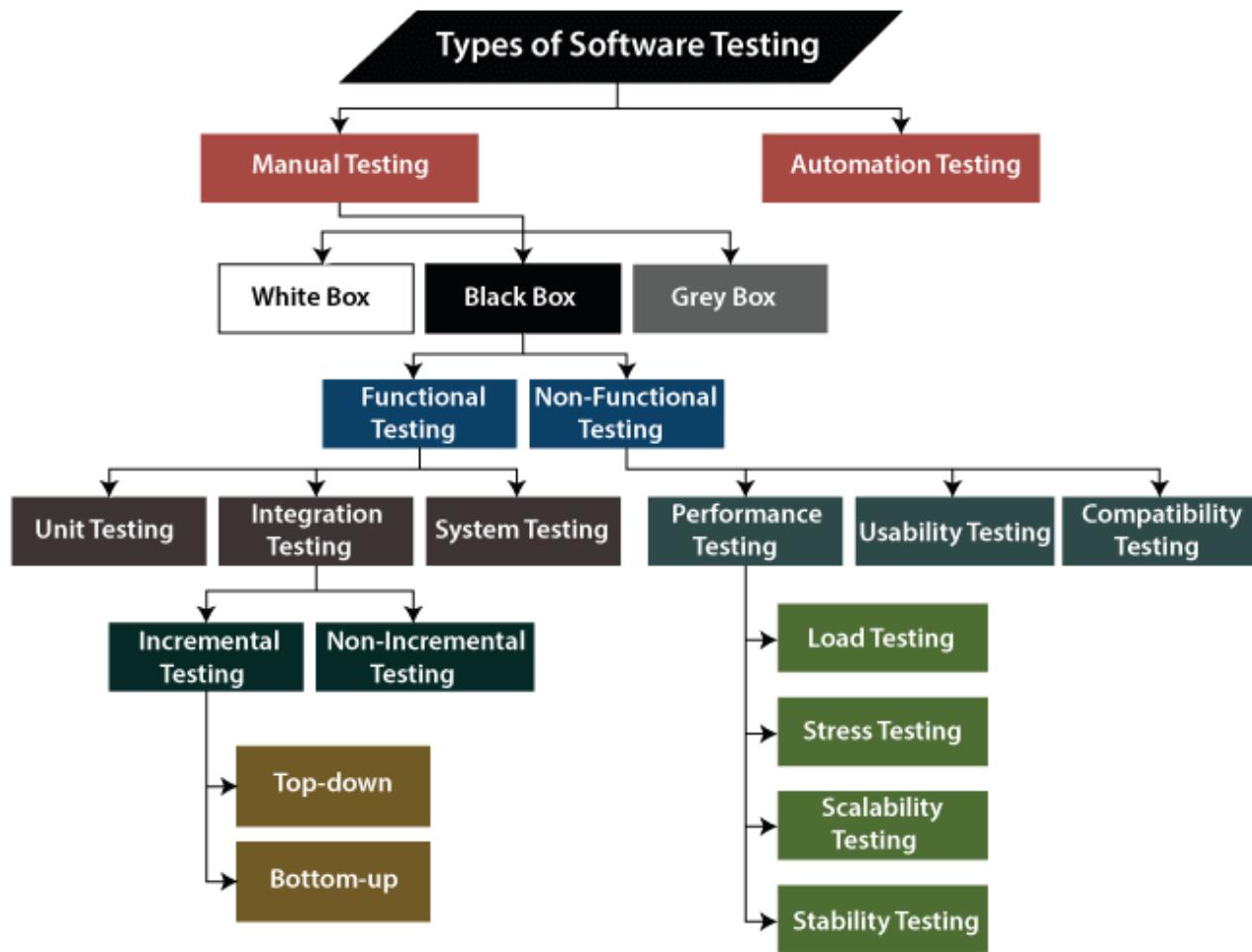
Entry Criteria	Activities	Deliverable
All document and reports related to software.	Evaluates the strategy of development, testing procedure, possible defects to use these practices in the future if there is a software with the same specification	Test closure report

Types of Software Testing

In this section, we are going to understand the various types of software testing, which can be used at the time of the Software Development Life Cycle.

As we know, **software testing** is a process of analyzing an application's functionality as per the customer prerequisite.

If we want to ensure that our software is bug-free or stable, we must perform the various types of software testing because testing is the only method that makes our application bug-free.



The different types of Software Testing

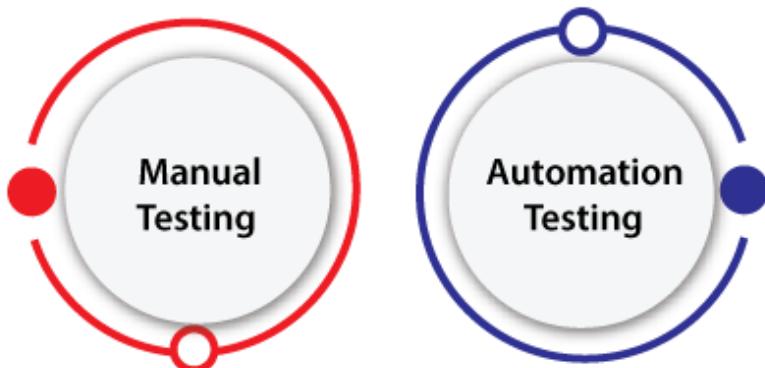
The categorization of software testing is a part of diverse testing activities, such as **test strategy, test deliverables, a defined test objective, etc.** And software testing is the execution of the software to find defects.

The purpose of having a testing type is to confirm the **AUT** (Application Under Test).

To start testing, we should have a **requirement, application-ready, necessary resources available**. To maintain accountability, we should assign a respective module to different test engineers.

The software testing mainly divided into two parts, which are as follows:

Types of Software Testing



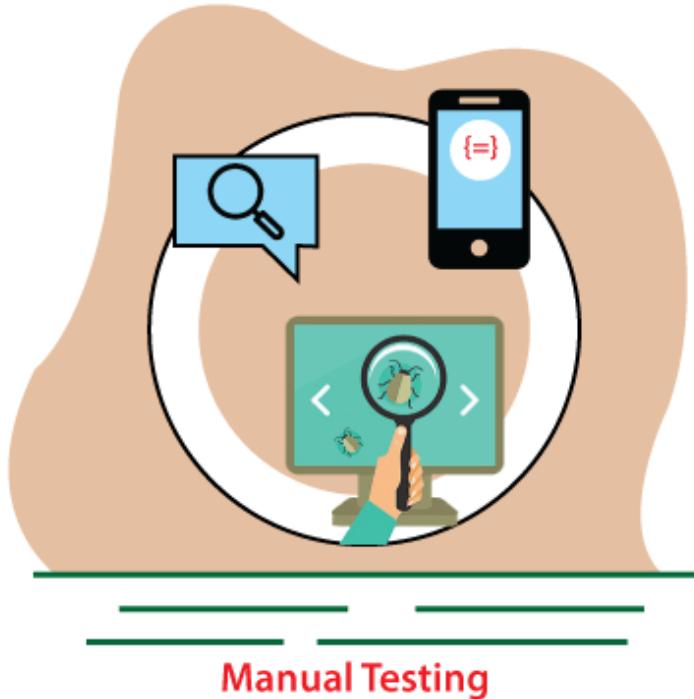
- **Manual Testing**

- **Automation Testing**

What is Manual Testing?

Testing any software or an application according to the client's needs without using any automation tool is known as **manual testing**.

In other words, we can say that it is a procedure of **verification and validation**. Manual testing is used to verify the behavior of an application or software in contradiction of requirements specification.



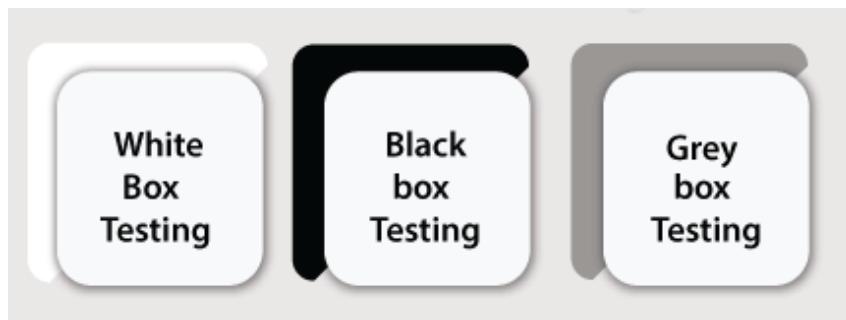
We do not require any precise knowledge of any testing tool to execute the manual test cases. We can easily prepare the test document while performing manual testing on any application.

To get in-detail information about manual testing, click on the following link: <https://www.javatpoint.com/manual-testing>.

Classification of Manual Testing

In software testing, manual testing can be further classified into **three different types of testing**, which are as follows:

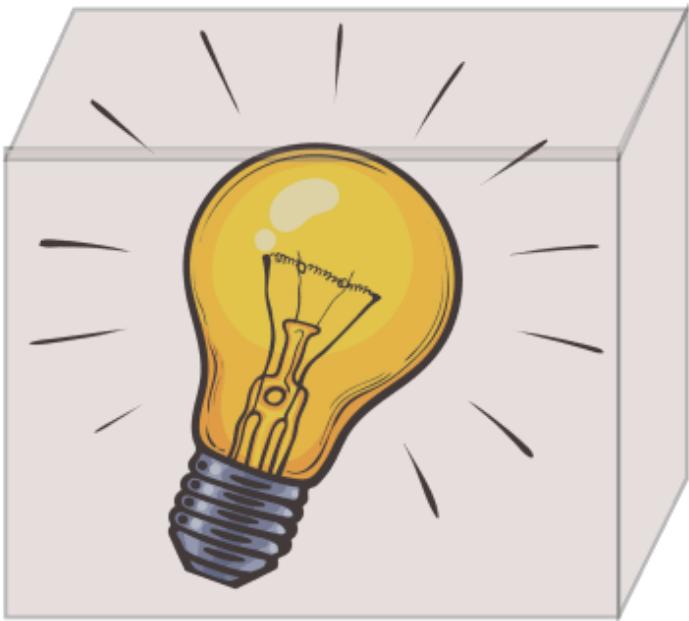
- **White Box Testing**
- **Black Box Testing**
- **Grey Box Testing**



For our better understanding let's see them one by one:

White Box Testing

In white-box testing, the developer will inspect every line of code before handing it over to the testing team or the concerned test engineers.

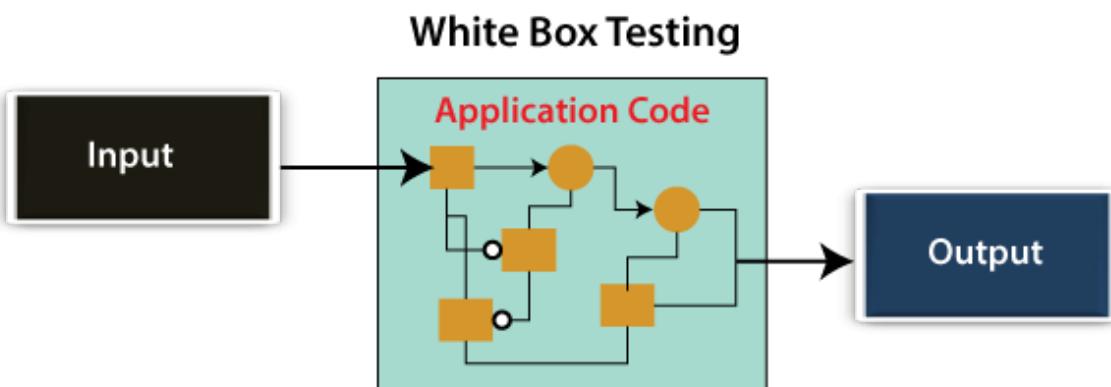


White Box Testing

Subsequently, the code is noticeable for developers throughout testing; that's why this process is known as **WBT (White Box Testing)**.

In other words, we can say that the **developer** will execute the complete white-box testing for the particular software and send the specific application to the testing team.

The purpose of implementing the white box testing is to emphasize the flow of inputs and outputs over the software and enhance the security of an application.



White box testing is also known as **open box testing, glass box testing, structural testing, clear box testing, and transparent box testing**.

To get the in-depth knowledge about white box testing refers to the below link: <https://www.javatpoint.com/white-box-testing>.

Black Box Testing

Another type of manual testing is **black-box testing**. In this testing, the test engineer will analyze the software against requirements, identify the defects or bug, and sends it back to the development team.



Black Box Testing

Then, the developers will fix those defects, do one round of White box testing, and send it to the testing team.

Here, fixing the bugs means the defect is resolved, and the particular feature is working according to the given requirement.

The main objective of implementing the black box testing is to specify the business needs or the customer's requirements.

In other words, we can say that black box testing is a process of checking the functionality of an application as per the customer requirement. The source code is not visible in this testing; that's why it is known as **black-box testing**.

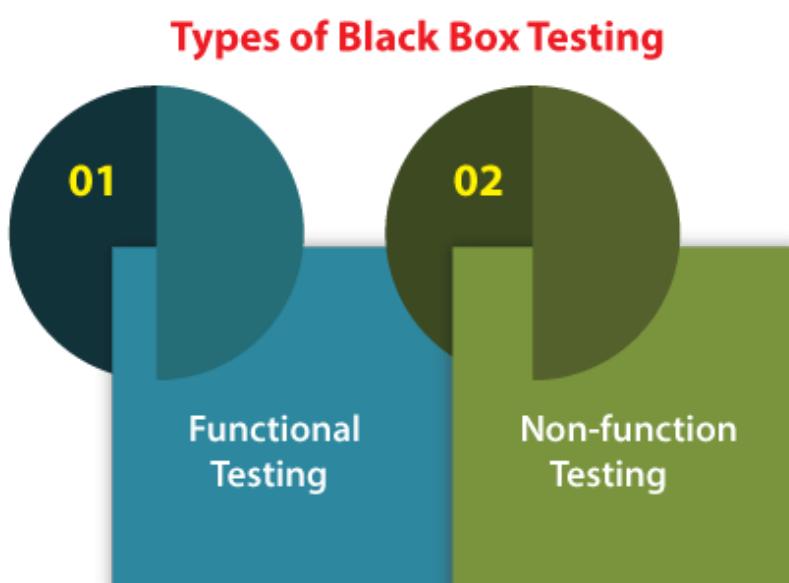


For more information about Black box testing, refers to the below link: <https://www.javatpoint.com/black-box-testing>.

Types of Black Box Testing

Black box testing further categorizes into two parts, which are as discussed below:

- **Functional Testing**
- **Non-function Testing**



Functional Testing

The test engineer will check all the components systematically against requirement specifications is known as **functional testing**. Functional testing is also known as **Component testing**.

In functional testing, all the components are tested by giving the value, defining the output, and validating the actual output with the expected value.

Functional testing is a part of black-box testing as its emphasizes on application requirement rather than actual code. The test engineer has to test only the program instead of the system.

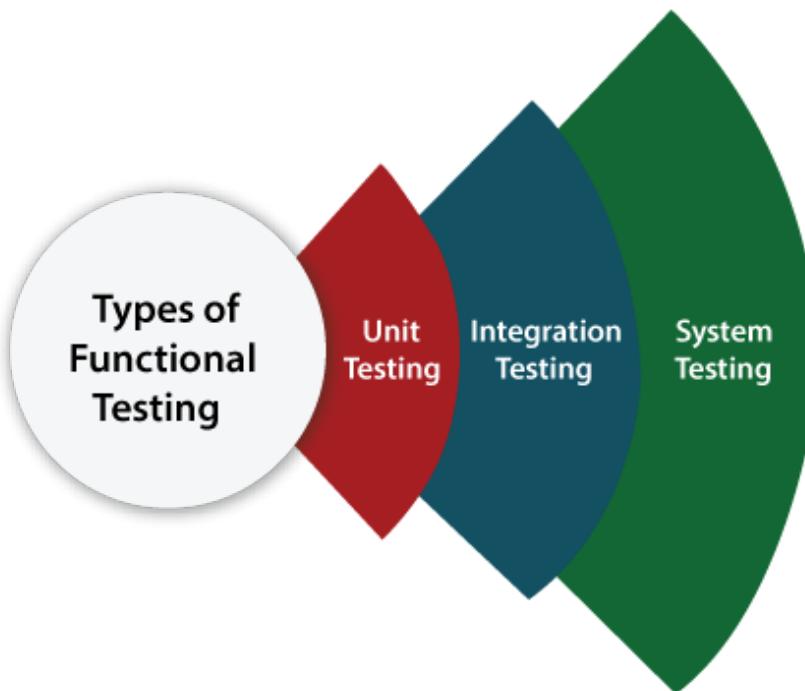
To get the detailed information about functional testing refers to the below link:
<https://www.javatpoint.com/functional-testing>.

Types of Functional Testing

Just like another type of testing is divided into several parts, functional testing is also classified into various categories.

The diverse **types of Functional Testing** contain the following:

- **Unit Testing**
- **Integration Testing**
- **System Testing**



Now, Let's understand them one by one:

1. Unit Testing

Unit testing is the first level of functional testing in order to test any software. In this, the test engineer will test the module of an application independently or test all the module functionality is called **unit testing**.

The primary objective of executing the unit testing is to confirm the unit components with their performance. Here, a unit is defined as a single testable function of a software or an application. And it is verified throughout the specified application development phase.

Click on the below link to get the complete information about unit testing: <https://www.javatpoint.com/unit-testing>.

2. Integration Testing

Once we are successfully implementing the unit testing, we will go **integration testing**. It is the second level of functional testing, where we test the data flow between dependent modules or interface between two features is called **integration testing**.

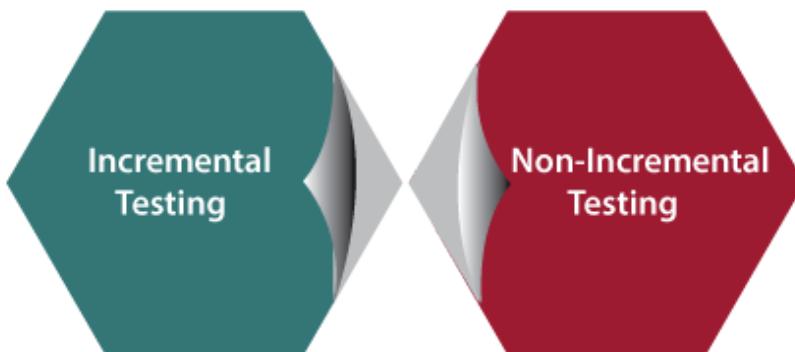
The purpose of executing the integration testing is to test the statement's accuracy between each module.

Types of Integration Testing

Integration testing is also further divided into the following parts:

- **Incremental Testing**
- **Non-Incremental Testing**

Types of Integration Testing



Incremental Integration Testing

Whenever there is a clear relationship between modules, we go for incremental integration testing. Suppose, we take two modules and analysis the data flow between them if they are working fine or not.

If these modules are working fine, then we can add one more module and test again. And we can continue with the same process to get better results.

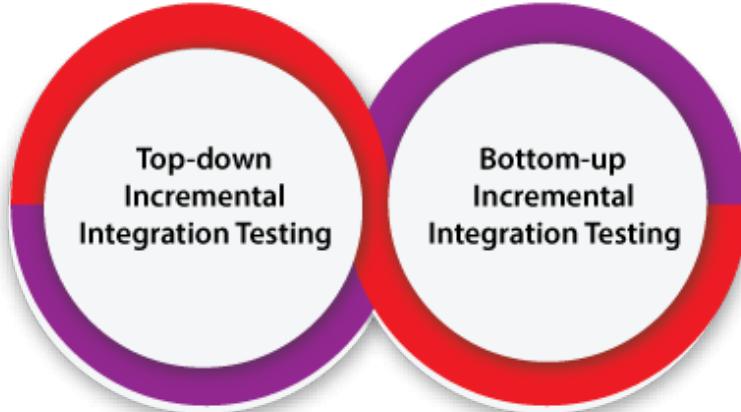
In other words, we can say that incrementally adding up the modules and test the data flow between the modules is known as **Incremental integration testing**.

Types of Incremental Integration Testing

Incremental integration testing can further classify into two parts, which are as follows:

1. **Top-down Incremental Integration Testing**
2. **Bottom-up Incremental Integration Testing**

Types of Incremental Integration Testing



Let's see a brief introduction of these types of integration testing:

1. Top-down Incremental Integration Testing

In this approach, we will add the modules step by step or incrementally and test the data flow between them. We have to ensure that the modules we are adding are the **child of the earlier ones**.

2. Bottom-up Incremental Integration Testing

In the bottom-up approach, we will add the modules incrementally and check the data flow between modules. And also, ensure that the module we are adding is the **parent of the earlier ones**.

Non-Incremental Integration Testing/ Big Bang Method

Whenever the data flow is complex and very difficult to classify a parent and a child, we will go for the non-incremental integration approach. The non-incremental method is also known as **the Big Bang method**.

To get the complete information about integration testing and its type refers to the following link: <https://www.javatpoint.com/integration-testing>.

3. System Testing

Whenever we are done with the unit and integration testing, we can proceed with the system testing.

In system testing, the test environment is parallel to the production environment. It is also known as **end-to-end** testing.

In this type of testing, we will undergo each attribute of the software and test if the end feature works according to the business requirement. And analysis the software product as a complete system.

Click on the below link to get the complete information about system testing: <https://www.javatpoint.com/system-testing>.

Non-functional Testing

The next part of black-box testing is **non-functional testing**. It provides detailed information on software product performance and used technologies.

Non-functional testing will help us minimize the risk of production and related costs of the software.

Non-functional testing is a combination of **performance, load, stress, usability and, compatibility testing**.

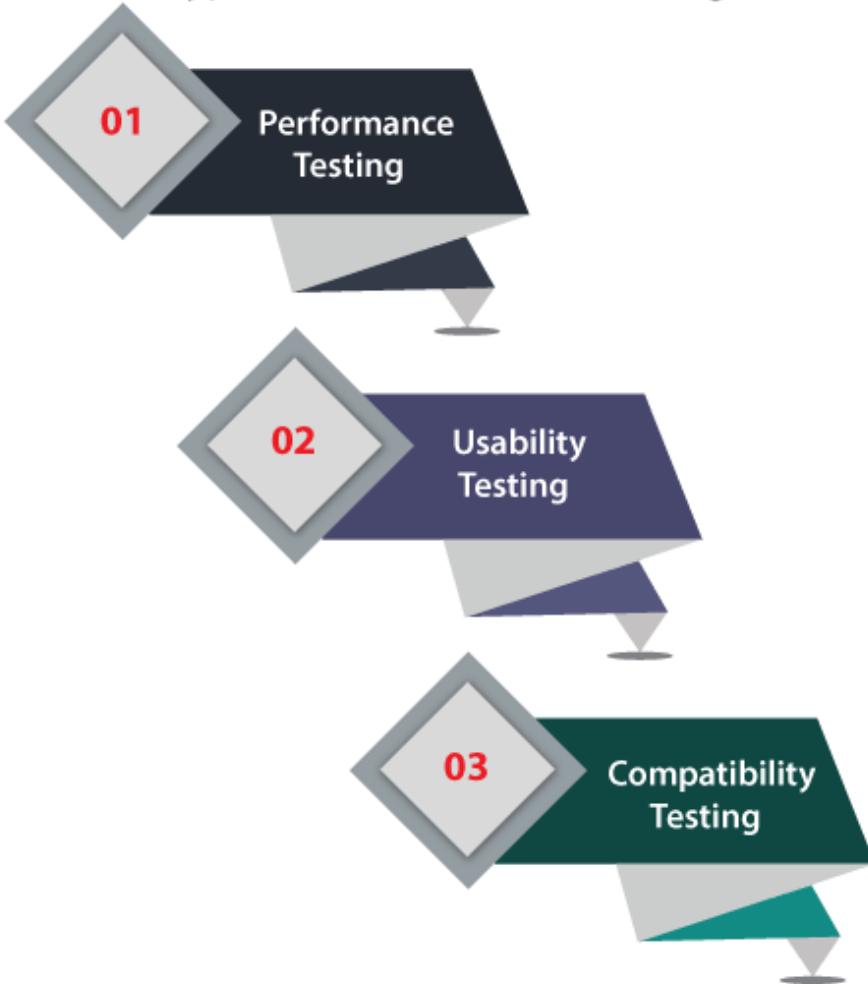
For more information about Non-functional testing, refer to the following link: <https://www.javatpoint.com/non-functional-testing>.

Types of Non-functional Testing

Non-functional testing categorized into different parts of testing, which we are going to discuss further:

- **Performance Testing**
- **Usability Testing**
- **Compatibility Testing**

Types of Non-functional Testing



1. Performance Testing

In performance testing, the test engineer will test the working of an application by applying some load.

In this type of non-functional testing, the test engineer will only focus on several aspects, such as **Response time**, **Load**, **scalability**, and **Stability** of the software or an application.

Classification of Performance Testing

Performance testing includes the various types of testing, which are as follows:

- **Load Testing**
- **Stress Testing**
- **Scalability Testing**
- **Stability Testing**

Classification of Performance Testing



- **Load Testing**

While executing the performance testing, we will apply some load on the particular application to check the application's performance, known as **load testing**. Here, the load could be less than or equal to the desired load.

It will help us to detect the highest operating volume of the software and bottlenecks.

To get the complete information related to the load testing refers to the below link:

<https://www.javatpoint.com/load-testing>.

- **Stress Testing**

It is used to analyze the user-friendliness and robustness of the software beyond the common functional limits.

Primarily, stress testing is used for critical software, but it can also be used for all types of software applications.

Refers to the below link for in-depth knowledge of stress testing: <https://www.javatpoint.com/stress-testing>.

- **Scalability Testing**

To analysis, the application's performance by enhancing or reducing the load in particular balances is known as **scalability testing**.

In scalability testing, we can also check the **system, processes, or database's ability** to meet an upward need. And in this, the **Test Cases** are designed and implemented efficiently.

Click on the following link to get the detailed information related to the scalability testing:

<https://www.javatpoint.com/scalability-testing>.

- **Stability Testing**

Stability testing is a procedure where we evaluate the application's performance by applying the load for a precise time.

It mainly checks the constancy problems of the application and the efficiency of a developed product. In this type of testing, we can rapidly find the system's defect even in a stressful situation.

To get detailed information about the stability testing refers to the below link:

<https://www.javatpoint.com/stability-testing>.

2. Usability Testing

Another type of **non-functional testing** is **usability testing**. In usability testing, we will analyze the user-friendliness of an application and detect the bugs in the software's end-user interface.

Here, the term **user-friendliness** defines the following aspects of an application:

- The application should be easy to understand, which means that all the features must be visible to end-users.
- The application's look and feel should be good that means the application should be pleasant looking and make a feel to the end-user to use it.

For more information about usability testing, we can refer to the following link:

<https://www.javatpoint.com/usability-testing>.

3. Compatibility Testing

In compatibility testing, we will check the functionality of an application in specific hardware and software environments. Once the application is functionally stable then only, we go for **compatibility testing**.

Here, **software** means we can test the application on the different operating systems and other browsers, and **hardware** means we can test the application on different sizes.

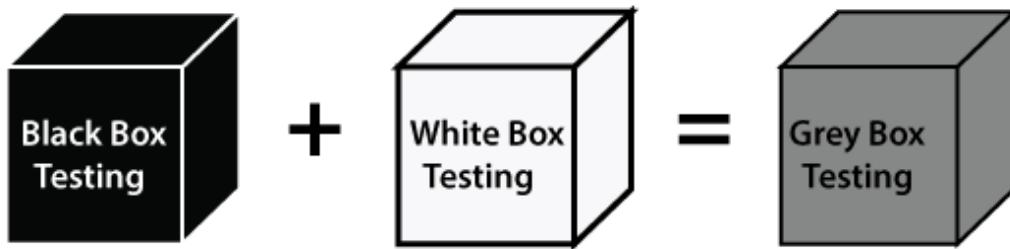
To get a thorough knowledge of compatibility testing refer to the below link:

<https://www.javatpoint.com/compatibility-testing>.

Grey Box Testing

Another part of **manual testing** is **Grey box testing**. It is a **collaboration of black box and white box testing**.

Since, the grey box testing includes access to internal coding for designing test cases. Grey box testing is performed by a person who knows coding as well as testing.



In other words, we can say that if a single-person team done both **white box and black-box testing**, it is considered **grey box testing**.

To get the in-detailes information about Grey box testing, we can refer to the below link:

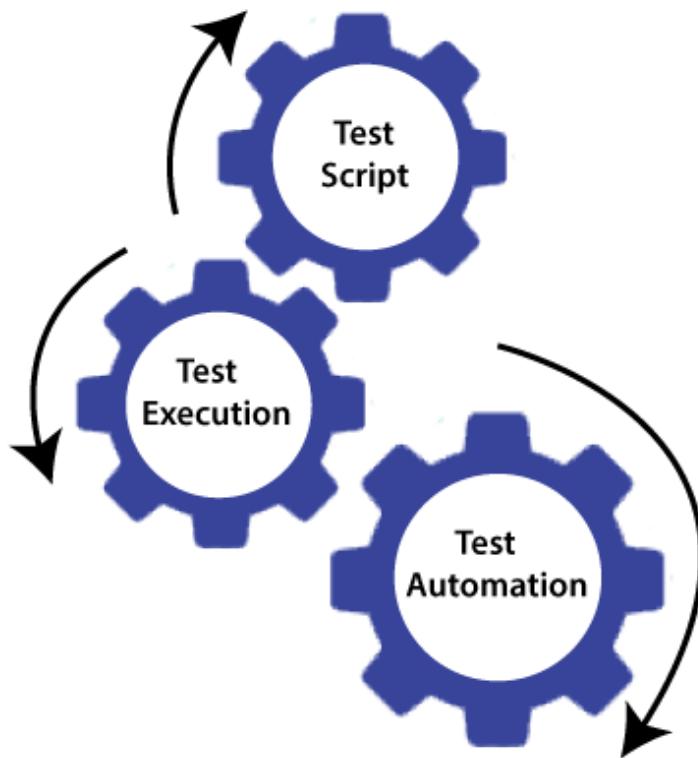
<https://www.javatpoint.com/grey-box-testing>.

Automation Testing

The most significant part of Software testing is Automation testing. It uses specific tools to automate manual design test cases without any human interference.

Automation testing is the best way to enhance the efficiency, productivity, and coverage of Software testing.

It is used to re-run the test scenarios, which were executed manually, quickly, and repeatedly.



In other words, we can say that whenever we are testing an application by using some tools is known as **automation testing**.

We will go for automation testing when various releases or several regression cycles goes on the application or software. We cannot write the test script or perform the automation testing without understanding the programming language.

For more information about automation testing, we can refer to the below link:

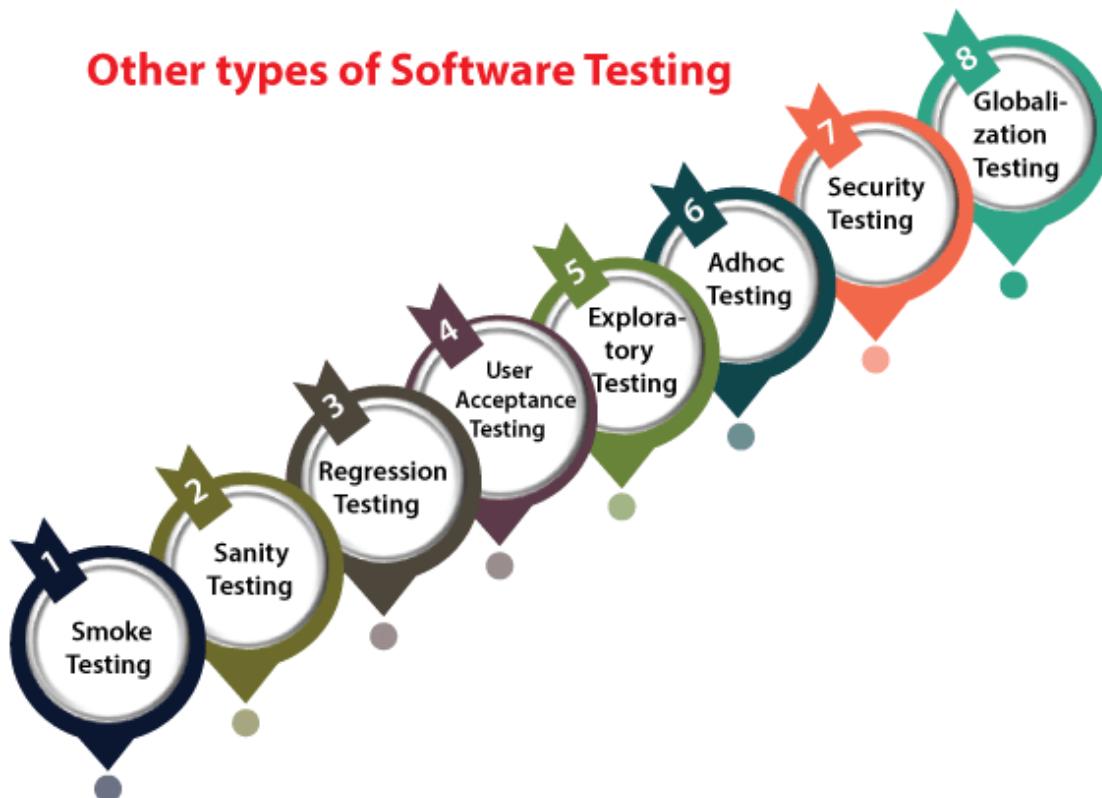
<https://www.javatpoint.com/automation-testing>.

Some other types of Software Testing

In software testing, we also have some other types of testing that are not part of any above discussed testing, but those testing are required while testing any software or an application.

- **Smoke Testing**
- **Sanity Testing**
- **Regression Testing**
- **User Acceptance Testing**
- **Exploratory Testing**
- **Adhoc Testing**
- **Security Testing**
- **Globalization Testing**

Let's understand those types of testing one by one:



In **smoke testing**, we will test an application's basic and critical features before doing one round of deep and rigorous testing.

Or before checking all possible positive and negative values is known as **smoke testing**. Analyzing the workflow of the application's core and main functions is the main objective of performing the smoke testing.

For more information about smoke testing, refers to the following link:

<https://www.javatpoint.com/smoke-testing>.

Sanity Testing

It is used to ensure that all the bugs have been fixed and no added issues come into existence due to these changes. Sanity testing is unscripted, which means we cannot document it. It checks the correctness of the newly added features and components.

To get the in-detail information about sanity testing, we can refer to the below link:

<https://www.javatpoint.com/sanity-testing>.

Regression Testing

Regression testing is the most commonly used type of software testing. Here, the term **regression** implies that we have to re-test those parts of an unaffected application.

Regression testing is the most suitable testing for automation tools. As per the project type and accessibility of resources, regression testing can be similar to **Retesting**.

Whenever a bug is fixed by the developers and then testing the other features of the applications that might be simulated because of the bug fixing is known as **regression testing**.

In other words, we can say that whenever there is a new release for some project, then we can perform Regression Testing, and due to a new feature may affect the old features in the earlier releases.

To get thorough knowledge related to regression testing, refer to the below link:

<https://www.javatpoint.com/regression-testing>.

User Acceptance Testing

The User acceptance testing (UAT) is done by the individual team known as domain expert/customer or the client. And knowing the application before accepting the final product is called as **user acceptance testing**.

In user acceptance testing, we analyze the business scenarios, and real-time scenarios on the distinct environment called the **UAT environment**. In this testing, we will test the application before UAI for customer approval.

For more information about the User acceptance testing, click on the below link:

<https://www.javatpoint.com/acceptance-testing>.

Exploratory Testing

Whenever the requirement is missing, early iteration is required, and the testing team has experienced testers when we have a critical application. New test engineer entered into the team then we go for the **exploratory testing**.

To execute the exploratory testing, we will first go through the application in all possible ways, make a test document, understand the flow of the application, and then test the application.

Click on the following link to get the complete information about exploratory testing:

<https://www.javatpoint.com/exploratory-testing>.

Adhoc Testing

Testing the application randomly as soon as the build is in the checked sequence is known as **Adhoc testing**.

It is also called **Monkey testing and Gorilla testing**. In Adhoc testing, we will check the application in contradiction of the client's requirements; that's why it is also known as **negative testing**.

When the end-user using the application casually, and he/she may detect a bug. Still, the specialized test engineer uses the software thoroughly, so he/she may not identify a similar detection.

Refers to the following to get the in-detail information about Adhoc testing:

<https://www.javatpoint.com/adhoc-testing>.

Security Testing

It is an essential part of software testing, used to determine the weakness, risks, or threats in the software application.

The execution of security testing will help us to avoid the nasty attack from outsiders and ensure our software applications' security.

In other words, we can say that security testing is mainly used to define that the data will be safe and endure the software's working process.

To get the complete detail about security testing, refer to the below link: <https://www.javatpoint.com/security-testing>.

Globalization Testing

Another type of software testing is **Globalization testing**. Globalization testing is used to check the developed software for multiple languages or not. Here, the words **globalization** means enlightening the application or software for various languages.

Globalization testing is used to make sure that the application will support multiple languages and multiple features.

In present scenarios, we can see the enhancement in several technologies as the applications are prepared to be used globally.

Refer to the following link to get the completed information related to the globalization testing:

<https://www.javatpoint.com/globalization-testing>.

Conclusion

In the tutorial, we have discussed various types of software testing. But there is still a list of more than 100+ categories of testing. However, each kind of testing is not used in all types of projects.

We have discussed the most commonly used types of Software Testing like **black-box testing, white box testing, functional testing, non-functional testing, regression testing, Adhoc testing, etc.**

Also, there are alternate classifications or processes used in diverse organizations, but the general concept is similar all over the place.

These testing types, processes, and execution approaches keep changing when the project, requirements, and scope change.

Levels of Testing

In this section, we are going to understand the various **levels of software testing**.

As we learned in the earlier section of the software testing tutorial that testing any application or software, the test engineer needs to follow multiple testing techniques.

In order to detect an error, we will implement software testing; therefore, all the errors can be removed to find a product with more excellent quality.

What are the levels of Software Testing?

Testing levels are the procedure for finding the missing areas and avoiding overlapping and repetition between the development life cycle stages. We have already seen the various phases such as **Requirement collection, designing, coding testing, deployment, and maintenance** of **SDLC (Software Development Life Cycle)**.

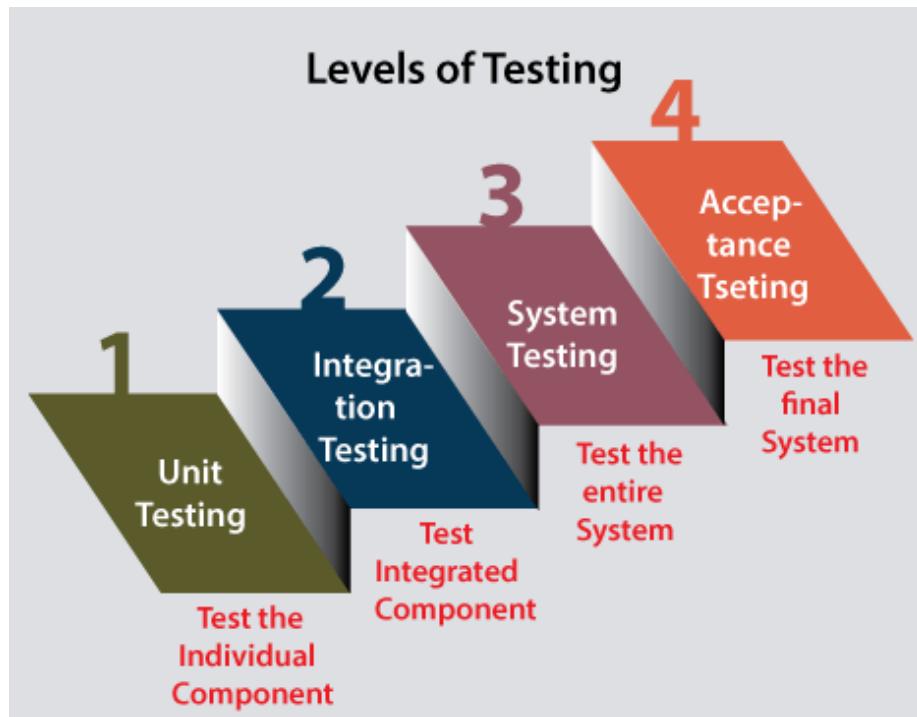
In order to test any application, we need to go through all the above phases of SDLC. Like SDLC, we have multiple levels of testing, which help us maintain the quality of the software.

Different Levels of Testing

The levels of software testing involve the different methodologies, which can be used while we are performing the software testing.

In **software testing**, we have four different levels of testing, which are as discussed below:

1. **Unit Testing**
2. **Integration Testing**
3. **System Testing**
4. **Acceptance Testing**



As we can see in the above image that all of these testing levels have a specific objective which specifies the value to the software development lifecycle.

For our better understanding, let's see them one by one:

Level1: Unit Testing

Unit testing is the first level of software testing, which is used to test if software modules are satisfying the given requirement or not.

The first level of testing involves **analyzing each unit or an individual component** of the software application.

Unit testing is also the first level of **functional testing**. The primary purpose of executing unit testing is to validate unit components with their performance.

A unit component is an individual function or regulation of the application, or we can say that it is the smallest testable part of the software. The reason of performing the unit testing is to test the correctness of inaccessible code.

Unit testing will help the test engineer and developers in order to understand the base of code that makes them able to change defect causing code quickly. The developers implement the unit.

For more information on unit testing, refers to the following link:

<https://www.javatpoint.com/unit-testing>.

Level2: Integration Testing

The second level of software testing is the **integration testing**. The integration testing process comes after **unit testing**.

It is mainly used to test the **data flow from one module or component to other modules**.

In integration testing, the **test engineer** tests the units or separate components or modules of the software in a group.

The primary purpose of executing the integration testing is to identify the defects at the interaction between integrated components or units.

When each component or module works separately, we need to check the data flow between the dependent modules, and this process is known as **integration testing**.

We only go for the integration testing when the functional testing has been completed successfully on each application module.

In simple words, we can say that **integration testing** aims to evaluate the accuracy of communication among all the modules.

For more information on integration testing, refers to the following link:

<https://www.javatpoint.com/integration-testing>.

Level3: System Testing

The third level of software testing is **system testing**, which is used to test the software's functional and non-functional requirements.

It is **end-to-end testing** where the testing environment is parallel to the production environment. In the third level of software testing, **we will test the application as a whole system**.

To check the end-to-end flow of an application or the software as a user is known as **System testing**.

In system testing, we will go through all the necessary modules of an application and test if the end features or the end business works fine, and test the product as a complete system.

In simple words, we can say that System testing is a sequence of different types of tests to implement and examine the entire working of an integrated software computer system against requirements.

For more information on System testing, refers to the following link:

<https://www.javatpoint.com/system-testing>.

Level4: Acceptance Testing

The **last and fourth level** of software testing is **acceptance testing**, which is used to evaluate whether a specification or the requirements are met as per its delivery.

The software has passed through three testing levels (**Unit Testing, Integration Testing, System Testing**). Some minor errors can still be identified when the end-user uses the system in the actual scenario.

In simple words, we can say that Acceptance testing is the **squeezing of all the testing processes that are previously done**.

The acceptance testing is also known as **User acceptance testing (UAT)** and is done by the customer before accepting the final product.

Usually, UAT is done by the domain expert (customer) for their satisfaction and checks whether the application is working according to given business scenarios and real-time scenarios.

For more information on System testing, refers to the following link:

<https://www.javatpoint.com/acceptance-testing>.

Conclusion

In this tutorial, we have learned all the levels of testing. And we can conclude that tests are grouped based on where they are added in the **Software development life cycle**.

A level of software testing is a process where every unit or component of a software or system is tested.

The main reason for implementing the **levels of testing** is to make the **software testing** process efficient and easy to find all possible test cases at a specific level.

To check the behavior or performance of software testing, we have various testing levels. The above-described software testing levels are developed to identify missing areas and understanding between the development life cycle conditions.

All these SDLC models' phases (**requirement gathering, analysis, design, coding or execution, testing, deployment, and maintenance**) undergo the process of software testing levels.

Test Maturity Model

In this section, we are going to explore and discuss the following topics related to **Test Maturity Model**, and we will also analyze the importance of TMM when it is applied to software testing process.

- **What is Test Maturity Model?**
- **The five levels of the test maturity model**
- **Difference between TMM and CMM**
- **Benefits of the test maturity model**

Before going deep into all the above mention topics, firstly, we will understand the **Test maturity Model**.

What is Test Maturity Model?

When software is tested, there are so many techniques are followed to accomplish maximum quality and minimize defects or errors.

Test Maturity Model is one of such models which has a set of structured levels and it is based on the **Capability Maturity Model (CMM)**.

The **Illinois Institute of Technology** initially developed the **test maturity model**, but now it is managed by **TMMI Foundation**.

We have used the Test maturity model to develop strategies and a reference framework for increasing the testing process.

Presently, the TMM is replaced by **Test Maturity Model Integration (TMMI)** which is five level model that provides a framework to measure the maturity of the testing processes.

We need to perform the testing phase of the **Software Development Life Cycle** very efficiently as it plays a significant role in order to complete any project successfully.

Test Maturity Model or TMMI is one such process that has made software testing life cycle more resourceful. It is one such model with a detailed model for test process improvement.

A Test maturity model's primary purpose is to find the maturity and provide targets to enhance the **software testing** process in order to accomplish development. It can be used as a stand-alone model or completed with any process improvement model.

Why we need TMMI?

The **Test Maturity Model Integration/TMMI** is progressively discover its way into many IT organizations to update and ease their testing process.

But the question arises, why do we need the TMMI model?

Below are a few significant points, which are helpful for us to understand the need for TMMI.

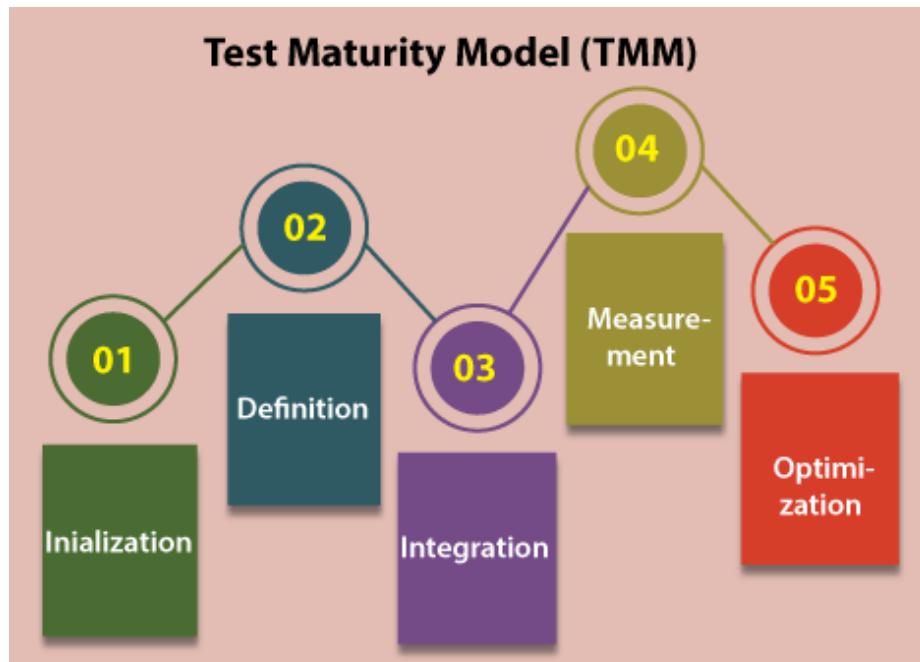
- The **Test Maturity Model Integration (TMMI)** helps in the assessment and enhancement of the testing process.
- The implementation of this model improves the test process, the software quality, and the efficiency of test engineering.
- Several software process developments such as **CMM/CMMI** provide limited consideration to testing. Hence, we need something like TMMI for a process, which is mainly made for testing.
- However, many efforts were made to expand the testing process; still, zero defects are identified from reality for the software engineering. So, TMMI is the further attempt to achieve zero defects.
- It is a test process improvement model that can integrate with other process development models and be used as a standalone model.

Levels of Test Maturity Model

The test Maturity model contains **five different levels**, which will help the organization fix its maturity. To achieving a higher level of test maturity, these five levels will lead us to identify the next development steps:

Let's understand the five levels of TMM one by one in detail:

1. Initialization
2. Definition
3. Integration
4. Measurement and Management
5. Optimization



Level1: Initialization

- **Initialization** is the first level of the Test maturity model. There is no defined testing process in the first level of TMM.
- The purpose behind initialization levels is to ensure that the software should execute successfully and there is no obstruction.
- At this level, there will be Exploratory or Adhoc testing performed on the software, and there are no quality checks before delivering the product.

Level2: Definition

- The second level of a Test maturity model is **Definition**, which is all about defining the requirements.
- We can create the test strategies, test plans, and test cases in order to build a software according to the given requirements by the client,
- The critical purpose of the definition level is to ensure that the software product implements according to the requirements, develop testing, debugging goals, and policies that are followed consistently.

Level3: Integration

- The third level of a test maturity model is
- The primary purpose of executing this level into the test maturity model is to ensure that the testing is integrated with the software lifecycle and becomes a part of it.
- **For instance**, as we know that the V model has both **development and testing** phases, which means that the testing comes after when the development process is completed.

- The entire testing objectives are based on risk management as testing is implemented independently.

Level4: Measurement and Management

- The fourth level of a Test maturity model is **measurement** and **management** where testing becomes part of all the activities in the software life cycle.
- Here, we will be managing and measuring the requirements.
- The primary purpose of executing this level into the **test maturity model** is to ensure that the establishment of a test measurement program.
- To determine the quality measures this level, include **reviewing, requirements gathering, and design of the software.**

Level5: Optimization

- The **last and fifth level** of the Test Maturity level is **optimization**.
- The essential purpose of this level is to optimize the test process itself.
- In simple words, we can say that the testing processes are verified, and measures are taken to enhance the further process.
- In this, quality control and bug inhibition are performed during the software life cycle.
- At the optimization level, we mainly focus on defect prevention rather than defect detection and with the help of the different tools, we can perform this testing.

After seeing all the **five levels of the test maturity model**, we can say that each level has its role and responsibility. And the objective of all the levels has to create its well-defined structure.

The primary concept of the **Test Maturity Model** was taken from Capability Maturity Model (CMM).

Essentially it is a structured tool that is used for software development and a model to support different business processes. In **Test Maturity Model**, the term **Maturity** is measured by the degree of optimized processes.

Difference between CMM & TMM

Some of the significant difference between **CMM** and **TMM** models are as discussed in the following table:



S.No.	CMM	TMM
1.	The capability Maturity Model or CMM is used to consider the maturity of an organization's software processes.	Test Maturity Model or TMM specify testing and is related to checking the quality of the software testing model.
2.	It has significantly controlled the software development procedures.	It is used as a corresponding framework along with CMMi.

3.	CMMi mainly focuses on software development practices.	The complete focuses of the TMMi framework are on the processes which are applied to software testing to enhance the quality and efficiency of the testing process.
----	--	---

Advantages of Test Maturity Model

Let's see some of the significant advantages of using the Test Maturity Model (TMM) in an organization:

Defect prevention

- As we know from the above explanation, TMM emphasizes defect prevention rather than bug detection by making the testing process a part of all phases of the **software development life cycle**.
- It also makes sure that the maximum defects are identified and the mostly final product is defect-free.

Organized

- As we have already discussed the five levels of TMM and we can conclude that each level is well defined and has a particular purpose to achieve, which makes the test maturity model a well-organized model with solid objectives.

Clear requirements

- When the necessities of the software, designs are reviewed, test plans, and test cases are tested in contradiction of requirements. Or if the primary test aim is more precise, then we can achieve more accurate testing.

Assurance of quality

- We can achieve a higher quality of the product if we integrate testing with all the phases of the software life cycle.
- Analysis of test processes would enhance the outcome, which assures a good quality product.

Overview

As compared to **the CMM (Compatibility Maturity Model)**, **the TMM (Test Maturity Model)** is equally a new topic, but the primary purpose of both the models continues the same.

If we want to display how a structured set of levels leads to the high-quality expected output by enhancing the processes, and performance of an organization.

For any organization, software maintenance is an expensive and time-consuming process when bugs are identified after project delivery.

Therefore, while identifying the defects is significant, it is also necessary that software makes minor errors throughout the development phase. A standard testing process like **TMM** can help us in order to accomplish this.

As we understood from the above discussion, the **Test Maturity Model** is specially designed to **address testing**. And to help the organization in order to enhance the maturity of their testing exercises.

To make sure the enhancement in testing processes in an IT organization, the TMMi model was developed. And according to the business, these models can be generalized and applied for improved results.

It is introduced because the previous model did not focus on the testing processes.

Still, the TMMi model is developed to focus on planning and development, and the CMMi model procedures guide it.

Waterfall model

It is the first approach and the basic model used in software development. It is a simple model that is easy to use as well as understand. The execution happens in the sequence order, which means that the outcome of the one-stage is equal to the input of another stage. That's why it is also known as the Linear-sequential life cycle model.

To avoid the overlapping issues of the multiple phases, every stage should be completed before moving to the next stage. Each stage of the waterfall model involves the deliverable of the previous stage, like requirements, are transferred to the design phase, design moved to development, and so on. When we have the Life critical (hospital application) and Machine critical (Military project), we will widely use the waterfall model.

The waterfall model is divided into various stages, which are as follows:

- **Requirement collection**
- **Feasibility study**
- **Design**
- **Coding**
- **Testing**
- **Installation**
- **Maintenance**

Let us understand them one by one:

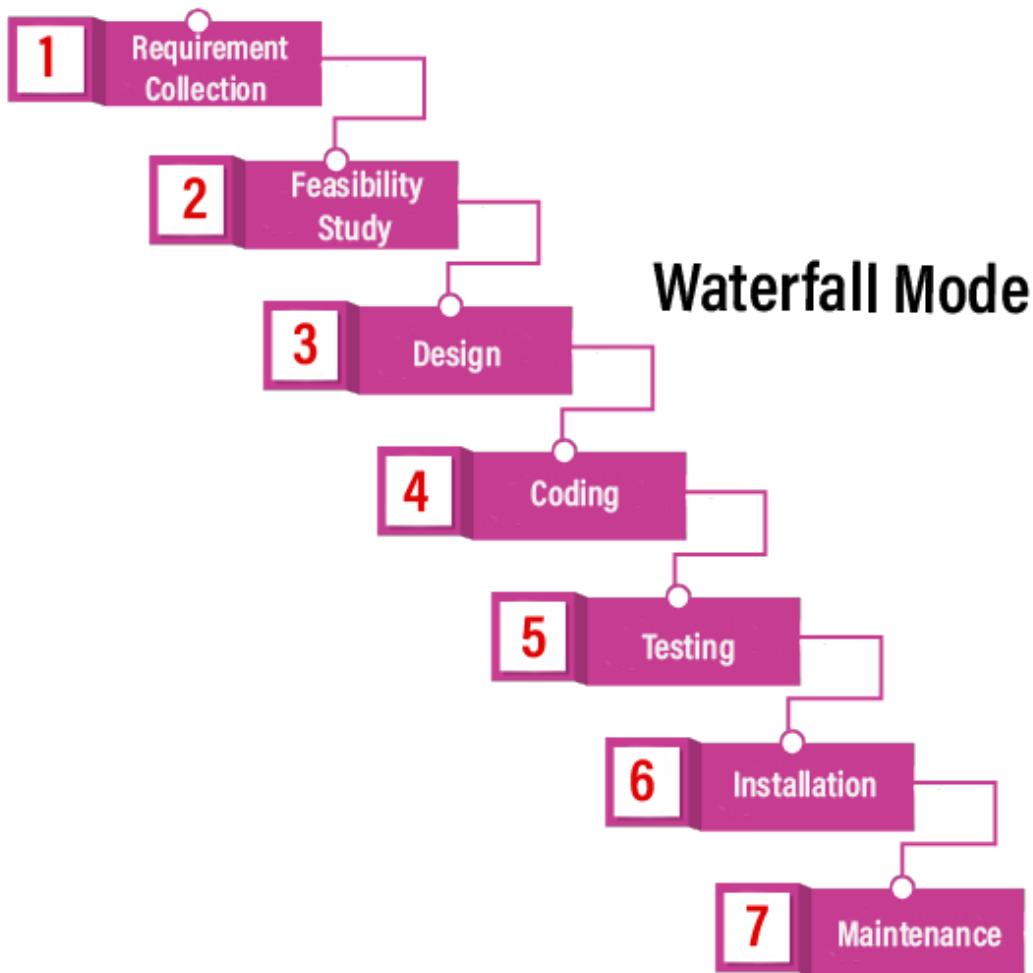
Requirement Collection

Requirement collection is the first phase of the waterfall model, where a business analyst will assemble all the information or business needs of the client in the form of a requirement document. And this document should be clear and easy to understand, and all requirements are correctly listed.

By taking help of Software Requirement Specification [SRS], Customer Requirement Specification [CRS], and Business Requirement Specification [BRS], the SRS document is generated. And this SRS document covers the whole thing that should be developed and designed.

Features of a functional requirement

- It should be written in a simple language so it can be easily understandable.
- The specification should be in the proper flow.
- The requirement should be countable.



Feasibility Study

The feasibility study is based on the needs of the project, where many people (human resource, business analyst, architecture) evaluate whether the project can be done or not. To develop a good project, we should follow the various characteristics, which are based on the customer requirements:

Aspects	Description
Legal	Can the company handle the project as cyber law and other monitoring agreements?
Technical	Check whether the available machine supports the software or not?
Operation feasibility	The company should be able to generate operation that is given by the clients?
Economic	Should the company be able to complete the product within given budget or not?
Schedule	The project should be done within the given schedule or not.

Design

Once we are done with the feasibility study, we will move to our next stage, which is designing. In this, we will create the architecture of the product, with the help of some essential tools like a combination of different software and hardware, various programming languages (PHP, Java, .Net, etc.), database (MySQL, Oracle). And then the designer gets ready with a plan for the application that could be classified into two different parts:

- **High-Level Design**
- **Low-Level Design**

High-Level Design [HLD]:

In this, the designer will concentrate only on the models such as decision trees, flow diagrams, decision tables, flow charts, data dictionary, and the architect does it.

Low-Level Design [LLD]:

In this, the designer will concentrate on the components like a User interface (UI), and the developer manager does it.

Coding

Once we are done with the design stage, we are ready to develop the application. For this, the developer will start writing the code based on their programming language knowledge, and it could be any language such as Python, C, Java, C#, C++, and so on. Whereas the back-end developers will do the back-end coding based on the needed operations, and the front-end developers will develop the attractive GUI.

Testing

After the compilation of coding, it will hand over to the concern test engineer. And after that, the test engineer will start testing the functionality of the application based on the client's requirement.

While testing the application, they may encounter some defects or bugs (not working as per the client's needs) in the application and send those bugs to the developer with the proper justification. And the developer will verify that the given bug is valid or not. If it is correct, it will be fixed by the developer and change with the new one. After that tester will re-test it and verify that the bug is fixed or not.

Installation

Once the application is tested, we will move to the next stage (installation). In this, the process will remain until the software is stable or bug-free and fulfilling all the customer requirements. When the application is stable, it will install into the client's environment for their use.

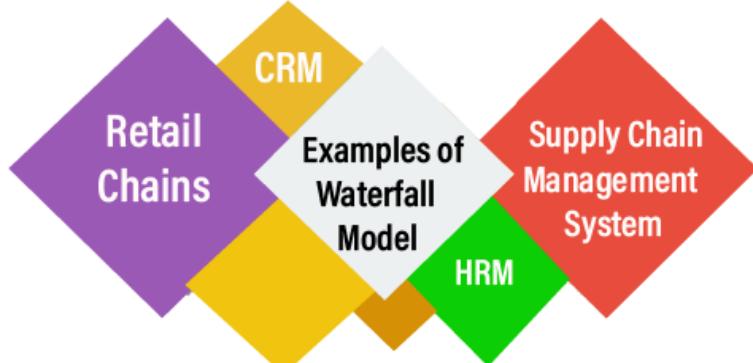
After getting the software, the client will perform one round of testing to their satisfaction. If they face any error, they will inform the development team to resolve those issues for the particular application. When all the issue gets resolved, the application will be deployed for the end-users use.

Maintenance

After completing the six stages successfully, we will move to the last stage (maintenance) of the waterfall model. In this, the process will remain until the software comes to an end, where the end-user starts using the application, and they may have some issues that need to be tested and fixed. Taking care of the product, time to time is called the maintenance, which includes the variations that happen in the hardware and software to maintain the operational effectiveness and also increase the performance.

Example of the waterfall model

Earlier it was used for the applications such as **Human Resource Management [HRM]**, **Supply Chain Management System**, **Customer Relationship Management [CRM]**, and **Retail Chains**, etc. but now in present time, the waterfall models are replaced by other models such as **Iterative models and Agile Methodology**, etc.



Pros and Cons of the Waterfall Model

Pros	Cons
In the Waterfall model, the requirement should be clear.	This model has no parallel deliverable, which means that two teams can work together.
It is suitable for a smaller project where needs are well understood.	The waterfall model doesn't provide the requirement changes and requirement review.
This model is easy to understand, as well as easy to use.	Previously, when the waterfall is invented, there is no concept of testing, that's why the developer is used to test the application.
It will allow us to arrange the tasks efficiently.	In between, changes are not allowed because one phase is dependent on another stage.
In this model, release level changes are allowed.	Backward tracking is not possible.
In this model, the procedure and the results are well documented.	It is a time-consuming process.

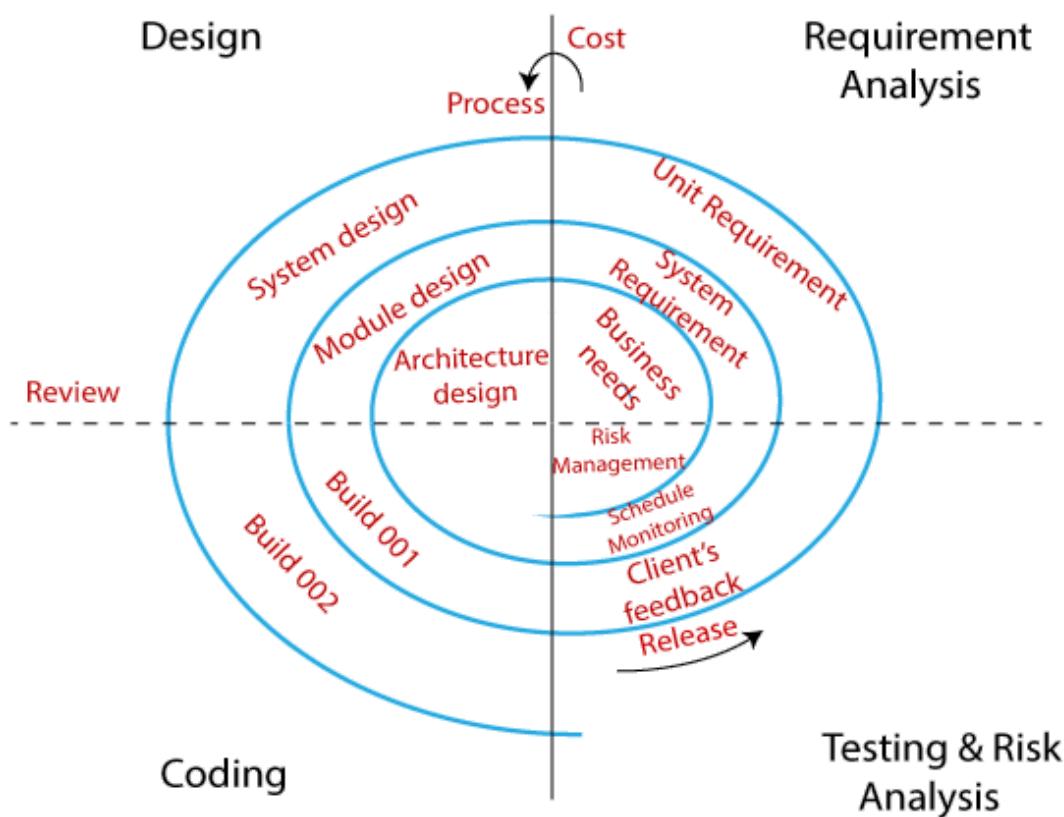
Spiral Model

The biggest problem we face in the waterfall model is that taking a long duration to complete the product, and the software became outdated. To solve this problem, we have a new approach, which is known as the Spiral model. The spiral model is also known as the cyclic model.

In this model, we create the application module by module and handed over to the customer so that they can start using the application at a very early stage. And we prepare this model only when the module is dependent on each other. In this model, we develop the application in the stages because sometimes the client gives the requirements in between the process.

The different phases of the spiral model are as follows:

- **Requirement analysis**
- **Design**
- **Coding**
- **Testing and risk analysis**



Requirement Analysis

The spiral model process starts with collecting business needs. In this, the following spirals will include the documentation of system requirements, unit requirements, and the subsystem needs. In this stage, we can easily understand the system requirements because the business analyst and the client have constant communication. And once the cycle is completed, the application will be deployed in the market.

Design

The second stage of the spiral model is designed, where we will plan the logical design, architectural design, flow charts, decision tree, and so on.

Coding

After the compilation of the design stage, we will move to our next step, which is the coding stage. In this, we will develop the product based on the client's requirement and getting the client's feedback as well. This stage refers to the construction of the real application in every cycle.

And those spirals had an excellent clarity of the requirements, and the design details of an application are known as the build with having version numbers. After that, these builds are transferred to the client for their responses.

Testing and Risk Analysis

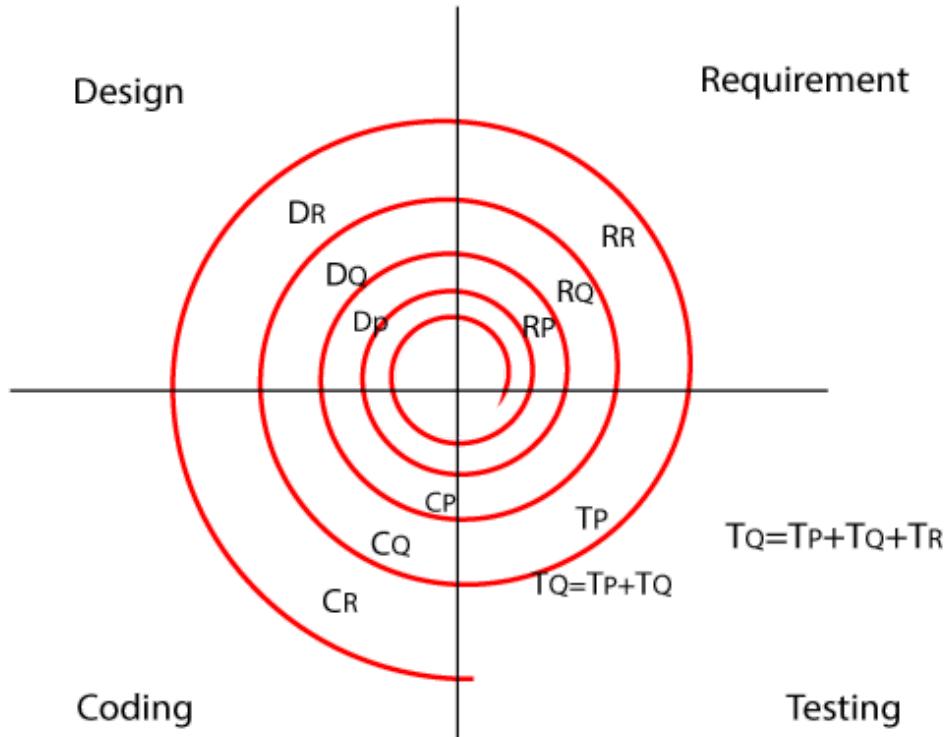
Once the development is completed successfully, we will test the build at the end of the first cycle and also analyze the risk of the software on the different aspects such as managing risks, detecting, and observing the technical feasibility. And after that, the client will test the application and give feedback.

Example of the Spiral model

Let us see one example for a better understanding of the spiral model:

In the spiral model, the software is developed in the small modules. Suppose we have the application A and this A application is created with the help of different models as P, Q, R.

Examples of spiral model



In the above image,

RP: the requirement analysis of module P, similarly with RQ, RR.

DP: Design of module P, and similarly with DQ, DR.

CP: Coding of module P, and similarly CQ, CR.

TP: Testing of module P, and similarly TQ, TR.

- In the P module, we get the requirement first, and then only we design the module. And the coding part of module A is done when it is tested for bugs.
- The next module is Q, and it has been created when the module P has been built. We follow the same process as we did in module P, but when we start testing the module Q, and we check the following condition such as:
 - Test the Q module
 - The test integration of module Q with P
 - Test module P
- After creating the module P, Q, we will proceed to the module R, where we will then follow the same process as module P and Q, and then test the following conditions:
 - First, check the module as R, Q, and P

- Then, check the integration of module in the below order: R → Q, R and P → P and Q

Note: Once the cycle continues for multiple modules, the module Q can be built only after the module P has been built correctly and similar for module R.

The best-suited example for the spiral model is **MS-Excel** because MS-Excel sheet having several cells, which are the components of an excel sheet. Since we have to create the cells first (module P), then we can perform operation on the cells like split cells into half (module Q), merge cells into two, and then we can draw graphs on the excel-sheet (module R).

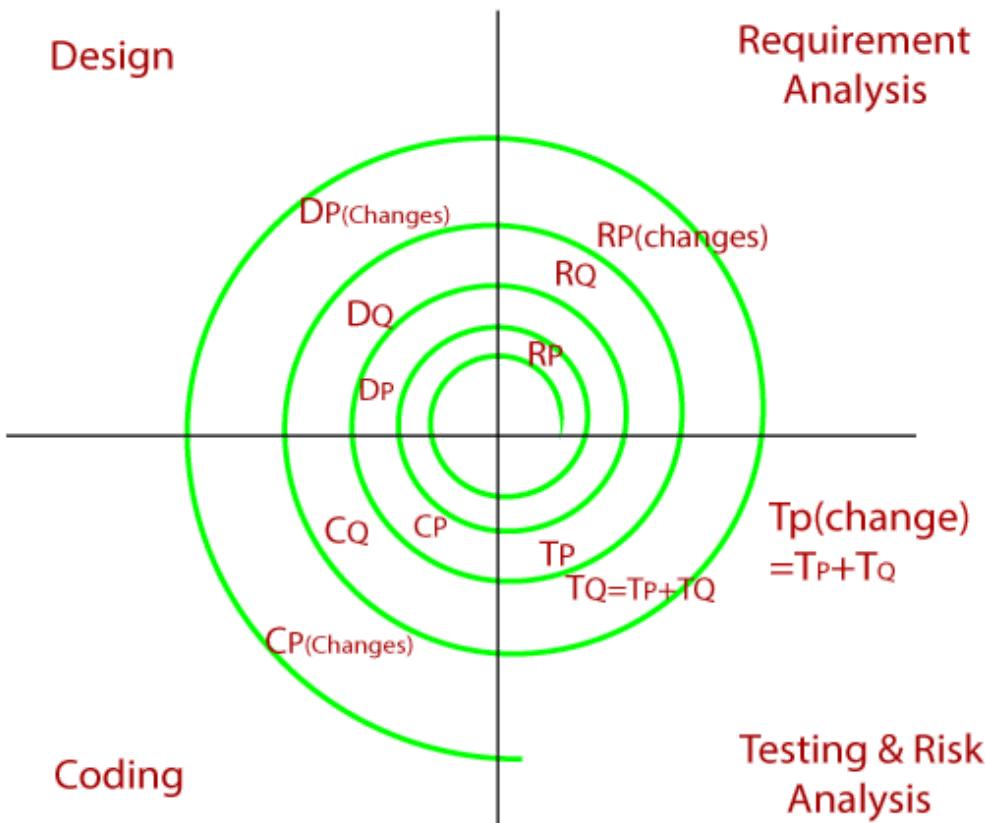
In the spiral model, we can perform two types of changes, which are as follows:

- Major changes
- Minor changes

Major Changes

When the customer request for the major changes in the requirements for the particular module, then we change only that module and perform testing for both integration and unit. And for this, we always prefer one new cycle because it may affect the existing modules. Major changes could be the functionality of the software.

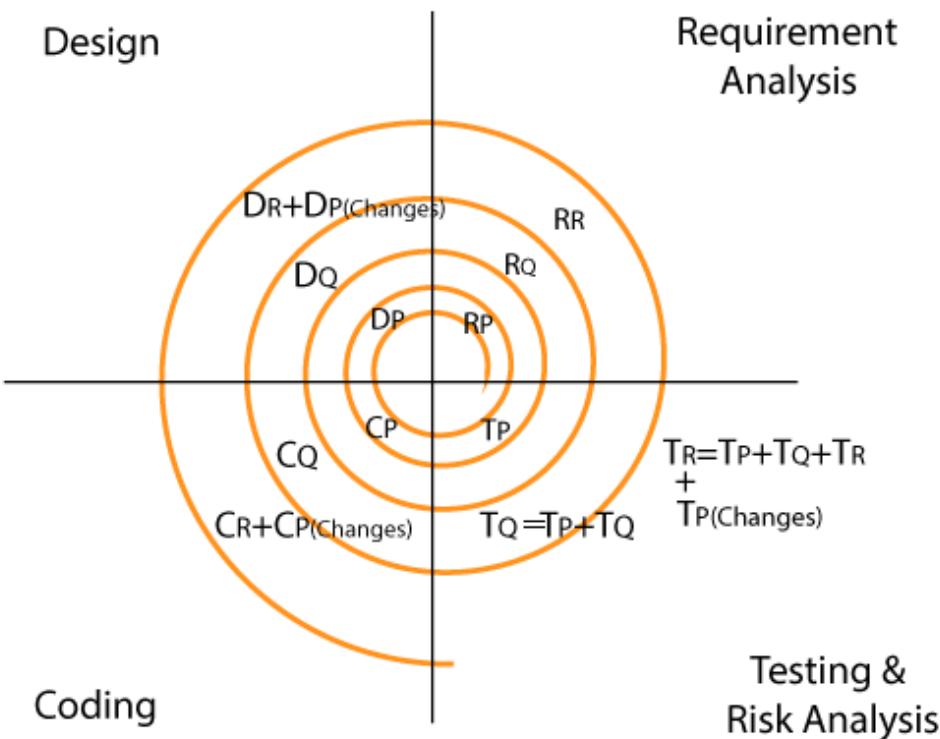
Major Changes



Minor changes

Whenever the client request for the minor changes in the particular application, then the software team makes the smaller changes along with the new module has to be developed simultaneously in a single cycle. And we never go for any new cycle or iteration because a minor variation does not affect the existing functionality, and it also takes the extra resource and time. The minor changes could be UI (frontend changes).

Minor Changes



Advantage and disadvantage of the spiral model

Advantage	Disadvantage
Flexible changes are allowed in spiral model.	It is not suitable for the small and low-risk product because it could be costly for a smaller project.
The development can be distributed into smaller parts.	It is a traditional model, and thus developers only did the testing job as well.
The customer can use the application at an early stage also.	There is no requirement of review process and no parallel deliverables allowed in the spiral model.
More clarity for Developers and Test engineers	In the spiral model, management is a bit difficult; that's why it is a complex process.
It will provide the wide use of prototypes.	The maximum number of intermediate phases needs unnecessary paperwork.

Hybrid Model

The hybrid model is the combination of two or more primary (traditional) models and modifies them as per the business requirements. This model is dependent on the other SDLC models, such as spiral, V and V, and prototype models. The hybrid model is mainly used for small, medium, and large projects. It focuses on the risk management of the product.

We go for the hybrid model whenever we want to obtain the features of two models in a single model. And when the model is dependent and the customer is new to the industry.

The most commonly used combination of two models is as follows:

- **Spiral and prototype**
- **V & V and Prototype**

Note: The waterfall model cannot be combined with any model because there is no requirement of review.

Spiral & Prototype

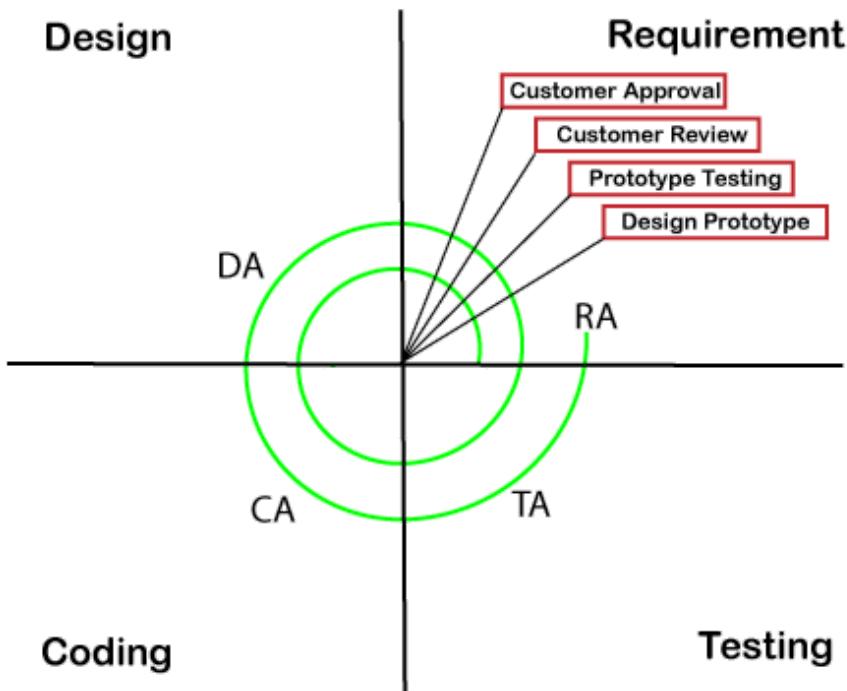
The spiral and prototype model is used for the below conditions:

- We go for spiral and prototype models whenever there is dependency.
- When the customer gives requirements in stages, and we develop the product in stages using this model.
- When the customer is very new to the software industry and not clear about the requirements.
- When the developers are new to particular software.

Process of Spiral and Prototype model

The process of spiral and Prototype model will complete in various steps, which are as follows:

Spiral and prototype model



- The process of this model starts with **collecting the requirements** from the customer for the different modules like A, B, and C
- After collecting the business needs of the software, we will **create the prototype A**.
- Once we develop the Prototype, we will **test the Prototype A**.
- After successfully testing the Prototype, we will send it to the customer for their **review and approval**.

- Once they reviewed and approved the Prototype, we will design that Prototype for the actual module.
- Once the designing phase completes, the **developer starts writing the code** for the modules.
- After the completion of development, it will send it to the testing team, where **they will test the module**.
- And when the testing phase is done, it will **deploy to the customer**. And this process is continued until all the modules (B, C) present in the software.

V & V and prototype Model

We go for this model for the following reasons:

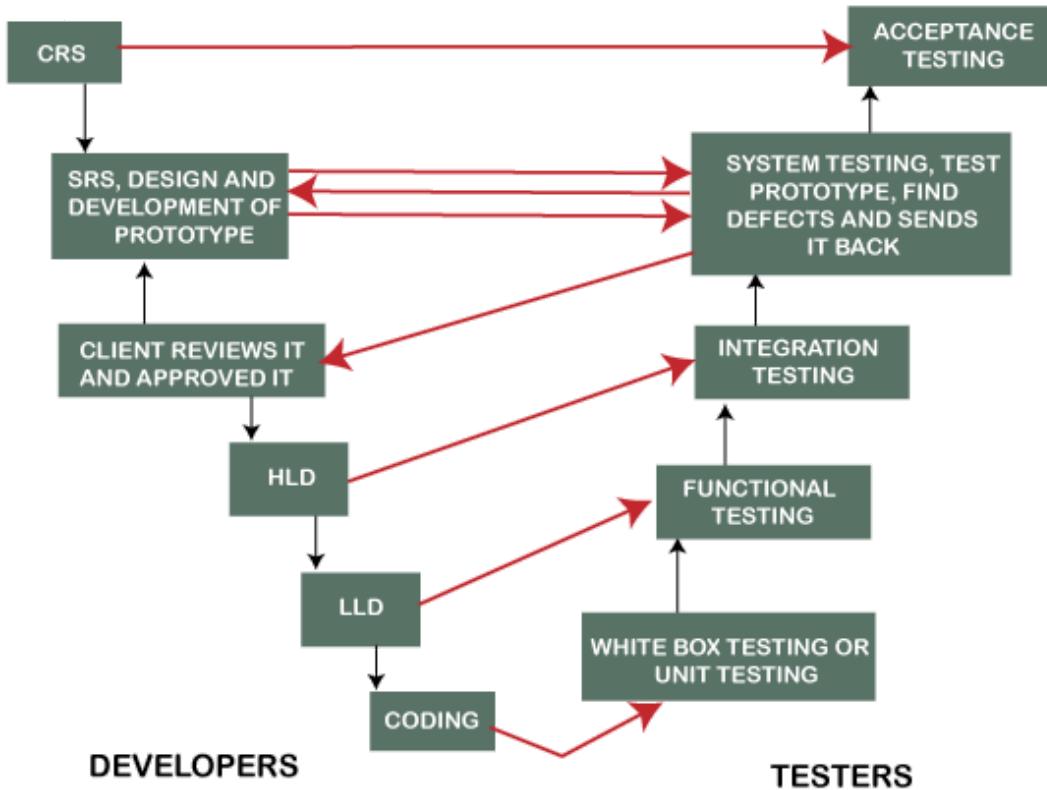
- When the customer and developers are both new to the industry.
- When the clients are expecting a very high-quality product within the required time because every phase is tested, and the developer and testing team are working parallelly.

V & V and Prototype model process

In the hybrid model, the testing team is involved to test the Prototype.

In this, testing will start from the early stage of product development, which avoids the downward flow of bugs, which helps us to reduce the re-work.

V & V and prototype model



The V & V model and prototype process will be completed in the following steps:

Step1

The process starts with collecting the business needs in the form of CRS documents, and the tester will do the following:

- Review the CRS
- And write the user acceptance test case and test plans.

Step2

Then BA will convert this CRS document to the SRS document, and the web developer will design and develop the Prototype, send it to the tester, and the tester will test the following:

- First, they will review the SRS document.
- And Write the system testing test cases and test plans.

Step3

After that, the testing team will check the Prototype and identify the bugs and send it back to the concerned developer. Once the prototype testing is done, it is sent to the customer for their review and approval.

Step4

Once the customer approves it, we will design the high-level design of the particular Prototype and send it to the testing team where they will do the following:

- Review the HLD
- Writes the integration testing test documents.

Step5

Once it is done, we will start working on the low-level design and send it to the tester where they will perform the following:

- Review the LLD
- And writes the functional test cases and test plans.

Step6

After that, the developer starts writing the code for the particular Prototype and does one round of white box testing from their end and send it to the testing team for further testing where they perform various type of Testing.

This process is going on until the modules and prototypes are stable. And then it will deliver to the customer.

Advantage and disadvantages of Hybrid Model

Following are the pros and cons of the hybrid model:

Advantages

- The hybrid model is highly flexible.
- In this model, the customer rejection is less because of the Prototype.
- It is easy to implement because it has the flexibility of synchronization.
- It is easy to use and apply, especially with small and medium projects.
- In this, the development process will be smooth and quick because here we follow only the relevant process cycles.

Disadvantages

- Every hybrid model is different from each other.
- It does not follow the usual standards.

Prototype Model

The most significant disadvantage of previous models (waterfall and spiral) is that there were lots of customer rejection that happens after the application was developed, and there was no involvement of the customers in between the project.

Hence, they started the new approach, which is known as the **prototype model**. In this, we will collect the requirements from the customer and prepare a **prototype (sample)**, and get it reviewed and approved by the customer. And only when they satisfied, we will start working on the original projects so that there won't be any customer rejection.

The prototype is just the sample or a dummy of the required software product. If all the mentioned modules are present, then only the developer and tester will perform prototype testing.

When we use the Prototype model

Generally, we go for this model because of the following reasons:

- Whenever the customer is new to the software industry or when he doesn't know how to give the requirements to the company.
- When the developers are new to the domain.

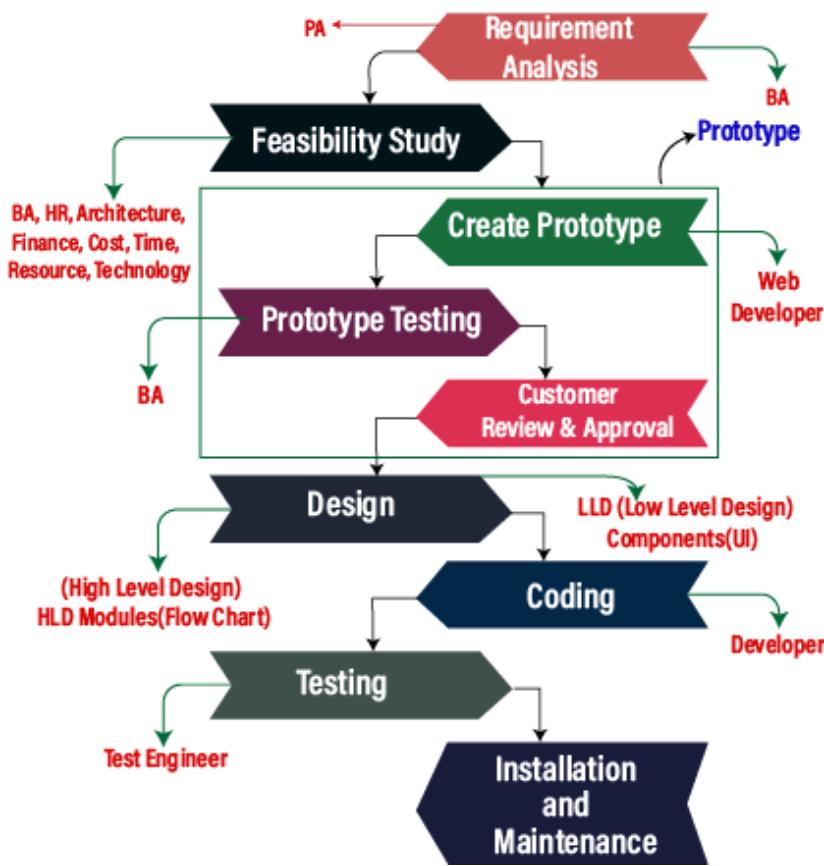
Note: the difference between the testing and prototype testing is that- in the testing, we will work on the functionality, which gives some input and output. And in the prototype testing, we will test only the look and feel, which means that the UI and frontend.

Prototype model process

Prototyping Model has different phases, which are as follows:

- **Requirement analysis**
- **feasibility study**
- **Create a prototype**
- **Prototype testing**
- **Customer review and approval**
- **Design**
- **Coding**
- **Testing**
- **Installation and maintenance**

Prototype Model



Requirement analysis

This model starts with collecting the requirements from the customers. And these requirements of the project should be in-detailes. These details are received by the Business Analyst and Product Analyst. Where **Business analyst** is assigned for **service-based software companies**, and the **Product analyst** is assigned for **product-based software companies**.

Feasibility study

In the next stage, the **BA, HR, Architecture, and Finance** teams head will sit together and talk about the cost of the product, which resource is going to be needed, which technology is used to develop the product and how much time is required to complete the product and deliver.

Create a prototype

After we completed the feasibility study, we will move to our next stage, where we will be creating the prototype (sample or dummy) based on the data collects from the client, and the web developer will design the prototype.

Here, we have the following types of prototype:

- **Static prototype**
- **Dynamic prototype**

Static prototype

In the static prototype, we kept the entire prototype of the requirements in a word document with having all the guidelines, screenshot, and the description of how to build the software, how the completed product will look like and how it will work and so on.

Dynamic prototype

The dynamic prototype is parallel to the browser, but here we can't provide any details, only the functionality is there without entering the data. It is like a dummy page made out of the html with having tags and links to the various pages to the expressive features of the product.

Prototype testing

Once we build the prototype, the BA will test the prototype and perform one round of prototype testing.

Note: The prototype testing is testing, where we will test only the look and feel, which means that the UI and frontend.

Customer review and approval

Once the prototype testing is done, it will be handed over to the customer for their review and approval. If the customer is not happy with the given sample, we will change the prototype based on the customer's guidelines and feedback. This process will go on until the customer approved and satisfied with the prototype. It is a bit time-consuming because we have to perform the changes again and again in the prototype.

Design

After getting the approved prototype, we will start the high level and low-level design for the final product and consider all the suggestions given by the customer at the time of the final prototype.

Coding

Once the design phase has been completed successfully, we move to our coding phase, where the concerned developer starts developing the product based on their programming knowledge.

Testing

After the compilation of the development phase, it is handed over to the test engineer. And the test engineer test the application functionality, and all inputs and outputs.

Installation and maintenance

Once our final product is developed and tested according to the final prototype, it will be deployed to the production. And the product will go through the time to time maintenance to reduce any interruption, which helps to avoid significant failures.

Note:

- Starting from the **Requirement collection** to **Customer review**, the documented format is converted to a prototype format because it is an extended requirement collection phase, and the actual design begins from the design phase.
- Previously, prototype development is done by developers. Still, now it is done by content developers or web designers where they develop the prototype of the product with the help of some tools.
- In this, the client gets a chance in the starting itself to ask for changes in the requirement as it is easy to do requirements changes in the prototype rather than the actual application. Therefore the cost will reduce, and expectations are met.

Advantage and disadvantage of the prototype model

There are the following advantages and disadvantages of the prototype model:

Advantage	Disadvantage
We can easily detect the missing functionality.	It is a time-consuming process because if customer changes in the prototype. And it will also waste our time by changing again and again in the dummy (prototype), which will delay the working of the real project.

In this, the development team and customer have clear communication regarding the requirements and the outcome of the product.	There is no requirement review, but the prototype review is there.
In this, customer satisfaction exists.	There are no parallel deliverables, which means that the two teams cannot work together.
We can re-use the prototype in the design phase and for similar applications.	Sometime the partial application may cause the software not to be used as the complete system was designed.
In this model, customer rejection is less as compared to the other models.	Insufficient or partial problem analysis.
Issues can be identified in the early phase.	We may also lose customer attention if they are not happy with the final product or original prototype.

V-model/ V and V model /Verification and Validation model

This model came up to overcome the drawback of the waterfall model. And in this model, testing starts from the requirement stage itself.

In this model, first, all the activities go on the **downward direction**, and at one point in time, it starts moving in the **upward direction** to re-use the test document for the testing process and forms a **V** shape. Hence it is known as the **V model**.

When we go for this model

We go for V and V model for the following reasons:

- For the large and complex application, here, large means that the n numbers of modules and complex specify lots of dependencies between modules.
- And It is also used for the long term projects.

Before going further in this model, first, we will understand the requirements:

Requirements

It is a document which is collected from the customer; here, we have two different types of requirements documents, which are as follows:

- **CRS/BRS**
- **SRS/FS**

CRS/BRS

The CRS or BRS stands for **Customer Requirement Specification or Business Requirement Specification**. For the CRS, the details will be written in the simple business (English) language by the BA (business analyst), which cannot be understood by the developers and the test engineers.

Let us see one sample example for Customer Requirement Specification to the Gmail application:

1.	Customer secured entry
2.	Optional creates mails
3.	Able to see mails
4.	Unwanted content delete

15.	Successfully close the application.

SRS/ FS

It stands for **Software Requirement Specifications** or the **Functional Specification**; in this, all the details are converted to the detail document, which can be understood by the developers and the test engineers.

Let us see one sample example for Software Requirement Specifications to the Gmail application:

1.	Login (module)
1.1	User name→ Text box (functional specification)

1.1.1	User name→ Accept only 5 alphabets
1.2	Password→ text box
1.2.1	Password→Accept only 8 characters, in which one should be capital and one special character(@,\$,%,&)
1.3	OK→ Button
1.3.1	OK→ enabled
2.	Compose
2.1	To→Text Box

3.	Inbox
3.1	-----

4.	Logout

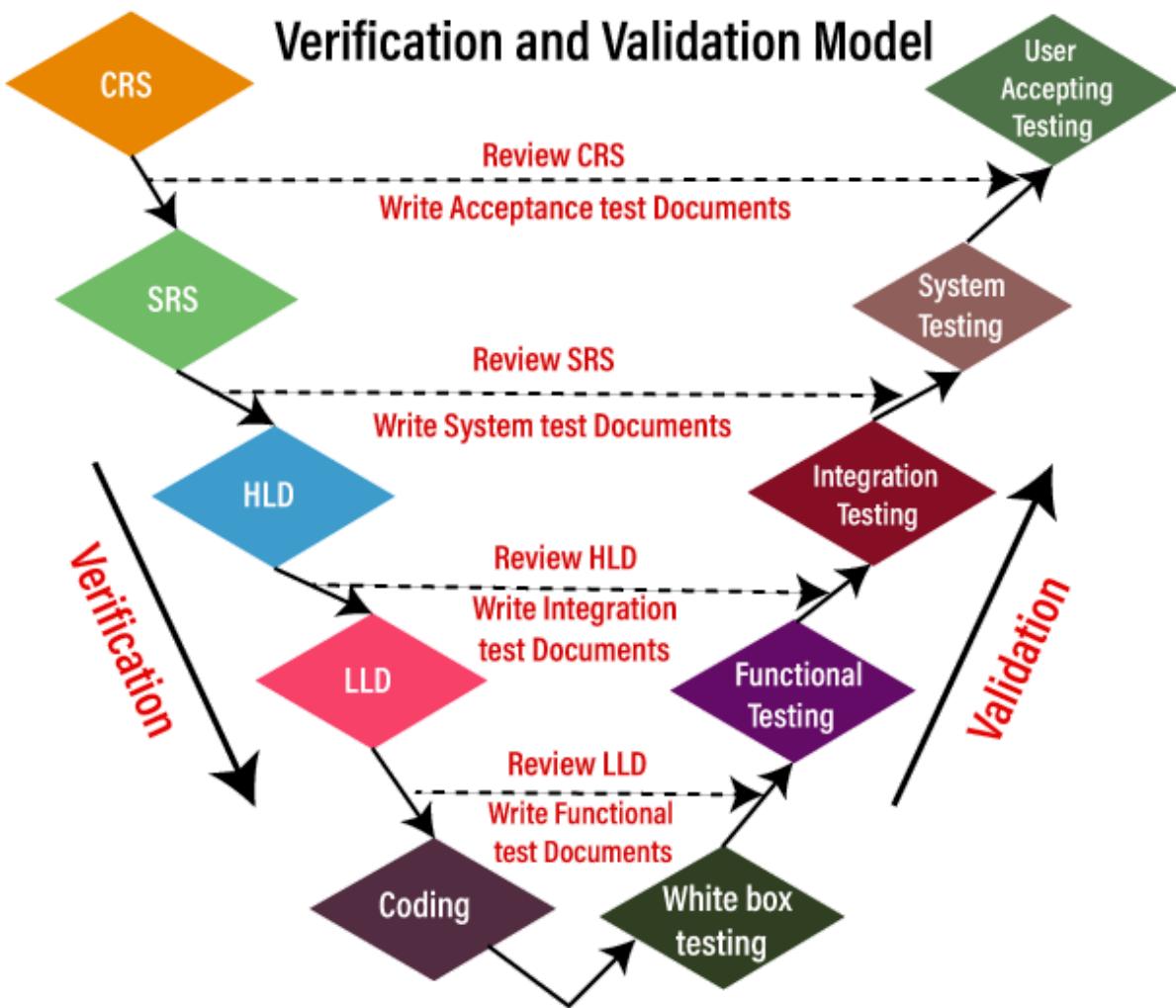
Characteristics of a functional requirement

- The requirements should be **In-Details**, which means it has all the details about **modules, components, and the functional specification** and in the **Proper flow**, which means that it should be in the **sequence order**.
- The requirements should be written in a simple language which is easy to understand by everyone.
- The requirement should be measurable or Countable.

V and V Model Process

The entire V model executes in two-phase, the complete **review process** is done in the **verification phase**, and the whole **testing process** is done under the **validation phase**; that's why it is also known as **verification and validation** model.

Where the verification and validation process includes different stages:



Stage 1

It will start from collecting the CRS (customer requirement specification) document, from the client by the Business Analyst where the test engineer will check the following scenarios:

- **Review the CRS** based on
 - Incorrect requirements
 - Missing requirements
 - Conflicts in the requirements
- **Write Acceptance Test documents**

Note: In all the stages, the Test Documents includes the test plans and test cases.

Once the test engineer team reviews the CRS and found any bugs or defects, they will send it to the development team for fixing the bugs. After fixing the bugs, the development team updates the CRS and concurrently developing the SRS document.

Stage 2

After completing the CRS, the SRS is sent to the testing team for the review process, and the developers start creating the HLD (high-level design) for the application. And the testing team will test the SRS on the following scenarios:

- **Review the SRS against CRS**
 - Each CRS is transferred to SRS
 - CRS is not transformed properly to SRS
- **Write the system Test documents**

Once the testing team reviews every detail of the SRS and CRS has been converted correctly to SRS, we will move to our next stage.

Stage 3

After the completion of HLD, the developers start creating the LLD (Low-level design) for the application, and in the meantime, the tester will check the following tests on the HLD:

- **Review HLD**
- **Write integration test documents**

Stage 4

Once the testing team has done reviewing the HLD, the developers write the coding and develops the application, and the testing team will do the following tasks:

- **Review the LLD**
- **Write functional test documents**

Stage 5

After the completion of the coding part, the developers will perform one round of unit testing, which is also called white box testing, and check every line of the code and make sure that the code is correct.

After performing the unit testing, the application is sent to the testing team, where they perform multiple testing such as **functional testing, integration testing, and system testing, and acceptance testing**.

And once the testing part is done, the application will finally deliver to the customer.

Note: How to handle requirement changes in V and V? Whenever there is a change happen in the requirement, the same procedure continues, and the documents will be updated.

Advantage and Disadvantage of V and V Model

Let us see the pros and cons of the V and V model:

Advantage	disadvantage
In this, review exists in every phase, that's why we may get less number of bugs in the application.	It is a bit expensive process because Initial investment is high as the testing team is needed from the starting stage itself.
The V model provides the Parallel deliverable, which implies that the two teams can work together like here; the development and testing team are working parallelly.	It is a time-consuming process because if requirement changes happen, we need to change every text documents.
This model helps to deliver Robust or stable products.	In this, we need to do more documentation work because of the test cases and all other documents.
In this model, the test Engineers have more knowledge about the product because testing is involved in every stage of product development.	The V model is not suitable for object-oriented projects.
The text document can be re-used.	We cannot go back and replace the functionality once the application is in the testing phase.

Manual Testing

Manual testing is a software testing process in which test cases are executed manually without using any automated tool. All test cases executed by the tester manually according to the end user's perspective. It ensures whether the application is working, as mentioned in the requirement document or not. Test cases are planned and implemented to complete almost 100 percent of the software application. Test case reports are also generated manually.

Manual Testing is one of the most fundamental testing processes as it can find both visible and hidden defects of the software. The difference between expected output and output, given by the software, is defined as a defect. The developer fixed the defects and handed it to the tester for retesting.

Manual testing is mandatory for every newly developed software before automated testing. This testing requires great efforts and time, but it gives the surety of bug-free software. Manual Testing requires knowledge of manual testing techniques but not of any automated testing tool.

Manual testing is essential because one of the **software testing** fundamentals is "100% automation is not possible."

Why we need manual testing

Whenever an application comes into the market, and it is unstable or having a bug or issues or creating a problem while end-users are using it.

If we don't want to face these kinds of problems, we need to perform one round of testing to make the application bug free and stable and deliver a quality product to the client, because if the application is bug free, the end-user will use the application more conveniently.

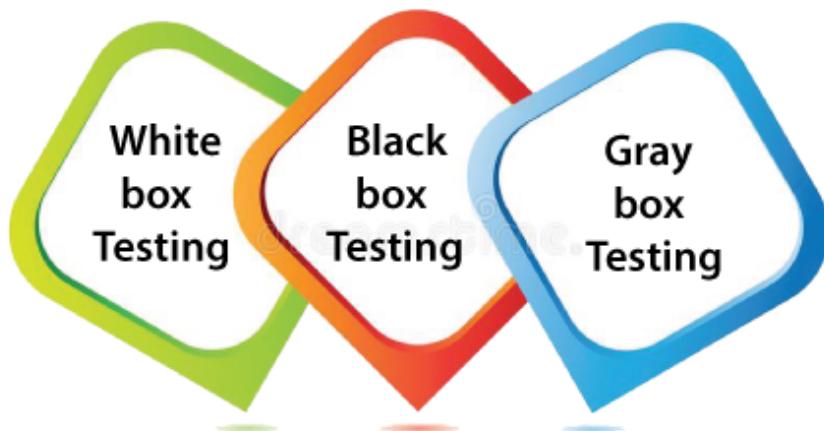
If the test engineer does manual testing, he/she can test the application as an end-user perspective and get more familiar with the product, which helps them to write the correct test cases of the application and give the quick feedback of the application.

Types of Manual Testing

There are various methods used for manual testing. Each technique is used according to its testing criteria. Types of manual testing are given below:

- White Box Testing
- Black Box Testing
- Gray Box Testing

Types of Manual Testing



White-box testing

The white box testing is done by Developer, where they check every line of a code before giving it to the Test Engineer. Since the code is visible for the Developer during the testing, that's why it is also known as White box testing.

For more information about white box testing, refers to the below link:

<https://www.javatpoint.com/white-box-testing>

Black box testing

The black box testing is done by the Test Engineer, where they can check the functionality of an application or the software according to the customer /client's needs. In this, the code is not visible while performing the testing; that's why it is known as black-box testing.

For more information about black-box testing, refers to the below link:

<https://www.javatpoint.com/black-box-testing>

Gray Box testing

Gray box testing is a combination of white box and Black box testing. It can be performed by a person who knew both coding and testing. And if the single person performs white box, as well as black-box testing for the application, is known as Gray box testing.

To get more details about gray box testing, refers to the below link:

<https://www.javatpoint.com/grey-box-testing>

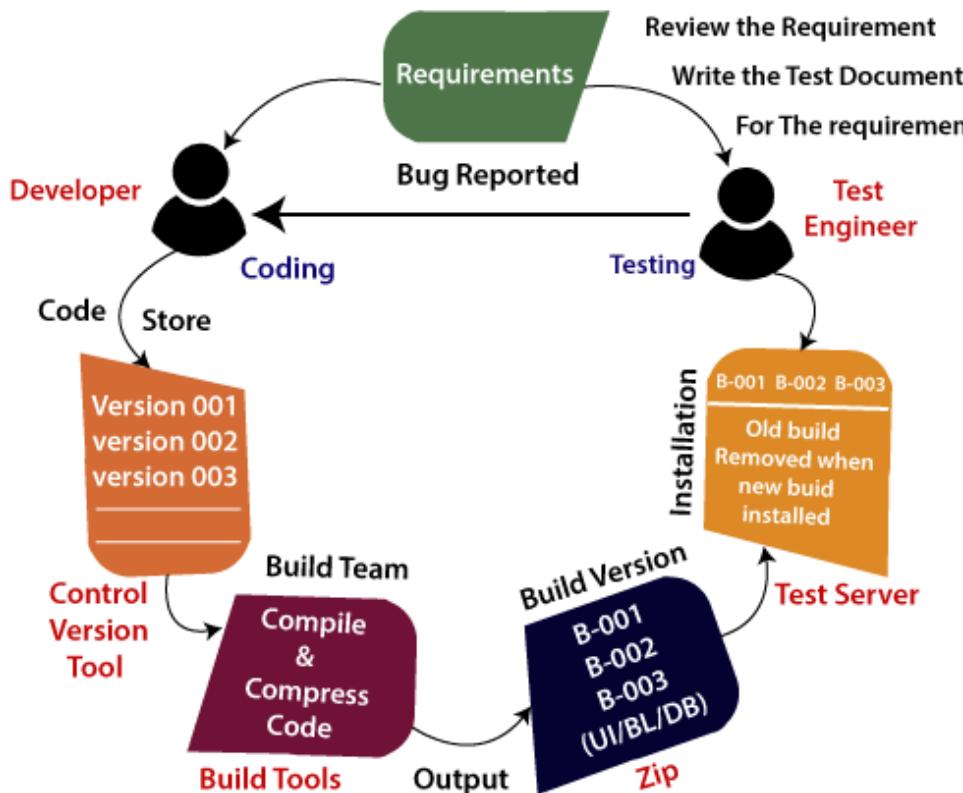
How to perform Manual Testing

- First, tester observes all documents related to software, to select testing areas.
- Tester analyses requirement documents to cover all requirements stated by the customer.
- Tester develops the test cases according to the requirement document.
- All test cases are executed manually by using Black box testing and white box testing.
- If bugs occurred then the testing team informs the development team.
- The Development team fixes bugs and handed software to the testing team for a retest.

Software Build Process

- Once the requirement is collected, it will provide to the two different team development and testing team.
- After getting the requirement, the concerned developer will start writing the code.
- And in the meantime, the test engineer understands the requirement and prepares the required documents, up to now the developer may complete the code and store in the **Control Version tool**.
- After that, the code changes in the UI, and these changes handle by one separate team, which is known as the **build team**.
- This build team will take the code and start compile and compress the code with the help of a build tool. Once we got some output, the output goes in the zip file, which is known as **Build** (application or software). Each Build will have some unique number like (B001, B002).
- Then this particular Build will be installed in the test server. After that, the test engineer will access this test server with the help of the Test URL and start testing the application.
- If the test engineer found any bug, he/she will be reported to the concerned developer.
- Then the developer will reproduce the bug in the test server and fix the bug and again store the code in the Control version tool, and it will install the new updated file and remove the old file; this process is continued until we get the stable Build.

- Once we got the stable Build, it will be handed over to the customer.



Software Build Process

Note1

- Once we collect the file from the Control version tool, we will use the build tool to compile the code from high-level language to machine level language. After compilation, if the file size will increase, so we will compress that particular file and dumped into the test server.
- This process is done by **Build team, developer** (if build team is not there, a developer can do it) or the **test lead** (if the build team directly handle the zip and install the application to the test server and inform the test engineer).
- Generally, we can't get a new Build for every bug; else, most of the time will be wasted only in creating the builds.

Note2

Build team

The main job of the build team is to create the application or the Build and converting the high-level language into low-level language.

Build

It is software, which is used to convert the code into application format. And it consists of some set of features and bug fixes that are handed over to the test engineer for testing purposes until it becomes stable.

Control version tool

It is a software or application, which is used for the following purpose:

- In this tool, we can save different types of files.
- It is always secured because we access the file from the tools using the same login credentials.
- The primary objective of the tools is to track the changes done for the existing files.

Example of Build process

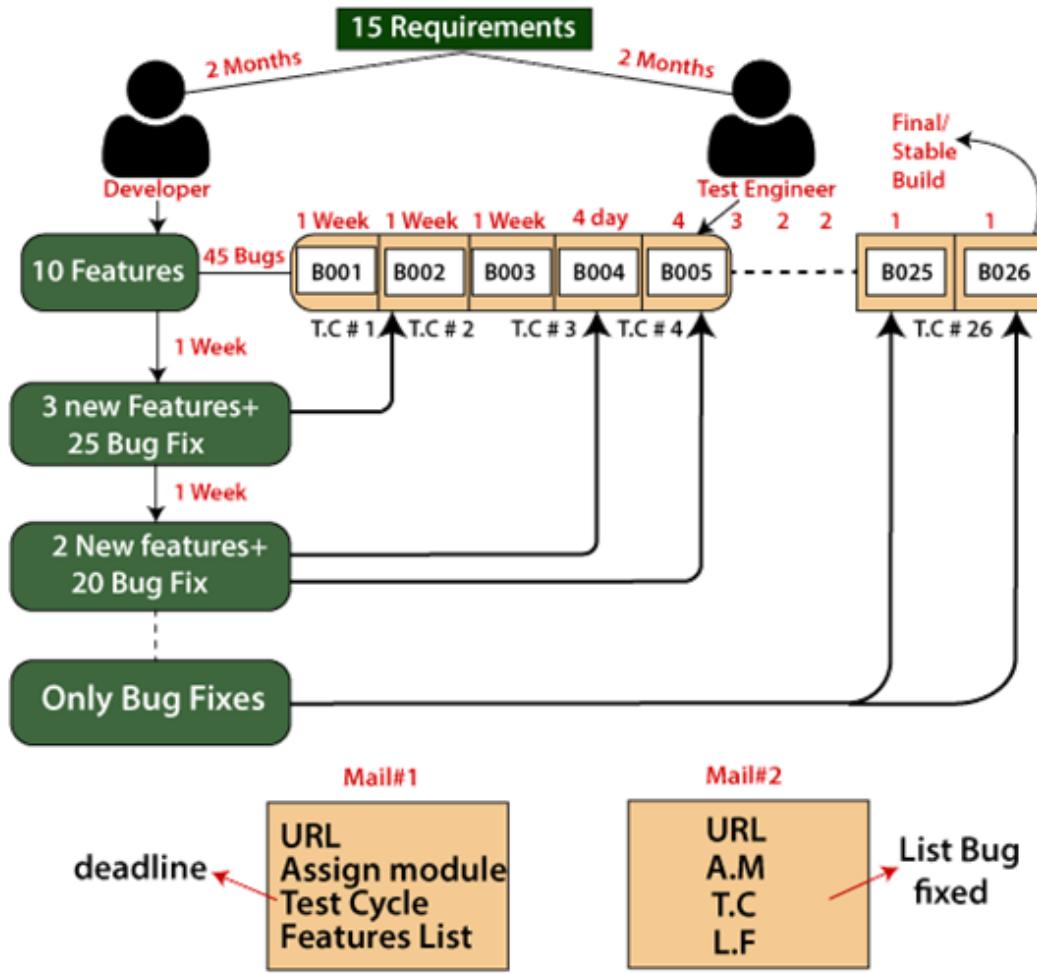
Let see one example to understand how to build process work on the real scenarios:

As soon as the test engineer gets the bug, they will send it to the developers, and they need some time to analyze; after that, he/she only fixes the bug (Test engineer can't give the collection of bug).

The developer is decided how many bugs he can fix according to their time. And the test engineer is decided, which bug should be fixed first according to their needs because the test engineers cannot afford to stop testing.

And the test engineer getting the mail, they can only know that which bug is fixed by the **list of the bug fixes**.

The time will increase because at the first Build, and developers should write the code in the different features. And at the end, he/she can only do the bug fixes and the numbers of days will be decreased.



Note3

Test cycle

The test cycle is the time duration given to the test engineer to test every Build.

Differences between the two build

The bugs found in one build and can be fixed any of the future Build, which depends on the test engineer's requirement. Each new Build is the modified version of the old one, and these modifications could be the bug fixes or adding some new features.

How frequently we were getting the new Build

In the beginning, we used to get weekly builds, but in the latest stage of testing, when the application was getting stable, we used to get the new Build once in 3 days, two days, or a daily basis as well.

How many builds we get

If we consider one year of any project duration, we got 22-26 builds.

When we get the bug fixes

Generally, we understand the bug fixes only after the test cycle is completed, or the collection of bugs is fixed in one build, and handover in the next builds.

Advantages of Manual Testing

- It does not require programming knowledge while using the Black box method.
- It is used to test dynamically changing GUI designs.
- Tester interacts with software as a real user so that they are able to discover usability and user interface issues.
- It ensures that the software is a hundred percent bug-free.
- It is cost-effective.
- Easy to learn for new testers.

Disadvantages of Manual Testing

- It requires a large number of human resources.
- It is very time-consuming.
- Tester develops test cases based on their skills and experience. There is no evidence that they have covered all functions or not.
- Test cases cannot be used again. Need to develop separate test cases for each new software.
- It does not provide testing on all aspects of testing.
- Since two teams work together, sometimes it is difficult to understand each other's motives, it can mislead the process.

Manual testing tools

In manual testing, different types of testing like unit, integration, security, performance, and bug tracking, we have various tools such as **Jira**, **Bugzilla**, **Mantis**, **Zap**, **NUnit**, **Tessy**, **LoadRunner**, **Citrus**, **SonarQube**, etc. available in the market. Some of the tools are open-source, and some are commercial.

For more information about testing tools, refers to the below link:

<https://www.javatpoint.com/software-testing-tools>



Let's us understand them one by one:

LoadRunner

It is most commonly used performance testing tools. LoadRunner is mainly used to support performance testing for the wide range of procedures, number of approaches, and application environments.

The main purpose of executing the LoadRunner tool is to classify the most common sources of performance issues quickly.



Features of LoadRunner

- LoadRunner tool contains n-numbers of applications, which reduces the time to understand and describe the reports.
- We can get thorough performance test reports by using the LoadRunner tool.
- It will reduce the cost of distributed load testing and also offer the operational tool for deployment tracking.

Citrus

Citrus is an integration testing tool, which is the most commonly used test framework. It is written in **Java programming language**. It is mostly used to request and respond to server-side and client-side and validate the XML **JSON** files.

To accomplish the end-to-end use case testing, citrus supports several HTTP, JMS, and SOAP protocols.



Characteristics of Citrus

Following are some of the important features of Citrus tool:

- It is used to send and receive messages.
- Citrus is available as both an open-source and a licensed in the market.
- It delivers a low-cost solution.
- We can authenticate the database by using the citrus tool.
- It will describe the sequence of messages, offer the test plan, and document the test coverage.
- It creates the message and verifies the responses.

ZAP

ZAP is an open-source web application security scanner. It stands for **Zed Attack Proxy**. Just like some other tools, it is also written in the **JAVA programming language**. It is the most effective **Open Web Application Security Projects** [OWASP].



Features of ZAP

- It supports many operating systems such as Windows, Linux, OS X.
- It has a plugin-based architecture.
- It contains an online marketplace that permits us to add new or updated features.
- ZAP's GUI control panel is easy to use.

NUnit

NUnit is one of the most frequently used unit testing tools. It is an open-source tool and primarily derived from the **JUnit**.

It was completely written in the **C# programming language** and suitable for all **.Net languages**.

In other words, we can say that the NUnit tool is entirely redesigned to become the advantage of many .Net language qualities. **For example:**

- **Reflection-related capabilities.**
- **Other custom attributes.**



Characteristics of NUnit

- It allows the assertions as a static method of the advantage class.
- It sustains the data-driven tests.
- It supports several platforms, like .NET core Xamarin mobile, Silverlight, and efficient framework.
- The ability of NUnit help us to execute the tests simultaneously.
- It uses a console runner to load and execute the tests.

JIRA

The most regularly used bug tracking tool is **JIRA**, which is an open-source tool. It is used for bug tracking, project management, and issue tracking.

In this tool, we can easily track all kinds of bugs or defects related to the software and produced by the test engineers.



Features of JIRA

- It is a time-saving tool.
- Jira is used to track the defects and issues.
- It is used to establish the documentation tasks.

- Jira is a very useful tool in tracking the improvement of our documentation.

To get the complete information about the Jira tool, refer to the below link: <https://www.javatpoint.com/jira-tutorial>.

SonarQube

Another testing tool of manual testing is SonarQube, which improves our workflow with continuous code quality and code security. It is flexible with the use of plug-ins.

It is completely written in the JAVA programming language. It offers fully automated evaluation and integration with Ant, Maven, Gradle, MSBuild, and constant integration tools. SonarQube has the ability to record a metrics history and gives the evolution graph.



Features of SonarQube

Below are some of the significant features of the SonarQube tool:

- It supports several programming languages like C, C++, Python, JAVA, HTML, CSS, VB.NET, PHP, COBOL, PL/SQL, etc.
- Under the GNU Lesser General Public License, SonarQube is freely available.
- SonarQube is affiliated with some important external tools like GitHub, Active Directory, LDAP, and others.
- SonarQube merged with Visual Studio, Eclipse, and IntelliJ IDEA development environments due to the **SonarLint** plug-ins.

JMeter

JMeter is an open-source tool that is used to test the performance of both static and dynamic resources and dynamic web applications.

It is completely designed on the JAVA application to load the functional test behavior and measure the application's performance.

It facilitates users or developers to use the source code for the development of other applications.



Features of JMeter

Below are some of the essential characteristics of JMeter:

- It is platform-independent, which accepts a JVM like **Windows, Mac, and Linux, etc.**
- It supports a user-friendly GUI, which is interactive and straightforward.
- It is incredibly extensible to load the performance test in multiple types of servers.

For more information about JMeter, refer to the below link:

<https://www.javatpoint.com/jmeter-tutorial>.

Bugzilla

Another bug tracking tool used in manual testing is **Bugzilla**.

It is most widely used by many organizations to track the various bugs of the application.

Bugzilla is an open-source tool that helps the customer and the client to keep track of the defects. Bugzilla is also considered a test management tool because in this, we can easily link other test case management tools such as ALM, Quality Centre, etc.



Features of Bugzilla

Bugzilla has some additional features which help us to report the bug easily:

- It supports various operating systems such as Windows, Linux, and Mac.
- With the help of Bugzilla, we can list a bug in several formats.
- User preferences can measure email notification.
- Bugzilla has advanced searching capabilities.

Mantis

Mantis is a web-based bug tracking system. MantisBT stands for **Mantis Bug Tracker**. It is used to follow the software defects and performed in the PHP programming language. It is also an open-source tool.



Features of Mantis

Some of the standard features of the particular tool are as follows:

- With the help of this tool, we have full-text search accessibility.
- Audit trails of changes made to issues.
- It provides the revision control system integration.
- Revision control of text fields and notes

To get more details about bug tracking tools, refer to the following link: <https://www.javatpoint.com/defect-or-bug-tracking-tool>.

Tessy

Another integration testing tool is **Tessy**, which is used to perform the integration and unit testing for the embedded software. It also helps us to discover the code coverage of the software or an application.

It can easily manage the entire test organization, including business needs, test management, coverage quantity, and traceability.

Tessy contains three primary functions, which are as follows:

- Test Interface Editor (TIE)
- Test Data Editor (TDE)
- Workspace.



Features of TESSY

The standard features of the TESSY are as follows:

- It produces the test report for the test execution results.
- It supports various programming languages such as C and C++.
- Tessy is used to evaluate the interface of the function and describes the variable used by that function.

For more information about integration testing tools, refers to the following link:
<https://www.javatpoint.com/integration-testing-tools>.

Overview

In this article, we have seen detailed information about **Manual testing, which includes the definition of manual testing, the need of manual testing, type of manual testing, manual testing tools, the process of manual testing, and some important benefits and drawbacks of it.**

Finally, we can say that, it is a process where the test engineer needs to be very persistent, innovative, and responsive.

In manual testing, the test engineer needs to think and perform like end-user interpretation.

In order to implement manual testing, a test engineer needs productive skill and imagination. And they need to think of multiple situations or scenarios to test a specific application.

Even though we can test nearly all applications with the help of automation testing at present, still manual testing is necessary as it is the base of software testing.

Automation Testing

In the earlier article on **software testing**, we learned that software testing is classified into two types of testing, which are **Manual testing** and **Automation testing**. Both manual and automation testing have their characteristics and approaches that make both the testing technique different from each other.

Here, we are going to learn about the following related topics of automation testing:

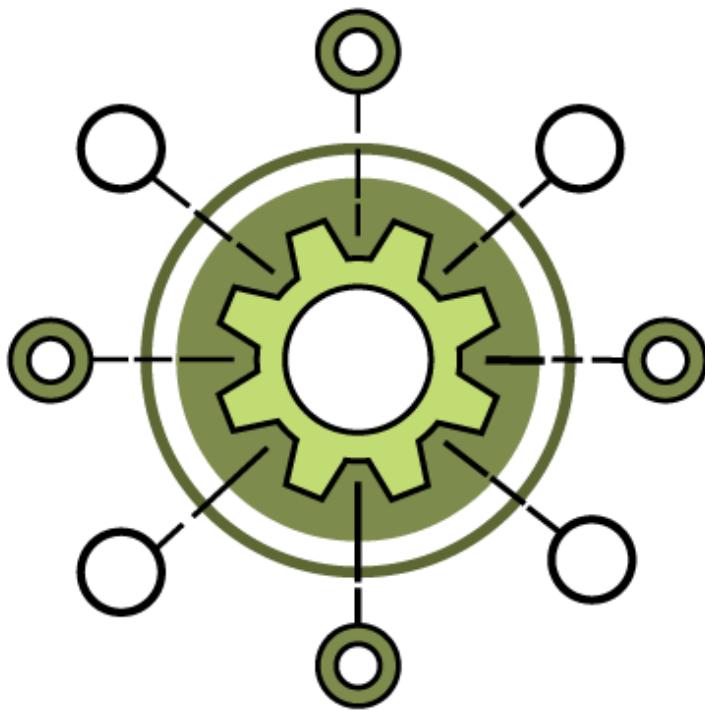
- **Introduction to Automation testing**
- **Why do we need to perform automation testing?**
- **Different approaches used in automation testing**
- **Automation testing process**
- **What are different challenges faced during the automation testing process?**
- **Automation testing tools**
- **Benefits and Drawbacks of automation testing.**

Introduction to Automation Testing

Another **software testing** method is **automation testing**, which is used some specific tools to execute the test scripts without any human interference. It is the most acceptable way to enhance the efficiency, productivity, and test coverage of Software testing.

With the help of an **automation testing tool**, we can easily approach the test data, handle the test implementation, and compares the actual output against the expected outcome.

Automation Testing



In **automation testing**, the test automation engineer will write the test script or use the automation testing tools to execute the application. On the other hand, in manual testing, the test engineer will write the test cases and implement the software on the basis of written test cases.

In test automation, the **test engineer** can execute repetitive tasks and other related tasks. In manual testing, it is a tedious process to implement the repetitive task again and again.

In other words, we can say that the main concentration of **test Automation** is to change the manual human activity with systems or devices.

The **automation testing** process is a time-saving process as it spends less time in exploratory testing and more time in keeping the test scripts whereas enhancing the complete test coverage.

Note: We perform exploratory testing whenever the requirement does not exist.

Why do we need to perform automation testing?

- In software testing, automation testing is required to test the application because it offers us a better application with less effort and time.
- Some organizations still perform only manual testing to test the application as those companies are not fully aware of the automation testing process.
- But now, they are aware of automated testing and executing the test automation procedure in their application development process.
- To implement the automation testing, we required pretty a considerable investment of resources and money.

The execution of automation testing provides us various advantages, which are as discussed below:

- **Reusability**
- **Consistency**
- **Running tests anytime (24/7)**
- **Early Bug detection**
- **Less Human Resources**



1. Reusability

We can re-use the test scripts in automation testing, and we don't need to write the new test scripts again and again. And, we can also re-create the steps which are detailed as the earlier ones.

2. Consistency

As compared to manual testing, automation testing is more consistent and way faster than executing the regular monotonous tests that cannot be missed but may cause faults when tested manually.

3. Running Tests 24/7

In automation testing, we can start the testing process from anywhere in the world and anytime we want to. And even we can do that remotely if we don't have many approaches or the option to purchase them.

4. Early Bug Detection

We can easily detect the critical bugs in the software development process's initial phases by executing automation testing. It also helps us spend fewer working hours to fix these problems and reduce costs.

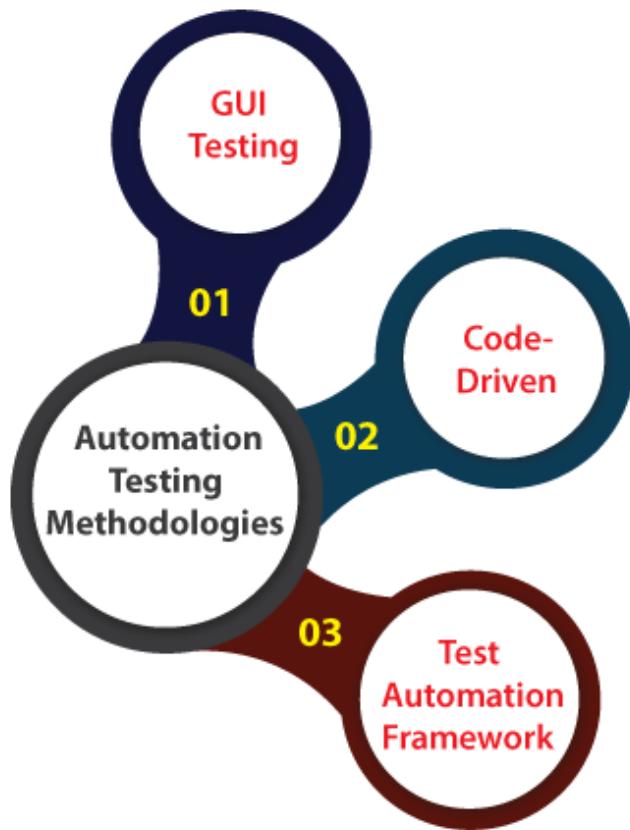
5. Less Human Resources

To implement the automation test script, we need a test automation engineer who can write the test scripts to automate our tests, rather than having several people who are repeatedly performing the tedious manual tests.

Automation Testing Methodologies

Automation testing contains the following three different methodologies and approaches, which will help the test engineer to enhance the software product's quality.

- **GUI Testing**
- **Code-Driven**
- **Test Automation Framework**



Now, let's understand the different approaches of automation testing one by one:

1. GUI (Graphical user interface) Testing

In this approach, we can implement that software or an application, which contains GUIs. So, that the automation test engineers can record user actions and evaluate them many times.

We know that the **Test cases** can be written in several programming languages like **JAVA, C#, Python, Perl**, etc.

2. Code-Driven

The code-driven technique is the subsequent methodology used in automation testing. In this method, the test engineer will mainly concentrate on test case execution in order to identify whether the several parts of code are performing according to the given requirement or not.

Hence, it is very a commonly used method in **agile** software development.

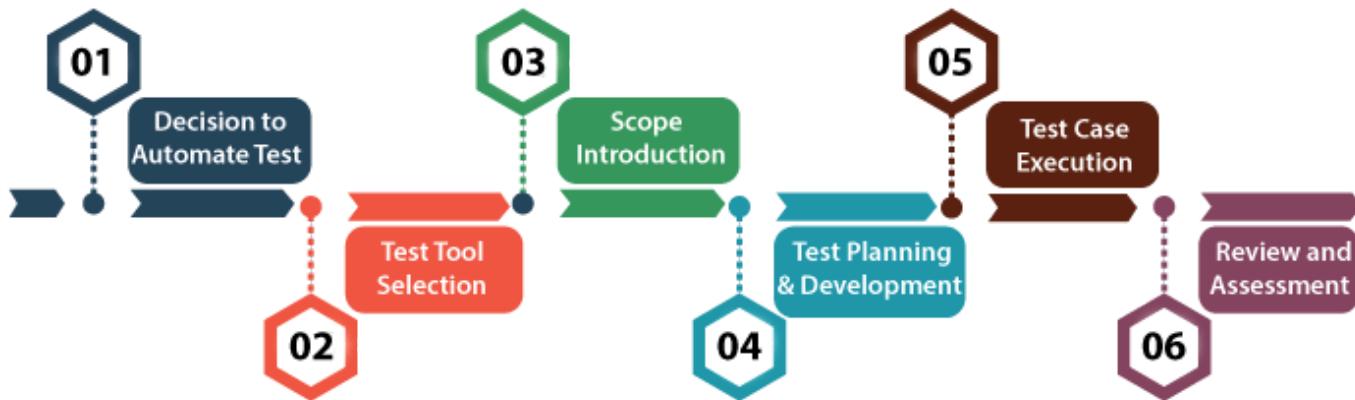
3. Test Automation Framework

Another approach in automation testing is **test automation framework**. The test automation framework is a set of rules used to generate valuable results of the automated testing activity.

Similarly, it brings together test data sources, function libraries, object details, and other reusable modules.

Automation Testing Process

The automation testing process is a systematic approach to organize and execute testing activities in a manner that provides maximum test coverage with limited resources. The structure of the test involves a multi-step process that supports the required, detailed and inter-related activities to perform the task.



The automation testing process completed in the following steps:

Step1: Decision to Automation Testing

It is the first phase of **Automation Test Life-cycle Methodology (ATLM)**. At this phase, the main focus of the testing team is to manage expectations from the test and find out the potential benefits if applying the automated testing correctly.

On adopting an automated testing suit, organizations have to face many issues, some are listed below:

- Testing tool experts are required for automation testing, so the first issue, to appoint a testing equipment specialist.
- The second issue is, choose the exact tool for the testing of a particular function.
- The issue of design and development standards in the implementation of an automated testing process.
- Analysis of various automated testing tools to choose the best tool for automation testing.
- The issue of money and time occurs as the consumption of money and time is high in the beginning of the testing.

Step2: Test Tool Selection

Test Tool Selection represents the second phase of the **Automation Test Life-cycle Methodology (ATLM)**. This phase guides the tester in the evaluation and selection of the testing tool.

Since the testing tool supports almost all testing requirements, the tester still needs to review the system engineering environment and other organizational needs and then make a list of evaluation parameters of the tools. Test engineers evaluate the equipment based on the provided sample criteria.

Step3: Scope Introduction

This phase represents the third phase of **Automation Test Life-cycle Methodology (ATLM)**. The scope of automation includes the testing area of the application. The determination of scope is based on the following points:

- Common functionalities of the software application that are held by every software application.
- Automation test sets the reusable range of business components.

- Automation Testing decides the extent of reusability of the business components.
- An application should have business-specific features and must be technically feasible.
- Automation testing provides the repetition of test cases in the case of cross-browser testing.

This phase ensures the overall testing strategy that should be well managed and modified if required. In order to ensure the availability of skills, testing skills of a particular member and whole team are analyzed against the required specific skills for a particular software application.

Step4: Test Planning and Development

Test planning and development is the fourth and most important phase of Automation Test Life -cycle Methodology (ATLM) because all the testing strategies are defined here. Planning of long -lead test activities, the creation of standards and guidelines, an arrangement of the required combination of hardware, software and network to create a test environment, defect tracking procedure, guidelines to control test configuration and environment all are identified in this phase. Tester determines estimated effort and cost for the entire project. Test strategy and effort estimation documents are the deliverables provided by this phase. Test case execution can be started after the successful completion of test planning.

Step5: Test Case Execution

Test case Execution is the sixth phase of Automation Test Life -cycle Methodology (ATLM). It takes place after the successful completion of test planning. At this stage, the testing team defines test design and development. Now, test cases can be executed under product testing. In this phase, the testing team starts case development and execution activity by using automated tools. The prepared test cases are reviewed by peer members of the testing team or quality assurance leaders.

During the execution of test procedures, the testing team directed to comply with the execution schedule. Execution phase implements the strategies such as integration, acceptance and unit testing that have defined in the test plan previously.

Step6: Review and Assessment

Review and assessment is the sixth and final stage of the automated testing life cycle but the activities of this phase are conducted throughout the whole life cycle to maintain continuous quality improvement. The improvement process is done via the evaluation of matrices, review and assessment of the activities.

During the review, the examiner concentrates whether the particular metric satisfies the acceptance criteria or not, if yes, then it is ready to use in software production. It is comprehensive as test cases cover each feature of the application.

The test team performs its own survey to inquire about the potential value of the process; if the potential benefit is less than sufficient, the testing team can change the testing tool. The team also provides a sample survey form to ask for feedback from the end user about the attributes and management of the software product.

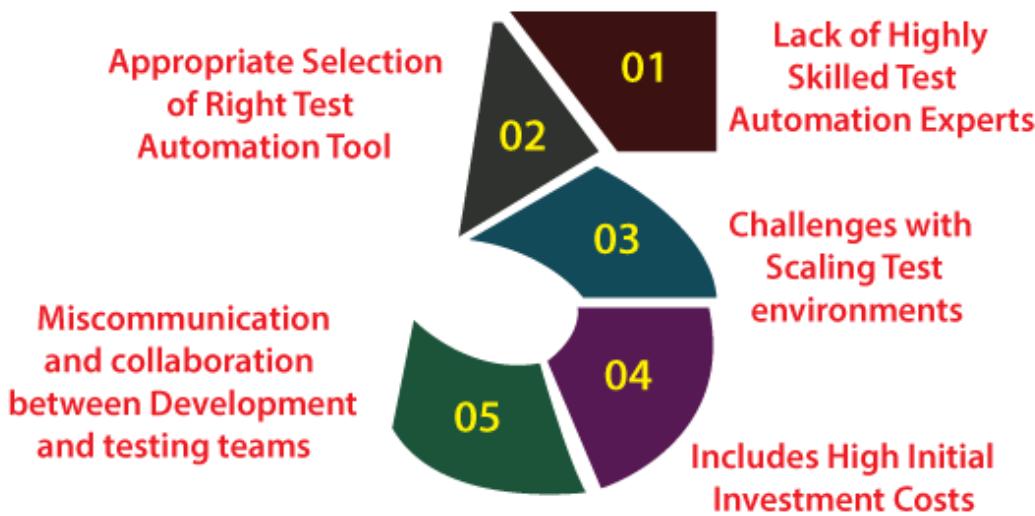
What are different challenges faced during the Automation testing process, and how to overcome them?

The **automation testing process** carries about large numbers of advantages for an organization. Usually, with automated testing, the testing team of the specified method of software validation can accomplish the amplified test coverage.

Though we may face various challenges during the automation testing process; therefore, we need a thorough follow-up process to reach successful test automation execution.

Some of the common challenges are as discuss below, which we might encounter throughout the test automation process of an application:

Different Challenges during the Automation Testing Process?



1. Lack of Highly Skilled Test Automation Experts

- The most fundamental reason behind lacking an automation testing process is the absence of highly skilled automation test engineers.
- To design and maintain the test automation framework and test scripts, the test automation engineer needs to be skilled and have details knowledge about automation technical skills.
- To create a test script correctly and sustain them, the team should implement the test scripts successfully to check the application performance and fix the technical issues.
- To achieve the operative automation aspects, the automation test engineers should have strong knowledge about test automation frameworks because it would be easy for them to design and execute the test scripts.

2. Appropriate Selection of Right Test Automation Tool

- The next challenge, we mostly face while performing the automation testing is selecting the **correct automation testing tool**.
- A suitable detailed assessment of the application under test (AUT) is mandatory.
- With the accessibility of various automation testing tools for both open-source and paid tools. And it is essential to check appropriately before accepting a tool.

3. Challenges with Scaling Test environments

- It is a crucial challenge in present day as testing teams that don't providing test environments in the cloud.
- It is essential to quickly deliver the changed test environments that automated testing requires, measure them up, execute the tests, to make sure that the success while testing in the cloud.
- Generally, the internal teams have a partial number of test environments which they can use to execute the fewer tests and deployed them at any given time.
- As a result, testing takes a much longer time. To overcome this, it is essential to **move test environments to the cloud to scale the test automation**, and it will also reduce testing teams operating costs.

4. Including High Initial Investment Costs

- Generally, the execution of the automation testing process in the initial phase is a bit expensive because it is necessary to evaluate, design, and build an automation testing framework, libraries, or reusable functions over a detailed AUT analysis.

- Additionally, if an automation test engineer picks the licensed test automation tool, the operating cost should be appraised.
- If the automation test engineer selects the open-source tools, then it should also be significant efforts spent on learning, training, and maintaining them.
- Later, the selection of automation tools, either open-source or licensed, depends on the AUT priority and availability of cost facility.

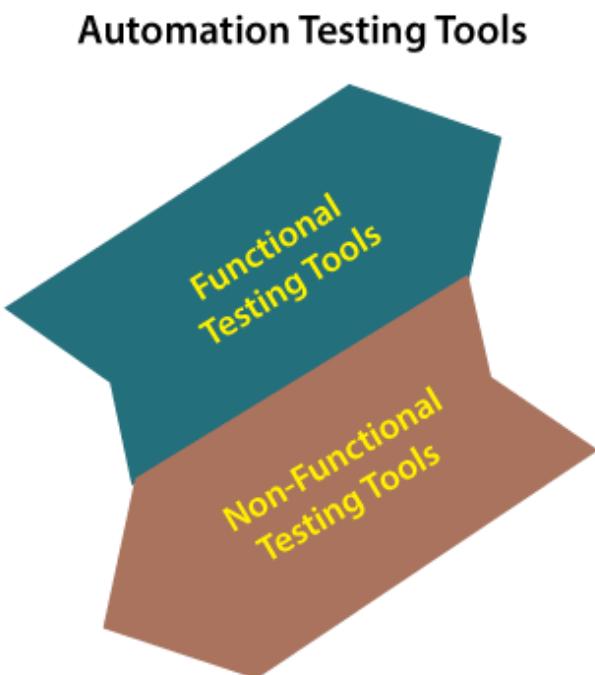
5. Miscommunication and lack of collaboration between Development and testing teams

- Miscommunication or lack of proper collaboration between the developer and the test engineers' team and operations teams does not lead us to successful test automation.
- Therefore, it is significant to place all the team (developer, testing, operation) members to categorize the test automation objectives and set targets.
- To ensure the test automation success, these teams need to spend some time to get an effective communication and to get a clear understanding of the business needs and project specifications.

Automation Testing Tools

Automation testing tools can describe in two categories, which are as follows:

- **Functional Testing Tools**
- **Non-Functional Testing Tools**



Let's understand them one by one in-details:

Functional Automation Testing Tools

The automation test engineer uses the functional automation testing tool to implement the functional **test cases**. **For example**, the Repetitive Regression tests are automated under the function automation testing tools.

This type of tools can be further divided into two different parts, which are as below:

- **Commercial Tool**
- **Open-source Tool**

Functional Automation Testing Tools

01 Commercial Tool

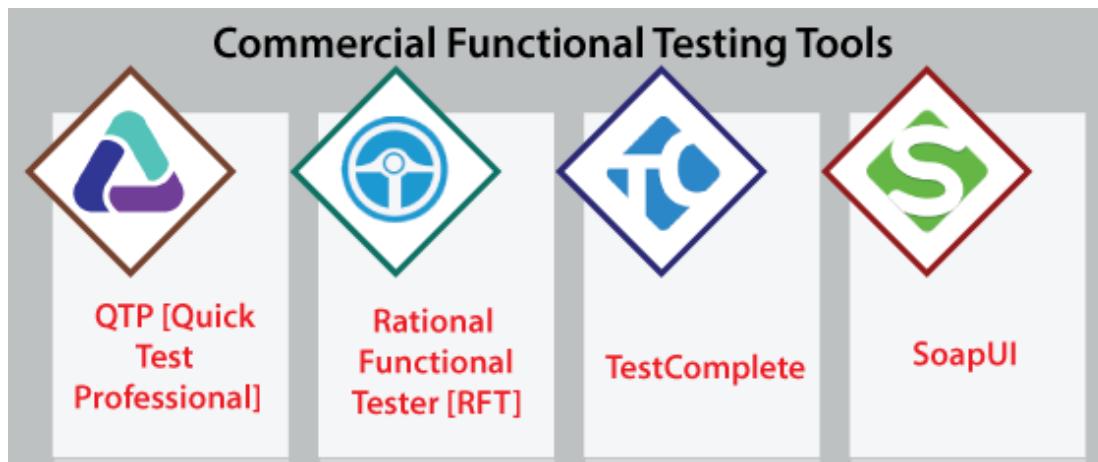
02 Open-source Tool

Commercial Functional Testing Tools

Commercial **functional testing** tools are those testing tools, which are not available freely in the market. These tools are also known as **licensed tools**. The licensed tools include various features and approaches as compared to the open-source tools.

Some of the most important commercial tools are as follows:

- **QTP[Quick Test Professional]**
- **Rational Functional Tester [RFT]**
- **TestComplete**
- **SoapUI**



Let's take a look on the above listed commercial tools:

QTP

- QTP signify as **Quick Test Professional**, but now it is known as Micro Focus **UFT (Unified Functional Testing)**.
- Mainly, it is used to automate the functional regression test cases of the web-based application or the software.
- In order to test the application, deploy the objects, and for analysis purposes, it is designed on the scripting languages such as **VBScript**.
- It follows the concept of **keyword-driven testing** for defining the test creation and maintenance.
- The new test engineers can easily use the QTP tool because it allows them to develop the test cases right from the application.



Characteristics of QTP

Some of the most standard attributes of QTP are as follows:

- The record and playback feature enhance the credibility of the QTP tool.
- Both technical and non-technical test engineers can operate the QTP tool.
- It supports several software development environments such as **SAP, JAVA, Oracle**, and so on.
- We can check both web-based and desktop-based applications by using the QTP.

RFT

- RFT stands for **Rational Functional Tester**, which is used to execute the functional regression test cases.
- It follows the concept of Data-driven testing and GUI testing.
- It builds the automated functional tests by recording an end-user's activities on the system under test and repeating the actions on request to perform a test.



Features of RFT

The RFT tool contains the following characteristics:

- RFT supports an extensive range of applications and protocols, like **HTML, Java, .NET, Windows, Eclipse, SAP, Visual Basic, PowerBuilder**
- It sustains custom controls by proxy SDK like Java or .Net.
- It allows developers to develop the keyword-associated script; hence, it can be re-used again, which enhances the productivity of the project.

TestComplete

- Another functional automated testing tool is TestComplete, which acquires by **SmartBear Software**.
- TestComplete has some inbuild capability, which helps the test engineers to develop the automated tests for Microsoft Windows, Web, Android, and iOS applications.
- In this tool, the tests can be recorded, scripted, or manually created with keyword-driven processes and used for automatic playback and error logging.
- It includes the three different modules, **like Web, Desktop, and Mobile**, where all the modules involve the functionality to develop the automated tests on the particular platform.



Characteristics of TestComplete Tool

Below is the essential characteristic of the TestComplete Tool:

- It is used to test several types of specific applications such as **Web, Windows, Android, iOS, .NET, VCL, and Java**.
- It involves the issue-tracking templates, which can be used to develop or change items stored in issue-tracking systems.
- It records the key actions essentially to repeat the test and rejects all unnecessary efforts.

SoapUI

- SoapUI is the most extensively used automation tool. It is mainly used to test the **web services** and **web APIs of SOAP and REST interfaces**.
- It allows the test engineers to test functional, regression testing, and other testing types on various Web services and APIs.
- SoapUI tool develops the mocks where test engineers can test real applications.
- It is entirely written on **JAVA** and Groovy programming languages.



Features of SoapUI

Some of the most common characteristics of the SoapUI tool are as follows:

- The core features of SoapUI contains various web services, such as **development, mocking, Inspection, simulation, and invoking**.
- It supports all standard protocols and technologies like **HTTP, HTTPS, AMF, JDBC, SOAP, WSDL, etc.**
- SoapUI tool offers a fast and efficient framework which creates many web service tests.
- It provides the services to obtain data from several sources of web service without developing any code.

To get more information about the SoapUI tool, refers to the following link: <https://www.javatpoint.com/soapui>.

Open-source Functional Testing Tools

Open-source functional testing tools are those tools, which are available freely in the market. These tools have less functionality and features than the **commercial/licensed tools**, but sometimes working on the commercial tool became an expensive process.

That's why some well-known organizations preferred to use open-source tools.

Following are the most commonly used open-source functional automation testing tools:

- **Selenium**
- **Sikuli**
- **Autoit**

Let's understand the open-source functional automation testing tools one by one:

Open-source Functional Testing Tools



Selenium

Whenever we are talking about the open-source tools of automation testing, one name came into every automation test engineer's mind: Selenium.

- Selenium is the **highly recommended** and widely used functional testing tool, which is well-suited for non-functional testing tools.
- Selenium is an open-source tool which means, it does not require any licensing.
- We can only test the web application using the selenium tool, and the Stand-alone application cannot automate in Selenium.
- It is most commonly used to implement Functional test scripts.
- It can be combined with several automation testing tools to achieve continuous testing, for example, **Maven and Jenkins**.
- It can be associated with many devices such as **TestNG and JUnit** to manage the test cases and generate the test reports.



Characteristics of Selenium

Following are the basic features of selenium:

- Selenium can be easily deployed on several platforms like **Windows, Linux, Solaris, and MacOS**.
- Furthermore, it also supports operating systems for mobile applications, **for example, iOS, Windows mobile, and android**.
- Selenium supports several programming languages such as **C#, Java, Perl, PHP, Python, and Ruby**.

- Selenium became extremely resourceful while we were implementing the test script and analyzing it across many browsers concurrently.
- It provides a user-friendly interface that helps the test engineer develop and perform the test scripts resourcefully and efficiently.

To get the complete information about Selenium, refers to the following link: <https://www.javatpoint.com/selenium-tutorial>.

Sikuli

- Another open-source functional automation testing tool is **Sikuli**.
- It is a GUI based test automation tool, which can easily automate the Flash objects as most of the automation testing tool like selenium does not support the flash-object automation.
- Most commonly Sikuli is used to interact with fundamentals of web pages and control the windows-based popups.
- With the help of Sikuli tool, we can easily test the windows application.



Features of Sikuli

The most standard features of the Sikuli tool are as discussed below:

- The Sikuli tool can be easily combined with Selenium WebDriver and all other automation testing tools by using the Sikuli Jar files.
- We can also automate desktop applications or stand-alone applications with the help of the Sikuli tool.
- It provides a simple API, which implies that all methods can be retrieved using screen class objects.

Note: As we know that Selenium WebDriver can only test the web application, but on the other hand, the Sikuli tool can test the web-based as well as window-based applications.

AutoIT

- **AutoIT** is another open-source tool used for functional automation testing.
- It is a freeware scripting language designed to test the Windows GUI and general scripting.
- The AutoIT script is written in a primary language.
- It can replicate any grouping of keystrokes, mouse movement, and window or control operation.



Features of AutoIT

AutoIT tools include the following features:

- It uses a straightforward syntax for its scripting language that can be easily understood and executed to automate any process.
- It has an inbuilt **RunAs** function that helps in executing any external program using some outside user.
- It can even record and create a script on its own for detailed processes, which need to be automated.
- All types of standard Windows controls and other GUIs without any object identification issue can easily relate to the AutoIT tool.

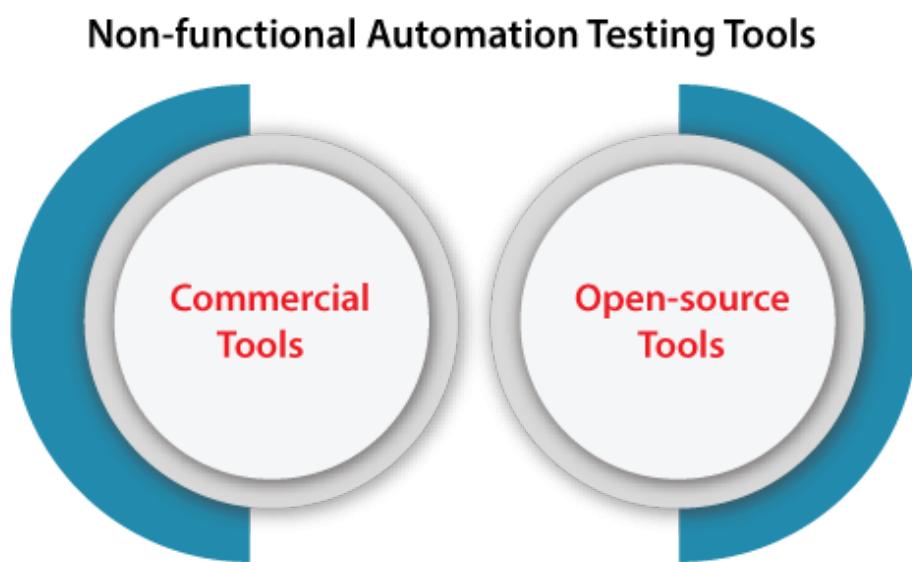
Non-functional Automation Testing Tools

The automation test engineer uses the non-functional automation testing tool to execute the non-functional performance **test cases**.

For example, testing the application's response time under significant load, let say, 100 users.

Just like functional automation testing tools, the non-functional automation testing tools divided into two different categories, which are as below:

1. **Commercial Tools**
2. **Open-source Tools**



Commercial Non-Functional Automation Testing Tools

These are those tools, which cannot be used freely, as they required a proper license. The commercial tools have additional features and functionality as compared to the other open-source testing tools.

The usage of these types of tools helps us to enhance the efficiency of the software product.

Let's see some of the most commonly used commercial non-functional automation testing tools.

- **LoadRunner**
- **Silk Performer**

LoadRunner [HP Performance Tester]

LoadRunner is one of the most popular **non-functional** tools. It is mainly used to support performance testing for a wide range of protocols, several technologies, and application environments. LoadRunner is the licensed tool.

It rapidly classifies the maximum number of performance issues. And precisely predict the application scalability and size.



Feature of LoadRunner

Some of the most significant features of the LoadRunner tool are as follows:

- LoadRunner tool will help us to reduce the cost of distributed load testing and hardware and software costs.
- We can easily view and handle the XML data within the test scripts as it supports the XML scripting language.
- We can get detailed performance test reports by using the LoadRunner tool.
- It offers the operational tool for deployment tracking.

Silk Performer

Another non-functional automation testing tool is Silk Performer. It can test the various application environments with thousands of simultaneous users.

It makes sure that applications and server up times are sustained when faced with the highest customer usage.

It is one of the most commonly used enterprise-class load and stress testing tools, supporting an extensive range of protocols.



Characteristic of Silk Performer

Following are the standard features of the Silk performer tool:

- It is used to pretends changeable virtual users.
- In this tool, the correlation and parameterization are user-friendly.
- It supports integrated server monitoring.
- It provides Version Control Management.
- There is no license condition required for controllers or Individual Protocols.
- It can quickly create reports with tables and graphs and allows customization.

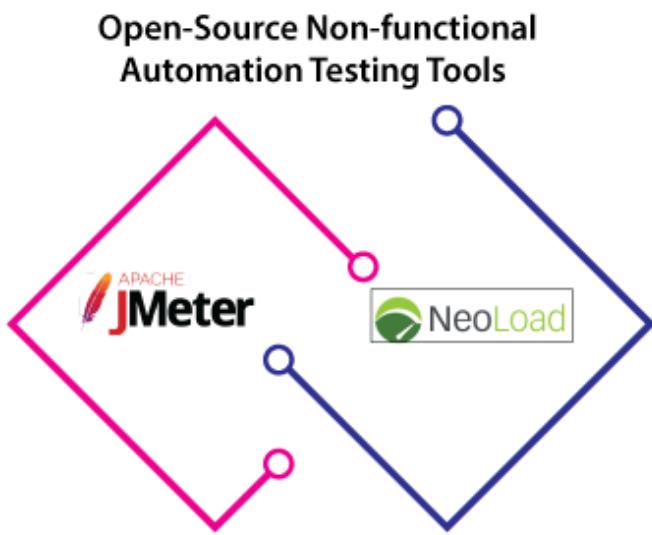
Open-Source Non-functional Automation Testing Tools

The open-source non-functional automation testing tools can be used easily as they are freely available in the market. These tools have less functionality as compared to the commercial testing tools.

But because they are open-source and used quickly, some organizations prefer to use these tools as they do not require any cost.

Some of the most commonly used open-source non-functional automation testing tools are as follows:

- **JMeter**
- **NeoLoad**



JMeter

JMeter is one of the most significantly used open-source non-functional automation testing tools. JMeter is entirely designed on the JAVA application to load the efficient test performance and measure an application's productivity.

It is mainly simplifying users or developers to use the source code for the enhancement of other applications. It is used to test the implementation of both static and dynamic resources and dynamic web applications.

It is used to reproduce the massive load on a server, object, or network, group of servers to test its power or explore the complete performance in multiple load types.

It is highly extensible to load the performance test in several types of servers like:

- **Database Server: LDAP, JDBC**
- **Web Server: SOAP, HTTPS, HTTP**
- **Mail Server: POP3**



Characteristics of JMeter

Some of the significant attributes of the JMeter tool are as follows:

- It sustains several testing approaches, **for example**, functional, distributed, and load testing.
- It supports a user-friendly GUI that is interactive and easy.
- It is a platform-independent tool as it is developed with the help of JAVA; therefore, it can execute on any platform that accepts a JVM like **Windows, Mac, and Linux, etc.**

For more information about JMeter, refers to the below link: <https://www.javatpoint.com/jmeter-tutorial>.

NeoLoad

Another most commonly used open-source tool in automation testing is NeoLoad and **Neotys** develop it.

It is used to test the performance test scenarios and also helps us to identify the bottleneck areas in the web and the mobile application development process.

It is faster as compared to other traditional tools. NeoLoad will support a wide range of web, mobile, and packaged applications, such as **SAP**, **Oracle**, **Salesforce**, etc., that cover all our testing needs.



Features of NeoLoad

Below are some of the vital features of NeoLoad:

- It is also used to share and manage the test resources.
- It will support various frameworks and protocols such as HTTP/2, HTML5, API, AngularJS, Web Socket, SOAP, etc.
- It provides a robust code-less design.
- NeoLoad can modify the functional test script into the performance test scripts.
- NeoLoad can automatically update the test scripts and developed the real-time test results.

Advantages of Automation Testing

- Automation testing takes less time than manual testing.
- A tester can test the response of the software if the execution of the same operation is repeated several times.
- Automation Testing provides re-usability of test cases on testing of different versions of the same software.
- Automation testing is reliable as it eliminates hidden errors by executing test cases again in the same way.
- Automation Testing is comprehensive as test cases cover each and every feature of the application.
- It does not require many human resources, instead of writing test cases and testing them manually, they need an automation testing engineer to run them.
- The cost of automation testing is less than manual testing because it requires a few human resources.

Disadvantages of Automation Testing

- Automation Testing requires high-level skilled testers.
- It requires high-quality testing tools.
- When it encounters an unsuccessful test case, the analysis of the whole event is complicated.
- Test maintenance is expensive because high fee license testing equipment is necessary.
- Debugging is mandatory if a less effective error has not been solved, it can lead to fatal results.

Overview

In this tutorial, we have understood the automation testing, automation testing approaches, automation testing process, automation testing tools, different challenges during the automation testing process, advantages and drawbacks of automation testing.

Finally, we can conclude that the **Automation testing** is a **software testing** technique, which implemented with the help of special automated testing software tools.

It is the best approach to execute a test case suite, which helps us enhance test coverage, efficiency, and performance speed in software testing.

The selection of automation testing tool, testing process, and team, are essential aspects in order to successful automation.

Automation testing is highly dependent on the technology the Application Under Test is built on.

Test Automation Maintenance Approach is an automation testing phase execute to test whether the new functionalities added to the software are working fine or not.

For the successful testing process of the software, the manual and automation techniques go hand-in-hand. We should clear about the automation process that it is used to decrease the testing time for certain types of tests.

White Box Testing

The box testing approach of software testing consists of black box testing and white box testing. We are discussing here white box testing which also known as glass box is **testing, structural testing, clear box testing, open box testing and transparent box testing**. It tests internal coding and infrastructure of a software focus on checking of predefined inputs against expected and desired outputs. It is based on inner workings of an application and revolves around internal structure testing. In this type of testing programming skills are required to design test cases. The primary goal of white box testing is to focus on the flow of inputs and outputs through the software and strengthening the security of the software.

The term 'white box' is used because of the internal perspective of the system. The clear box or white box or transparent box name denote the ability to see through the software's outer shell into its inner workings.

Developers do white box testing. In this, the developer will test every line of the code of the program. The developers perform the White-box testing and then send the application or the software to the testing team, where they will perform the **black box testing** and verify the application along with the requirements and identify the bugs and sends it to the developer.

The developer fixes the bugs and does one round of white box testing and sends it to the testing team. Here, fixing the bugs implies that the bug is deleted, and the particular feature is working fine on the application.

Here, the test engineers will not include in fixing the defects for the following reasons:

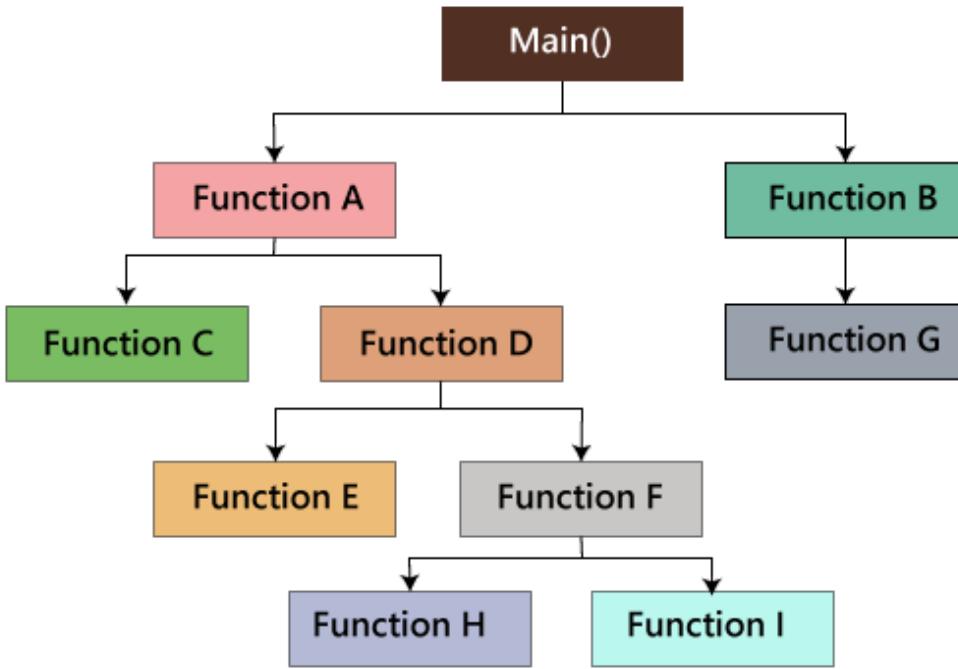
- Fixing the bug might interrupt the other features. Therefore, the test engineer should always find the bugs, and developers should still be doing the bug fixes.
- If the test engineers spend most of the time fixing the defects, then they may be unable to find the other bugs in the application.

The white box testing contains various tests, which are as follows:

- Path testing
- Loop testing
- Condition testing
- Testing based on the memory perspective
- Test performance of the program

Path testing

In the path testing, we will write the flow graphs and test all independent paths. Here writing the flow graph implies that flow graphs are representing the flow of the program and also show how every program is added with one another as we can see in the below image:



And test all the independent paths implies that suppose a path from main() to function G, first set the parameters and test if the program is correct in that particular path, and in the same way test all other paths and fix the bugs.

Loop testing

In the loop testing, we will test the loops such as while, for, and do-while, etc. and also check for ending condition if working correctly and if the size of the conditions is enough.

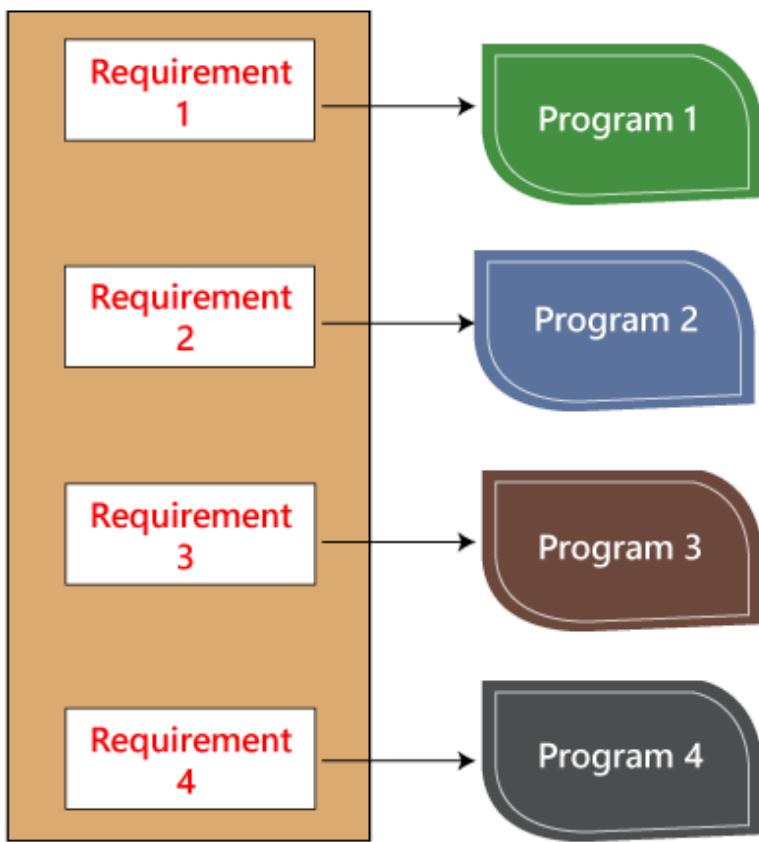
For example: we have one program where the developers have given about 50,000 loops.

```
{
while(50,000)
.....
.....
}
```

We cannot test this program manually for all the 50,000 loops cycle. So we write a small program that helps for all 50,000 cycles, as we can see in the below program, that test P is written in the similar language as the source code program, and this is known as a Unit test. And it is written by the developers only.

```
Test P
{
.....
..... }
```

As we can see in the below image that, we have various requirements such as 1, 2, 3, 4. And then, the developer writes the programs such as program 1,2,3,4 for the parallel conditions. Here the application contains the 100s line of codes.

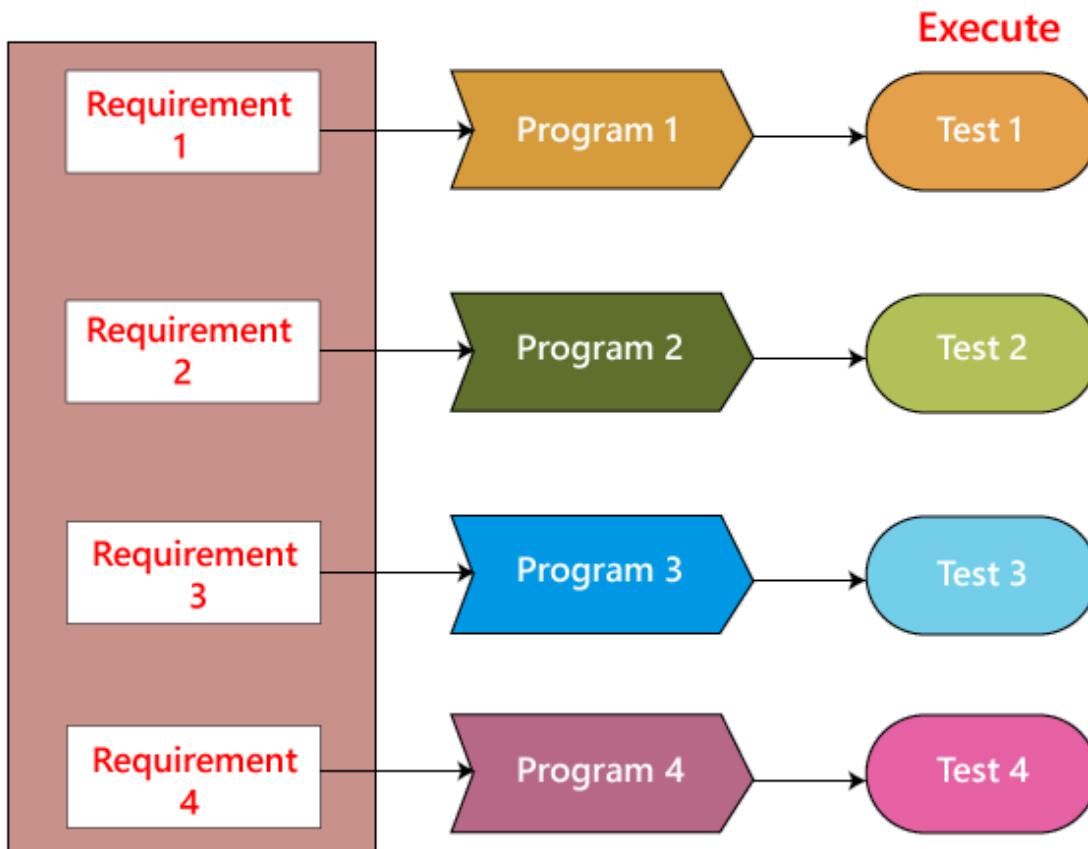


The developer will do the white box testing, and they will test all the five programs line by line of code to find the bug. If they found any bug in any of the programs, they will correct it. And they again have to test the system then this process contains lots of time and effort and slows down the product release time.

Now, suppose we have another case, where the clients want to modify the requirements, then the developer will do the required changes and test all four program again, which take lots of time and efforts.

These issues can be resolved in the following ways:

In this, we will write test for a similar program where the developer writes these test code in the related language as the source code. Then they execute these test code, which is also known as **unit test programs**. These test programs linked to the main program and implemented as programs.



Therefore, if there is any requirement of modification or bug in the code, then the developer makes the adjustment both in the main program and the test program and then executes the test program.

Condition testing

In this, we will test all logical conditions for both **true** and **false** values; that is, we will verify for both **if** and **else** condition.

For example:

```
if(condition) - true
{
.....
.....
.....
}
else - false
{
.....
.....
.....
}
```

The above program will work fine for both the conditions, which means that if the condition is accurate, and then else should be false and conversely.

Testing based on the memory (size) perspective

The size of the code is increasing for the following reasons:

- **The reuse of code is not there:** let us take one example, where we have four programs of the same application, and the first ten lines of the program are similar. We can write these ten lines as a discrete function, and it should be accessible by the above four programs as well. And also, if any bug is there, we can modify the line of code in the function rather than the entire code.
- The **developers use the logic** that might be modified. If one programmer writes code and the file size is up to 250kb, then another programmer could write a similar code using the different logic, and the file size is up to 100kb.
- The **developer declares so many functions and variables** that might never be used in any portion of the code. Therefore, the size of the program will increase.

For example,

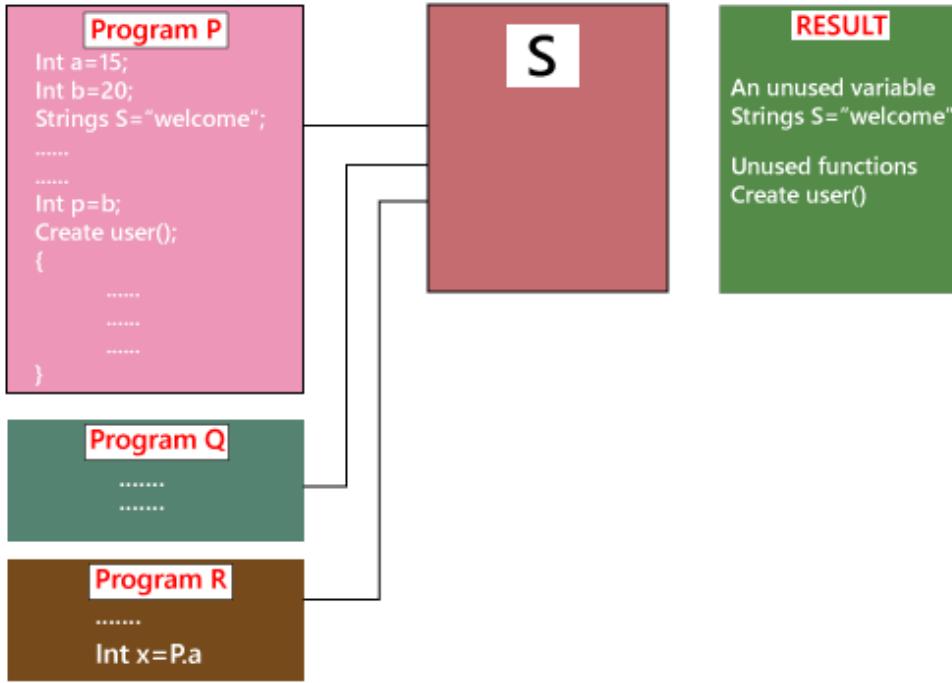
```
Int a=15;
Int b=20;
String S= "Welcome";
....
....
....
....
Int p=b;
Create user()
{
.....
.....
```

..... 200's line of code

}

In the above code, we can see that the **integer a** has never been called anywhere in the program, and also the function **Create user** has never been called anywhere in the code. Therefore, it leads us to memory consumption.

We cannot remember this type of mistake manually by verifying the code because of the large code. So, we have a built-in tool, which helps us to test the needless variables and functions. And, here we have the tool called **Rational purify**.



Suppose we have three programs such as Program P, Q, and R, which provides the input to S. And S goes into the programs and verifies the unused variables and then gives the outcome. After that, the developers will click on several results and call or remove the unnecessary function and the variables.

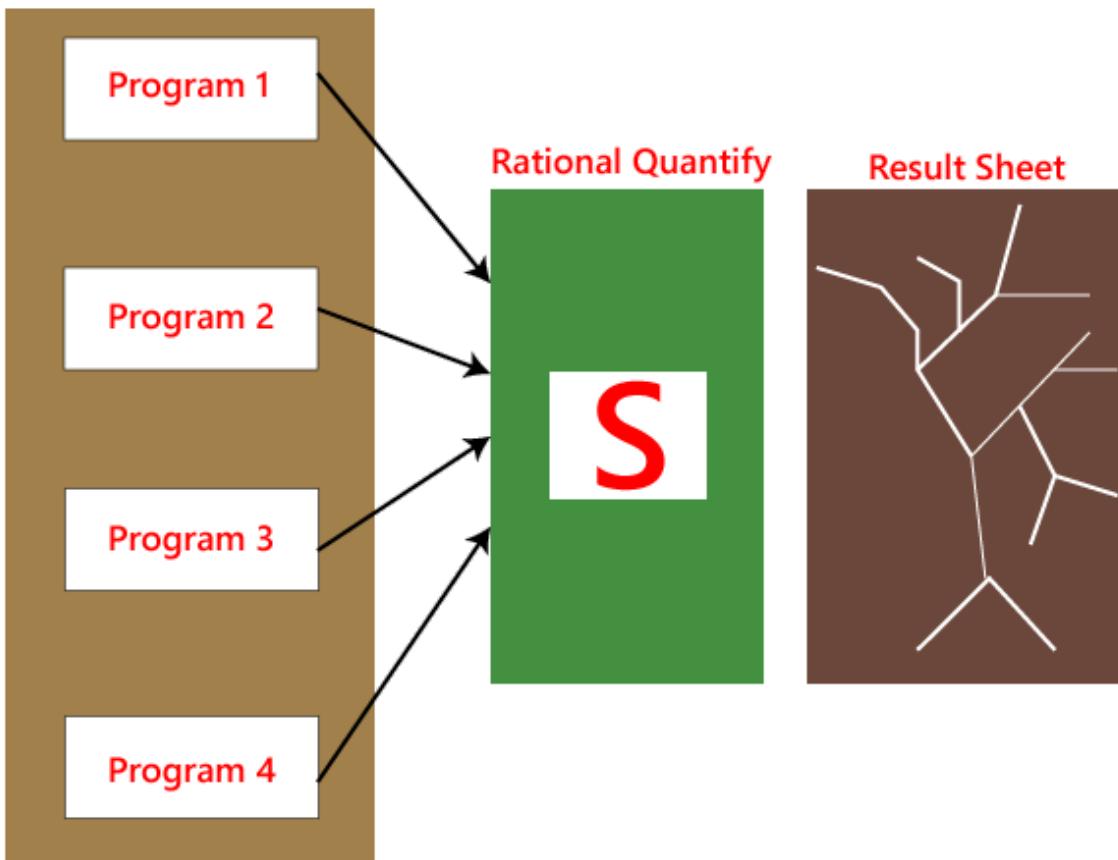
This tool is only used for the **C** programming language and **C++** programming language; for another language, we have other related tools available in the market.

- The developer does not use the available in-built functions; instead they write the full features using their logic. Therefore, it leads us to waste of time and also postpone the product releases.

Test the performance (Speed, response time) of the program

The application could be slow for the following reasons:

- When logic is used.
- For the conditional cases, we will use **or & and** adequately.
- Switch case, which means we cannot use **nested if**, instead of using a switch case.

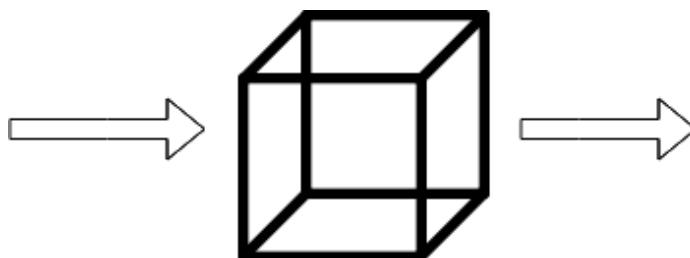


As we know that the developer is performing white box testing, they understand that the code is running slow, or the performance of the program is also getting deliberate. And the developer cannot go manually over the program and verify which line of the code is slowing the program.

To recover with this condition, we have a tool called **Rational Quantify**, which resolves these kinds of issues automatically. Once the entire code is ready, the rational quantify tool will go through the code and execute it. And we can see the outcome in the result sheet in the form of thick and thin lines.

Here, the thick line specifies which section of code is time-consuming. When we double-click on the thick line, the tool will take us to that line or piece of code automatically, which is also displayed in a different color. We can change that code and again and use this tool. When the order of lines is all thin, we know that the presentation of the program has enhanced. And the developers will perform the white box testing automatically because it saves time rather than performing manually.

Test cases for white box testing are derived from the design phase of the software development lifecycle. Data flow testing, control flow testing, path testing, branch testing, statement and decision coverage all these techniques used by white box testing as a guideline to create an error-free software.



Whitebox Testing

White box testing follows some working steps to make testing manageable and easy to understand what the next task to do. There are some basic steps to perform white box testing.

Generic steps of white box testing

- Design all test scenarios, test cases and prioritize them according to high priority number.
- This step involves the study of code at runtime to examine the resource utilization, not accessed areas of the code, time taken by various methods and operations and so on.

- In this step testing of internal subroutines takes place. Internal subroutines such as nonpublic methods, interfaces are able to handle all types of data appropriately or not.
- This step focuses on testing of control statements like loops and conditional statements to check the efficiency and accuracy for different data inputs.
- In the last step white box testing includes security testing to check all possible security loopholes by looking at how the code handles security.

Reasons for white box testing

- It identifies internal security holes.
- To check the way of input inside the code.
- Check the functionality of conditional loops.
- To test function, object, and statement at an individual level.

Advantages of White box testing

- White box testing optimizes code so hidden errors can be identified.
- Test cases of white box testing can be easily automated.
- This testing is more thorough than other testing approaches as it covers all code paths.
- It can be started in the SDLC phase even without GUI.

Disadvantages of White box testing

- White box testing is too much time consuming when it comes to large-scale programming applications.
- White box testing is much expensive and complex.
- It can lead to production error because it is not detailed by the developers.
- White box testing needs professional programmers who have a detailed knowledge and understanding of programming language and implementation.

Techniques Used in White Box Testing

Data Flow Testing	Data flow testing is a group of testing strategies that examines the control flow of programs in order to explore the sequence of variables according to the sequence of events.
Control Flow Testing	Control flow testing determines the execution order of statements or instructions of the program through a control structure. The control structure of a program is used to develop a test case for the program. In this technique, a particular part of a large program is selected by the tester to set the testing path. Test cases represented by the control graph of the program.
Branch Testing	Branch coverage technique is used to cover all branches of the control flow graph. It covers all the possible outcomes (true and false) of each condition of decision point at least once.
Statement Testing	Statement coverage technique is used to design white box test cases. This technique involves execution of all statements of the source code at least once. It is used to calculate the total number of executed statements in the source code, out of total statements present in the source code.
Decision Testing	This technique reports true and false outcomes of Boolean expressions. Whenever there is a possibility of two or more outcomes from the statements like do while statement, if statement and case statement (Control flow statements), it is considered as decision point because there are two outcomes either true or false.

Difference between white-box testing and black-box testing

Following are the significant differences between white box testing and black box testing:

White-box testing	Black box testing
The developers can perform white box testing.	The test engineers perform the black box testing.
To perform WBT, we should have an understanding of the programming languages.	To perform BBT, there is no need to have an understanding of the programming languages.
In this, we will look into the source code and test the logic of the code.	In this, we will verify the functionality of the application based on the requirement specification.
In this, the developer should know about the internal design of the code.	In this, there is no need to know about the internal design of the code.

Black box testing

Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding. The primary source of black box testing is a specification of requirements that is stated by the customer.

In this method, tester selects a function and gives input value to examine its functionality, and checks whether the function is giving expected output or not. If the function produces correct output, then it is passed in testing, otherwise failed. The test team reports the result to the development team and then tests the next function. After completing testing of all functions if there are severe problems, then it is given back to the development team for correction.



Generic steps of black box testing

- The black box test is based on the specification of requirements, so it is examined in the beginning.
- In the second step, the tester creates a positive test scenario and an adverse test scenario by selecting valid and invalid input values to check that the software is processing them correctly or incorrectly.
- In the third step, the tester develops various test cases such as decision table, all pairs test, equivalence division, error estimation, cause-effect graph, etc.
- The fourth phase includes the execution of all test cases.
- In the fifth step, the tester compares the expected output against the actual output.
- In the sixth and final step, if there is any flaw in the software, then it is cured and tested again.

Test procedure

The test procedure of black box testing is a kind of process in which the tester has specific knowledge about the software's work, and it develops test cases to check the accuracy of the software's functionality.

It does not require programming knowledge of the software. All test cases are designed by considering the input and output of a particular function. A tester knows about the definite output of a particular input, but not about how the result is arising. There are various techniques used in black box testing for testing like decision table technique, boundary value analysis technique, state transition, All-pair testing, cause-effect graph technique, equivalence partitioning technique, error guessing technique, use case technique and user story technique. All these techniques have been explained in detail within the tutorial.

Test cases

Test cases are created considering the specification of the requirements. These test cases are generally created from working descriptions of the software including requirements, design parameters, and other specifications. For the testing, the test designer selects both positive test scenario by taking valid input values and adverse test scenario by taking invalid input values to determine the correct output. Test cases are mainly designed for functional testing but can also be used for non-functional testing. Test cases are designed by the testing team, there is not any involvement of the development team of software.

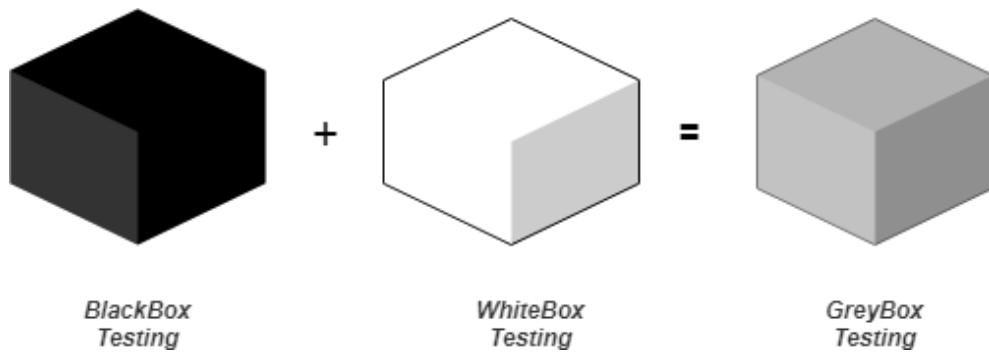
Techniques Used in Black Box Testing

Decision Table Technique	Decision Table Technique is a systematic approach where various input combinations and their respective system behavior are captured in a tabular form. It is appropriate for the functions that have a logical relationship between two and more than two inputs.
--------------------------	--

Boundary Value Technique	Boundary Value Technique is used to test boundary values, boundary values are those that contain the upper and lower limit of a variable. It tests, while entering boundary value whether the software is producing correct output or not.
State Transition Technique	State Transition Technique is used to capture the behavior of the software application when different input values are given to the same function. This applies to those types of applications that provide the specific number of attempts to access the application.
All-pair Testing Technique	All-pair testing Technique is used to test all the possible discrete combinations of values. This combinational method is used for testing the application that uses checkbox input, radio button input, list box, text box, etc.
Cause-Effect Technique	Cause-Effect Technique underlines the relationship between a given result and all the factors affecting the result. It is based on a collection of requirements.
Equivalence Partitioning Technique	Equivalence partitioning is a technique of software testing in which input data divided into partitions of valid and invalid values, and it is mandatory that all partitions must exhibit the same behavior.
Error Guessing Technique	Error guessing is a technique in which there is no specific method for identifying the error. It is based on the experience of the test analyst, where the tester uses the experience to guess the problematic areas of the software.
Use Case Technique	Use case Technique used to identify the test cases from the beginning to the end of the system as per the usage of the system. By using this technique, the test team creates a test scenario that can exercise the entire software based on the functionality of each function from start to end.

GreyBox Testing

Greybox testing is a software testing method to test the software application with partial knowledge of the internal working structure. It is a **combination of black box and white box testing** because it involves access to internal coding to design test cases as white box testing and testing practices are done at functionality level as black box testing.

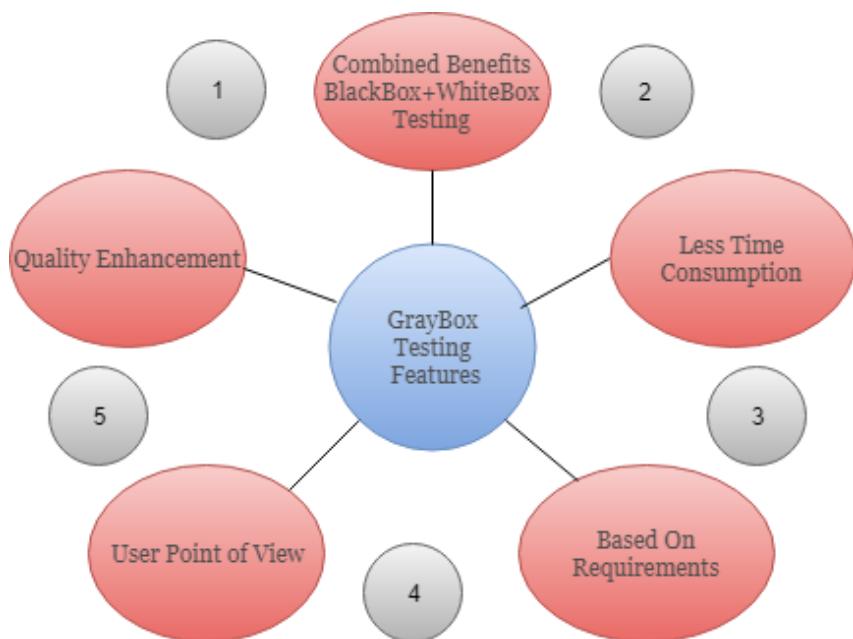


GreyBox testing commonly identifies context-specific errors that belong to web systems. For example; while testing, if tester encounters any defect then he makes changes in code to resolve the defect and then test it again in real time. It concentrates on all the layers of any complex software system to increase testing coverage. It gives the ability to test both presentation layer as well as internal coding structure. It is primarily used in integration testing and penetration testing.

Why GreyBox testing?

Reasons for GreyBox testing are as follows

- It provides combined benefits of both Blackbox testing and WhiteBox testing.
- It includes the input values of both developers and testers at the same time to improve the overall quality of the product.
- It reduces time consumption of long process of functional and non-functional testing.
- It gives sufficient time to the developer to fix the product defects.
- It includes user point of view rather than designer or tester point of view.
- It involves examination of requirements and determination of specifications by user point of view deeply.



GreyBox Testing Strategy

Grey box testing does not make necessary that the tester must design test cases from source code. To perform this testing test cases can be designed on the base of, knowledge of architectures, algorithm, internal states or other high -level descriptions of the program behavior. It uses all the straightforward techniques of black box testing for function testing. The test case generation is based on requirements and preset all the conditions before testing the program by assertion method.

Generic Steps to perform Grey box Testing are:

1. First, select and identify inputs from BlackBox and WhiteBox testing inputs.
2. Second, Identify expected outputs from these selected inputs.
3. Third, identify all the major paths to traverse through during the testing period.
4. The fourth task is to identify sub-functions which are the part of main functions to perform deep level testing.
5. The fifth task is to identify inputs for subfunctions.
6. The sixth task is to identify expected outputs for subfunctions.
7. The seventh task includes executing a test case for Subfunctions.
8. The eighth task includes verification of the correctness of result.

The test cases designed for Greybox testing includes Security related, Browser related, GUI related, Operational system related and Database related testing.

Techniques of Grey box Testing

Matrix Testing

This testing technique comes under Grey Box testing. It defines all the used variables of a particular program. In any program, variable are the elements through which values can travel inside the program. It should be as per requirement otherwise, it will reduce the readability of the program and speed of the software. Matrix technique is a method to remove unused and uninitialized variables by identifying used variables from the program.

Regression Testing

Regression testing is used to verify that modification in any part of software has not caused any adverse or unintended side effect in any other part of the software. During confirmation testing, any defect got fixed, and that part of software started working as intended, but there might be a possibility that fixed defect may have introduced a different defect somewhere else in the software. So, regression testing takes care of these type of defects by testing strategies like retest risky use cases, retest within a firewall, retest all, etc.

Orthogonal Array Testing or OAT

The purpose of this testing is to cover maximum code with minimum test cases. Test cases are designed in a way that can cover maximum code as well as GUI functions with a smaller number of test cases.

Pattern Testing

Pattern testing is applicable to such type of software that is developed by following the same pattern of previous software. In these type of software possibility to occur the same type of defects. Pattern testing determines reasons of the failure so they can be fixed in the next software.

Usually, automated software testing tools are used in Greybox methodology to conduct the test process. Stubs and module drivers provided to a tester to relieve from manually code generation.

Data Flow Testing

Data flow testing is used to analyze the flow of data in the program. It is the process of collecting information about how the variables flow the data in the program. It tries to obtain particular information of each particular point in the process.

Data flow testing is a group of testing strategies to examine the control flow of programs in order to explore the sequence of variables according to the sequence of events. It mainly focuses on the points at which values assigned to the variables and the point at which these values are used by concentrating on both points, data flow can be tested.

Data flow testing uses the control flow graph to detect illogical things that can interrupt the flow of data. Anomalies in the flow of data are detected at the time of associations between values and variables due to:

- If the variables are used without initialization.
- If the initialized variables are not used at least once.

Let's understand this with an example:

1. read x;	
2. If($x > 0$)	(1, (2, t), x), (1, (2, f), x)
3. a = x + 1	(1, 3, x)
4. if ($x \leq 0$) {	(1, (4, t), x), (1, (4, f), x)
5. if ($x < 1$)	(1, (5, t), x), (1, (5, f), x)
6. x = x + 1; (go to 5)	(1, 6, x)
else	
7. a = x + 1	(1, 7, x)
8. print a;	(6, (5, f)x), (6, (5, t)x) (6, 6, x) (3, 8, a), (7, 8, a).

In this code, we have a total 8 statements, and we will choose a path which covers all the 8 statements. As it is evident in the code, we cannot cover all the statements in a single path because if statement 2 is true then statements 4, 5, 6, 7 not covered, and if statement 4 is true then statement 2 and 3 are not covered.

So, we are taking two paths to cover all the statements.

1. x=1

Path - 1, 2, 3, 8

Output = 2

When we set value of x as 1 first it come on step 1 to read and assign the value of x (we took 1 in path) then come on statement 2 ($x > 0$ (we took 2 in path)) which is true and it comes on statement 3 (a = x+1 (we took 3 in path)) at last it comes on statement 8 to print the value of x (output is 2).

For the second path, we take the value of x is 1

2. Set x= -1

Path = 1, 2, 4, 5, 6, 5, 6, 5, 7, 8

Output = 2

When we set the value of x as -1 then first, it comes on step 1 to read and assign the value of x (we took 1 in the path) then come on step number 2 which is false because x is not greater than 0 ($x > 0$ and their x=-1). Due to false condition, it will not come on statement 3 and directly jump on statement 4 (we took 4 in path) and 4 is true ($x \leq 0$ and their x is less than 0) then come on statement 5 ($x < 1$ (we took 5 in path)) which is also true so it will come on statement 6 (x=x+1 (we took 6 in path)) and here x is incremented by 1.

So,

```
x=-1+1  
x=0
```

There is value of x become 0. Now it goes to statement 5($x < 1$ (we took 5 in path)) with value 0 and 0 is less than 1 so, it is true. Come on statement 6 ($x = x + 1$ (we took 6 in path))

```
x=x+1  
x= 0+1  
x=1
```

There x has become 1 and again goes to statement 5 ($x < 1$ (we took 5 in path)) and now 1 is not less than 1 so, condition is false and it will come to else part means statement 7 ($a = x + 1$ where the value of x is 1) and assign the value to a ($a = 2$). At last, it comes on statement 8 and print the value (Output is 2).

Make associations for the code:

Associations

In associations we list down all the definitions with all of its uses.

(1, (2, f), x), (1, (2, t), x), (1, 3, x), (1, (4, t), x), (1, (4, f), x), (1, (5, t), x), (1, (5, f), x), (1, 6, x), (1, 7, x), (6, (5, f)x), (6, (5, t)x), (6, 6, x), (3, 8, a), (7, 8, a).

How to make associations in data flow testing [<link>](#)

```
1 read x;  
2 If(x>0)          (1, (2, t), x), (1, (2, f), x)  
3 a= x+1           (1, 3, x)  
4 if (x<=0) {      (1, (4, t), x), (1, (4, f), x)  
5 if (x<1)         (1, (5, t), x), (1, (5, f), x)  
6 x=x+1; (go to 5) (1, 6, x)  
  
else  
7 a=x+1            (1, 7, x)  
8 print a;          (6, (5, f)x), (6, (5, t)x)  
                   (6, 6, x)  
                   (3, 8, a), (7, 8, a).
```

- **(1, (2, t), x), (1, (2, f), x)**- This association is made with statement 1 (read x;) and statement 2 (If($x > 0$)) where x is defined at line number 1, and it is used at line number 2 so, x is the variable. Statement 2 is logical, and it can be true or false that's why the association is defined in two ways; one is (1, (2, t), x) for true and another is (1, (2, f), x) for false.
- **(1, 3, x)**- This association is made with statement 1 (read x;) and statement 3 (a= x+1) where x is defined in statement 1 and used in statement 3. It is a computation use.
- **(1, (4, t), x), (1, (4, f), x)**- This association is made with statement 1 (read x;) and statement 4 (If($x \leq 0$)) where x is defined at line number 1 and it is used at line number 4 so x is the variable. Statement 4 is logical, and it can be true or false that's why the association is defined in two ways one is (1, (4, t), x) for true and another is (1, (4, f), x) for false.
- **(1, (5, t), x), (1, (5, f), x)**- This association is made with statement 1 (read x;) and statement 5 (if ($x < 1$)) where x is defined at line number 1, and it is used at line number 5, so x is the variable. Statement 5 is logical, and it can be true or false that's why the association is defined in two ways; one is (1, (5, t), x) for true and another is (1, (5, f), x) for false.
- **(1, 6, x)**- This association is made with statement 1 (read x;) and statement 6 ($x = x + 1$). x is defined in statement 1 and used in statement 6. It is a computation use.

- **(1, 7, x)**- This association is made with statement 1 (read x) and statement 7 ($a=x+1$). x is defined in statement 1 and used in statement 7 when statement 5 is false. It is a computation use.
- **(6, (5, f) x), (6, (5, t) x)**- This association is made with statement 6 ($x=x+1;$) and statement 5 if ($x<1$) because x is defined in statement 6 and used in statement 5. Statement 5 is logical, and it can be true or false that's why the association is defined in two ways one is $(6, (5, f) x)$ for true and another is $(6, (5, t) x)$ for false. It is a predicted use.
- **(6, 6, x)**- This association is made with statement 6 which is using the value of variable x and then defining the new value of x. $x=x+1 \rightarrow (-1+1)$ Statement 6 is using the value of variable x that is ?1 and then defining new value of x [$x=(-1+1) = 0$] that is 0.
- **(3, 8, a)**- This association is made with statement 3($a= x+1$) and statement 8 where variable a is defined in statement 3 and used in statement 8.
- **(7, 8, a)**- This association is made with statement 7($a=x+1$) and statement 8 where variable a is defined in statement 7 and used in statement 8.

Definition, c-use, p-use, c-use some p-use coverage, p-use some c-use coverage in data flow testing <link>

The next task is to group all the associations in Definition, c-use, p-use, c-use some p-use coverage, p-use some c-use coverage categories.

See the code below:

```

1. read x;
2. If(x>0)           (1, (2, t), x), (1, (2, f), x)
3. a= x+1            (1, 3, x)
4. if (x<=0) {        (1, (4, t), x), (1, (4, f), x)
5.   if (x<1)         (1, (5, t), x), (1, (5, f), x)
6.     x=x+1; (go to 5) (1, 6, x)
else
7.   a=x+1            (1, 7, x)
8.   print a;          (6,(5,f)x), (6,(5,t)x)
                      (6, 6, x)

```

So, these are the all association which contain definition, Predicate use (p-use), Computation use (c-use)

$(1, (2, f), x), (1, (2, t), x), (1, 3, x), (1, (4, t), x), (1, (4, f), x), (1, (5, t), x), (1, (5, f), x), (1, 6, x), (1, 7, x), (6,(5,f)x), (6,(5,t)x), (6, 6, x), (3, 8, a), (7, 8, a), (3, 8, a), (7, 8, a)$

Definition

Definition of a variable is the occurrence of a variable when the value is bound to the variable. In the above code, the value gets bound in the first statement and then start to flow.

- If($x>0$) is statement 2 in which value of x is bound with it. Association of statement 2 is $(1, (2, f), x), (1, (2, t), x)$.
- $a= x+1$ is statement 3 bounded with the value of x Association of statement 3 is $(1, 3, x)$

All definitions coverage

$(1, (2, f), x), (6,(5,f)x), (3, 8, a), (7, 8, a)$.

Predicate use (p-use)

If the value of a variable is used to decide an execution path is considered as predicate use (p-use). In control flow statements there are two

Statement 4 if ($x<=0$) is predicate use because it can be predicate as true or false. If it is true then if ($x<1$), $x=x+1$; execution path will be executed otherwise, else path will be executed.

Computation use (c-use)

If the value of a variable is used to compute a value for output or for defining another variable.

Statement 3 $a = x + 1$ (1, 3, x) **Statement 7** $a = x + 1$ (1, 7, x) **Statement 8** `print a` (3, 8, a), (7, 8, a).

These are **Computation use** because the value of x is used to compute and value of a is used for output.

All c-use coverage

(1, 3, x), (1, 6, x), (1, 7, x), (6, 6, x), (6, 7, x), (3, 8, a), (7, 8, a).

All c-use some p-use coverage

(1, 3, x), (1, 6, x), (1, 7, x), (6, 6, x), (6, 7, x), (3, 8, a), (7, 8, a).

All p-use some c-use coverage

(1, (2, f), x), (1, (2, t), x), (1, (4, t), x), (1, (4, f), x), (1, (5, t), x), (1, (5, f), x), (6, (5, f), x), (6, (5, t), x), (3, 8, a), (7, 8, a).

After collecting these groups, (By examining each point whether the variable is used at least once or not) tester can see all statements and variables are used. The statements and variables which are not used but exist in the code, get eliminated from the code.

Control Flow Testing

Control flow testing is a testing technique that comes under white box testing. The aim of this technique is to determine the execution order of statements or instructions of the program through a control structure. The control structure of a program is used to develop a test case for the program. In this technique, a particular part of a large program is selected by the tester to set the testing path. It is mostly used in unit testing. Test cases represented by the control graph of the program.

Control Flow Graph is formed from the node, edge, decision node, junction node to specify all possible execution path.

Notations used for Control Flow Graph

1. Node
2. Edge
3. Decision Node
4. Junction node

Node

Nodes in the control flow graph are used to create a path of procedures. Basically, it represents the sequence of procedures which procedure is next to come so, the tester can determine the sequence of occurrence of procedures.

We can see below in example the first node represent the start procedure and the next procedure is to assign the value of n after assigning the value there is decision node to decide next node of procedure as per the value of n if it is 18 or more than 18 so Eligible procedure will execute otherwise if it is less than 18 Not Eligible procedure executes. The next node is the junction node, and the last node is stop node to stop the procedure.

Edge

Edge in control flow graph is used to link the direction of nodes.

We can see below in example all arrows are used to link the nodes in an appropriate direction.

Decision node

Decision node in the control flow graph is used to decide next node of procedure as per the value.

We can see below in example decision node decide next node of procedure as per the value of n if it is 18 or more than 18 so Eligible procedure will execute otherwise if it is less than 18, Not Eligible procedure executes.

Junction node

Junction node in control flow graph is the point where at least three links meet.

Example

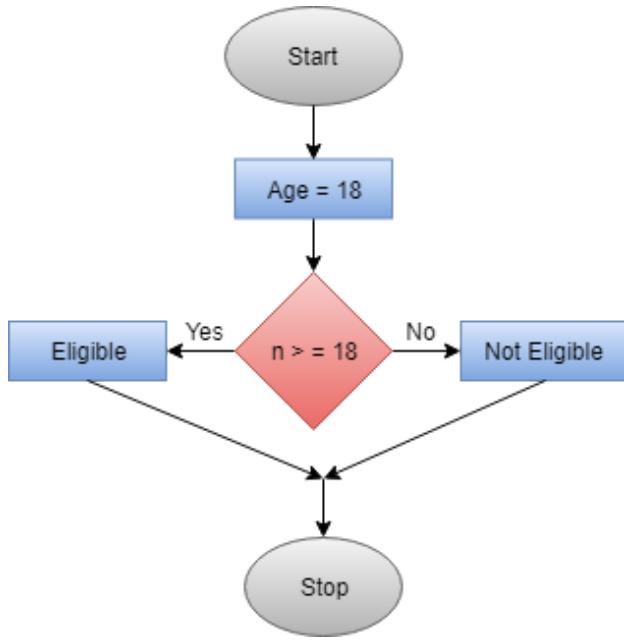
```
public class VoteEligibilityAge{  
  
    public static void main(String []args){  
        int n=45;  
        if(n>=18)  
        {  
            System.out.println("You are eligible for voting");  
        } else  
        {  
    }  
}
```

```

        System.out.println("You are not eligible for voting");
    }
}
}

```

Diagram - control flow graph



The above example shows eligibility criteria of age for voting where if age is 18 or more than 18 so print message "You are eligible for voting" if it is less than 18 then print "You are not eligible for voting."

Program for this scenario is written above, and the control flow graph is designed for the testing purpose.

In the control flow graph, start, age, eligible, not eligible and stop are the nodes, $n \geq 18$ is a decision node to decide which part (if or else) will execute as per the given value. Connectivity of the eligible node and not eligible node is there on the stop node.

Test cases are designed through the flow graph of the programs to determine the execution path is correct or not. All nodes, junction, edges, and decision are the essential parts to design test cases.

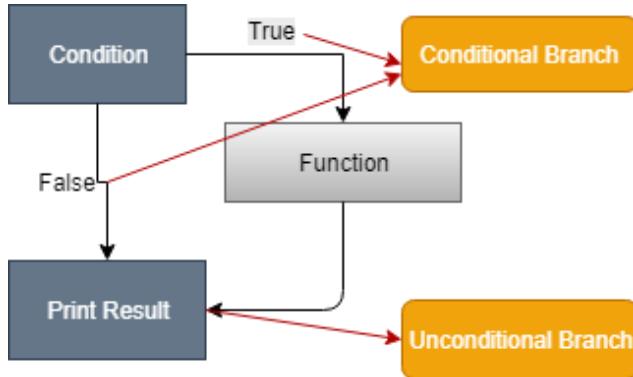
Branch Coverage Testing

Branch coverage technique is used to cover all branches of the control flow graph. It covers all the possible outcomes (true and false) of each condition of decision point at least once. Branch coverage technique is a whitebox testing technique that ensures that every branch of each decision point must be executed.

However, branch coverage technique and decision coverage technique are very similar, but there is a key difference between the two. Decision coverage technique covers all branches of each decision point whereas branch testing covers all branches of every decision point of the code.

In other words, branch coverage follows decision point and branch coverage edges. Many different metrics can be used to find branch coverage and decision coverage, but some of the most basic metrics are: finding the percentage of program and paths of execution during the execution of the program.

Like decision coverage, it also uses a control flow graph to calculate the number of branches.



How to calculate Branch coverage?

There are several methods to calculate Branch coverage, but pathfinding is the most common method.

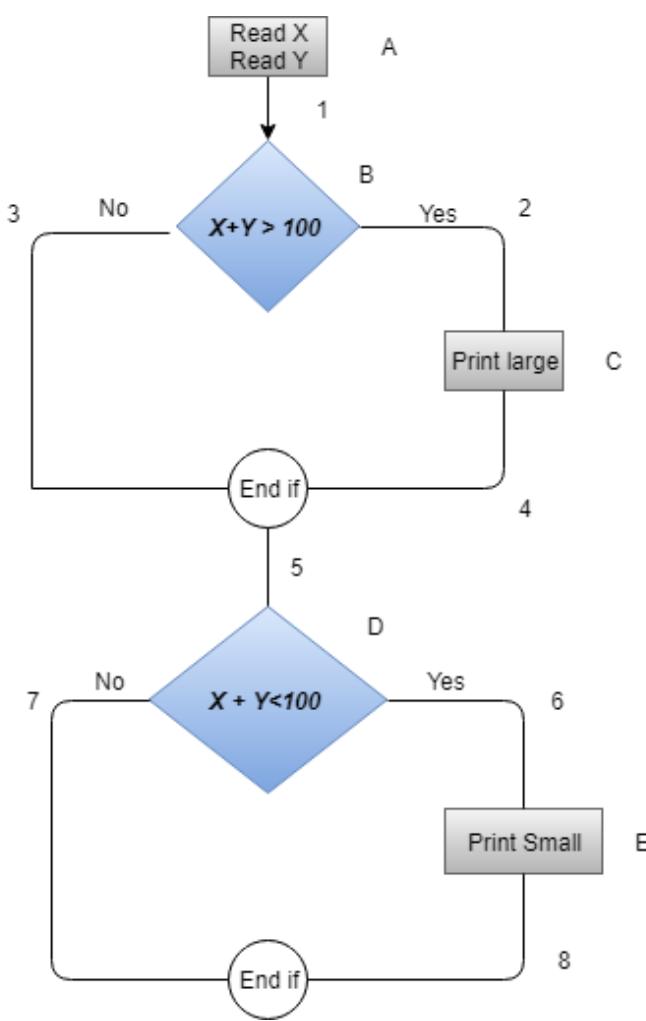
In this method, the number of paths of executed branches is used to calculate Branch coverage. Branch coverage technique can be used as the alternative of decision coverage. Somewhere, it is not defined as an individual technique, but it is distinct from decision coverage and essential to test all branches of the control flow graph.

Let's understand it with an example:

```
Read X  
Read Y  
IF X+Y > 100 THEN  
    Print "Large"  
ENDIF  
If X + Y<100 THEN  
    Print "Small"  
ENDIF
```

This is the basic code structure where we took two variables X and Y and two conditions. If the first condition is true, then print "Large" and if it is false, then go to the next condition. If the second condition is true, then print "Small."

Control flow graph of code structure



In the above diagram, control flow graph of code is depicted. In the first case traversing through "Yes" decision, the path is **A1-B2-C4-D6-E8**, and the number of covered edges is 1, 2, 4, 5, 6 and 8 but edges 3 and 7 are not covered in this path. To cover these edges, we have to traverse through "No" decision. In the case of "No" decision the path is A1-B3-5-D7, and the number of covered edges is 3 and 7. So by traveling through these two paths, all branches have covered.

Path 1 - A1-B2-C4-D6-E8

Path 2 - A1-B3-5-D7

Branch Coverage (BC) = Number of paths

=2

Case	Covered Branches	Path	Branch coverage
Yes	1, 2, 4, 5, 6, 8	A1-B2-C4-D6-E8	2
No	3,7	A1-B3-5-D7	

Statement Coverage Testing

Statement coverage is one of the widely used software testing. It comes under white box testing.

Statement coverage technique is used to design white box test cases. This technique involves execution of all statements of the source code at least once. It is used to calculate the total number of executed statements in the source code out of total statements present in the source code.

Statement coverage derives scenario of test cases under the white box testing process which is based upon the structure of the code.

$$\text{Statement coverage} = \frac{\text{Number of executed statements}}{\text{Total number of statements}} * 100$$

In white box testing, concentration of the tester is on the working of internal source code and flow chart or flow graph of the code.

Generally, in the internal source code, there is a wide variety of elements like operators, methods, arrays, looping, control statements, exception handlers, etc. Based on the input given to the program, some code statements are executed and some may not be executed. The goal of statement coverage technique is to cover all the possible executing statements and path lines in the code.

Let's understand the process of calculating statement coverage by an example:

Here, we are taking source code to create two different scenarios according to input values to check the percentage of statement coverage for each scenario.

Source Code Structure:

- Take input of two values like a=0 and b=1.
- Find the sum of these two values.
- If the sum is greater than 0, then print "This is the positive result."
- If the sum is less than 0, then print "This is the negative result."

```
input (int a, int b)
{
    Function to print sum of these integer values (sum = a+b)
    If (sum>0)
    {
        Print (This is positive result)
    } else
    {
        Print (This is negative result)
    }
}
```

So, this is the basic structure of the program, and that is the task it is going to do.

Now, let's see the two different scenarios and calculation of the percentage of Statement Coverage for given source code.

Scenario 1: If a = 5, b = 4

```
print (int a, int b) {
```

```

int sum = a+b;
if (sum>0)
print ("This is a positive result")
else
print ("This is negative result")
}

```

In scenario 1, we can see the value of sum will be 9 that is greater than 0 and as per the condition result will be "**This is a positive result.**" The statements highlighted in yellow color are executed statements of this scenario.

To calculate statement coverage of the first scenario, take the total number of statements that is 7 and the number of used statements that is 5.

Total number of statements = 7

Number of executed statements = 5

$$\text{Statement coverage} = \frac{\text{Number of executed statements}}{\text{Total number of statements}} * 100$$

$$\text{Statement coverage} = 5/7*100$$

$$= 500/7$$

$$= 71\%$$



Likewise, in scenario 2,

Scenario 2: If A = -2, B = -7

```

print (int a, int b) {
int sum = a+b;
if (sum>0)
print ("This is a positive result")
else
print ("This is negative result")
}

```

In scenario 2, we can see the value of sum will be -9 that is less than 0 and as per the condition, result will be "**This is a negative result.**" The statements highlighted in yellow color are executed statements of this scenario.

To calculate statement coverage of the first scenario, take the total number of statements that is 7 and the number of used statements that is 6.

Total number of statements = 7 Number of executed statements = 6

$$\text{Statement coverage} = \frac{\text{Number of executed statements}}{\text{Total number of statements}} * 100$$

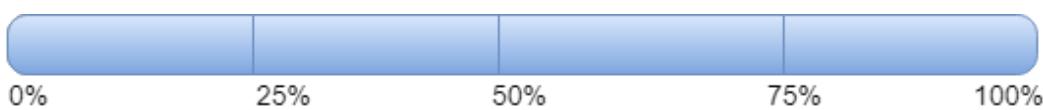
$$\text{Statement coverage} = 6/7*100
$$

$$= 600/7$$

$$= 85\%$$



But, we can see all the statements are covered in both scenario and we can consider that the overall statement coverage is 100%.



So, the statement coverage technique covers dead code, unused code, and branches.

Decision Coverage Testing

Decision coverage technique comes under white box testing which gives decision coverage to Boolean values. This technique reports true and false outcomes of Boolean expressions. Whenever there is a possibility of two or more outcomes from the statements like **do while statement**, **if statement** and **case statement** (Control flow statements), it is considered as decision point because there are two outcomes either true or false.

Decision coverage covers all possible outcomes of each and every Boolean condition of the code by using control flow graph or chart.

Generally, a decision point has two decision values one is true, and another is false that's why most of the times the total number of outcomes is two. The percent of decision coverage can be found by dividing the number of exercised outcome with the total number of outcomes and multiplied by 100.

$$\text{Decision Coverage} = \frac{\text{Number of Decision Outcomes Exercised}}{\text{Total Number of Decision Outcomes}} * 100$$

In this technique, it is tough to get 100% coverage because sometimes expressions get complicated. Due to this, there are several different methods to report decision coverage. All these methods cover the most important combinations and very much similar to decision coverage. The benefit of these methods is enhancement of the sensitivity of control flow.

We can find the number of decision coverage as follows.

Let's understand it by an example.

Consider the code to apply on decision coverage technique:

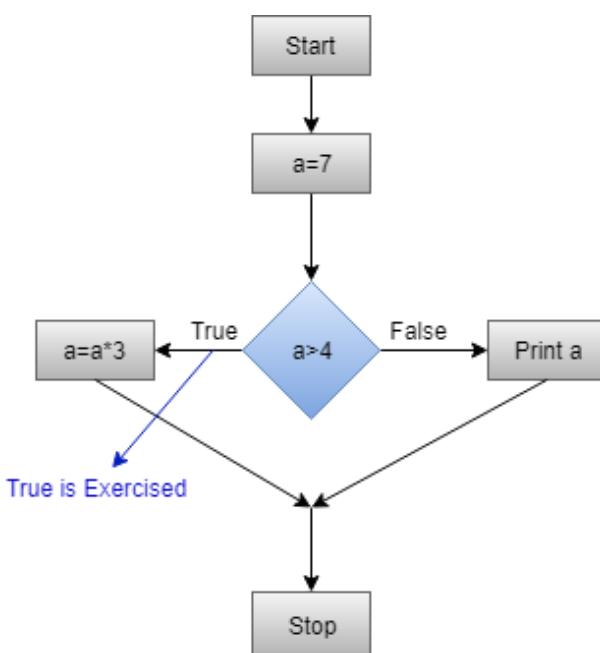
```
Test (int a)
{
    If(a>4)
        a=a*3
    Print (a)
}
```

Scenario 1: Value of a is 7 (a=7)

```
Test (int a=7)
{ if (a>4)
    a=a*3
    print (a)
}
```

The code highlighted in yellow is executed code. The outcome of this code is "True" if condition (a>4) is checked.

Control flow graph when the value of a is 7.



Calculation of Decision Coverage percent:

$$\text{Decision Coverage} = \frac{\text{Number of Decision Outcomes Exercised}}{\text{Total Number of Decision Outcomes}} * 100$$

Decision Coverage = $\frac{1}{2} * 100$ (Only "True" is exercised)

$$= 100/2 \\ = 50$$

Decision Coverage is 50%

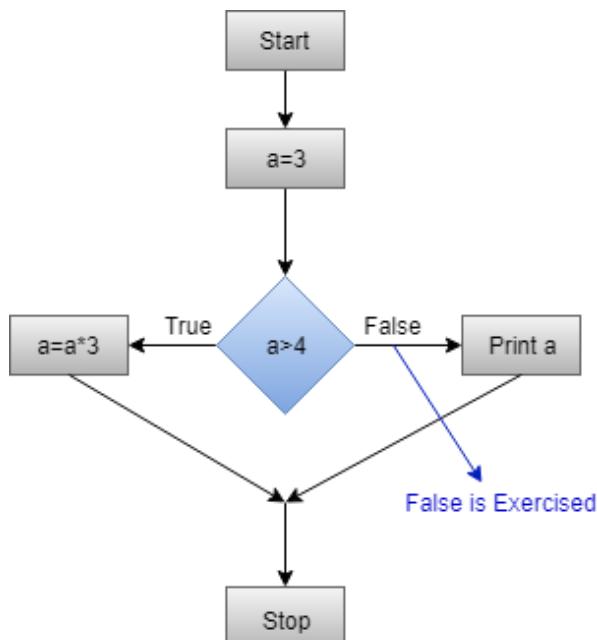
Scenario 2: Value of a is 3 (a=3)

```

Test (int a=3)
{ if (a>4)
  a=a*3
  print (a)
}
  
```

The code highlighted in yellow will be executed. The outcome of this code is ?False? if condition (a>4) is checked.

Control flow graph when the value of a is 3



Calculation of Decision Coverage percent:

Number of Decision Outcomes Exercised

$$\text{Decision Coverage} = \frac{\text{Number of Decision Outcomes Exercised}}{\text{Total Number of Decision Outcomes}} * 100$$

Total Number of Decision Outcomes $= \frac{1}{2} * 100$ (Only "False" is exercised)
 $= 100/2$ $= 50$

Decision Coverage = 50%

Result table of Decision Coverage:

Test Case	Value of A	Output	Decision Coverage
1	3	3	50%
2	7	21	50%

Decision table technique in Black box testing

Decision table technique is one of the widely used case design techniques for black box testing. This is a systematic approach where various input combinations and their respective system behavior are captured in a tabular form.

That's why it is also known as a cause-effect table. This technique is used to pick the test cases in a systematic manner; it saves the testing time and gives good coverage to the testing area of the software application.

Decision table technique is appropriate for the functions that have a logical relationship between two and more than two inputs.

This technique is related to the correct combination of inputs and determines the result of various combinations of input. To design the test cases by decision table technique, we need to consider conditions as input and actions as output.

Let's understand it by an example:

Most of us use an email account, and when you want to use an email account, for this you need to enter the email and its associated password.

If both email and password are correctly matched, the user will be directed to the email account's homepage; otherwise, it will come back to the login page with an error message specified with "Incorrect Email" or "Incorrect Password."

Now, let's see how a decision table is created for the login function in which we can log in by using email and password. Both the email and the password are the conditions, and the expected result is action.

Email (condition1)	T	T	F	F
Password (condition2)	T	F	T	F
Expected Result (Action)	Account Page	Incorrect password	Incorrect email	Incorrect email

In the table, there are four conditions or test cases to test the login function. In the first condition if both email and password are correct, then the user should be directed to account's Homepage.

In the second condition if the email is correct, but the password is incorrect then the function should display Incorrect Password. In the third condition if the email is incorrect, but the password is correct, then it should display Incorrect Email.

Now, in fourth and last condition both email and password are incorrect then the function should display Incorrect Email.

In this example, all possible conditions or test cases have been included, and in the same way, the testing team also includes all possible test cases so that upcoming bugs can be cured at testing level.

In order to find the number of all possible conditions, tester uses 2^n formula where n denotes the number of inputs; in the example there is the number of inputs is 2 (one is true and second is false).

$$\begin{aligned} \text{Number of possible conditions} &= 2^{\text{Number of Values of the second condition}} \\ &= 2^2 = 4 \end{aligned}$$

While using the decision table technique, a tester determines the expected output, if the function produces expected output, then it is passed in testing, and if not then it is failed. Failed software is sent back to the development team to fix the defect.

All-pairs Testing

All-pairs testing technique is also known as pairwise testing. It is used to test all the possible discrete combinations of values. This combinational method is used for testing the application that uses checkbox input, radio button input (radio button is used when you have to select only one option for example when you select gender male or female, you can select only one option), list box, text box, etc.

Suppose, you have a function of a software application for testing, in which there are 10 fields to input the data, so the total number of discrete combinations is 10^10 (100 billion), but testing of all combinations is complicated because it will take a lot of time.

So, let's understand the testing process with an example:

Assume that there is a function with a list box that contains 10 elements, text box that can accept 1 to 100 characters, radio button, checkbox and OK button.

The input values are given below that can be accepted by the fields of the given function.

1. Check Box - Checked or Unchecked
2. List Box - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
3. Radio Button - On or Off
4. Text Box - Number of alphabets between 1 to 100.
5. OK - Does not accept any value, only redirects to the next page.

Calculation of all the possible combinations:

Check Box = 2

List Box = 10

Radio Button = 2

Text Box = 100

Total number of test cases = $2 \times 10 \times 2 \times 100$

= 4000

The total number of test cases, including negative test cases, is 4000.

Testing of 4000 positive and negative test cases, is a very long and time-consuming process. Therefore, the task of the testing team is to reduce the number of test cases, to do this, the testing team considers the list box values in such a way that the first value is 0, and the other value can be any numeric number, neither positive nor negative. Ten values are now converted into 2 values.

Values of checkbox and radio button cannot be reduced because each has a combination of only 2 values. At last, the value of the text box is divided into three input categories valid integer, invalid integer, and alpha-special character.

Now, we have only 24 test cases, including negative test cases.

$2 \times 2 \times 2 \times 3 = 24$

Now, the task is to make combinations for all pair technique, into which each column should have an equal number of values, and the total value should be equal to 24.

In order to make text box column, put the most common input on the first place that is a **valid integer**, on the second place put the second most common input that is an **invalid integer**, and at the last place put the least common input that is an **Alpha Special Character**.

Then start filling the table, the first column is a text box with three values, the next column is a list box that has 2 values, the third column is a checkbox that has 2 values, and the last one is a radio button that also has 2 values.

Text box	List Box	Check Box	Radio Button
Valid Integer	0	Check	ON
Invalid Integer	Other	Uncheck	OFF
Valid Integer	0	Check	ON
Invalid Integer	Other	Uncheck	OFF
AlphaSpecialCharacter	0	Check	ON
AlphaSpecialCharacter	Other	Uncheck	OFF

In the table, we can see that the conventional software method results in 24 test cases instead of 4000 cases, and the pairwise testing method only in just 6 pair test cases.

Cause and Effect Graph in Black box Testing

Cause-effect graph comes under the black box testing technique which underlines the relationship between a given result and all the factors affecting the result. It is used to write dynamic test cases.

The dynamic test cases are used when code works dynamically based on user input. For example, while using email account, on entering valid email, the system accepts it but, when you enter invalid email, it throws an error message. In this technique, the input conditions are assigned with causes and the result of these input conditions with effects.

Cause-Effect graph technique is based on a collection of requirements and used to determine minimum possible test cases which can cover a maximum test area of the software.

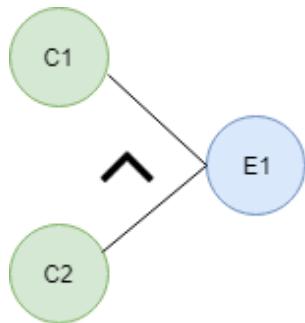
The main advantage of cause-effect graph testing is, it reduces the time of test execution and cost.

This technique aims to reduce the number of test cases but still covers all necessary test cases with maximum coverage to achieve the desired application quality.

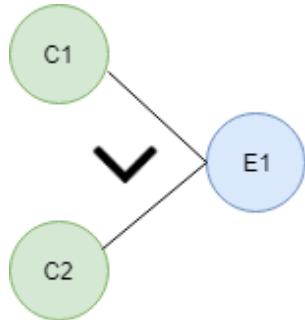
Cause-Effect graph technique converts the requirements specification into a logical relationship between the input and output conditions by using logical operators like AND, OR and NOT.

Notations used in the Cause-Effect Graph

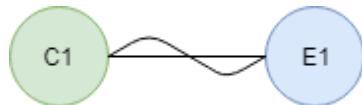
AND - E1 is an effect and C1 and C2 are the causes. If both C1 and C2 are true, then effect E1 will be true.



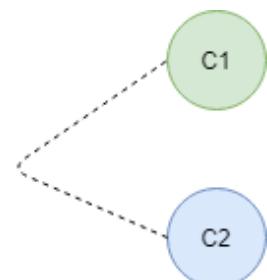
OR - If any cause from C1 and C2 is true, then effect E1 will be true.



NOT - If cause C1 is false, then effect E1 will be true.



Mutually Exclusive - When only one cause is true.



Let's try to understand this technique with some examples:

Situation:

The character in column 1 should be either A or B and in the column 2 should be a digit. If both columns contain appropriate values then update is made. If the input of column 1 is incorrect, i.e. neither A nor B, then message X will be displayed. If the input in column 2 is incorrect, i.e. input is not a digit, then message Y will be displayed.

- A file must be updated, if the character in the first column is either "A" or "B" and in the second column it should be a digit.
- If the value in the first column is incorrect (the character is neither A nor B) then message X will be displayed.
- If the value in the second column is incorrect (the character is not a digit) then message Y will be displayed.

Column 1	Column 2
<i>Correct value - A or B</i>	<i>Correct value- Any digit</i>
<i>Incorrect value - Any character except A or B</i>	<i>Incorrect value- Any character except digit</i>

Now, we are going to make a Cause-Effect graph for the above situation:

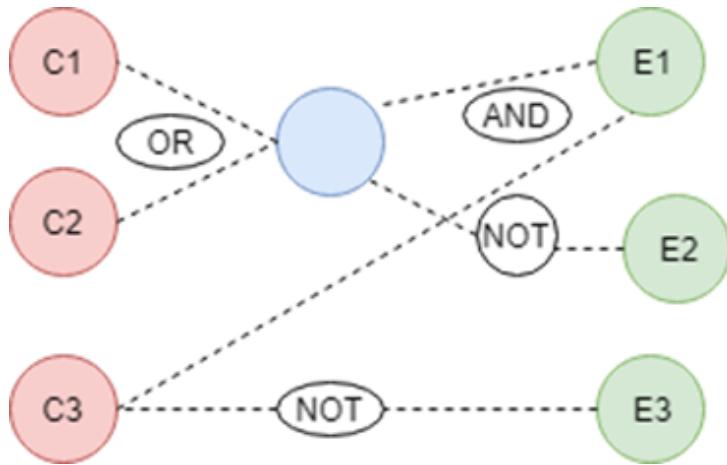
Causes are:

- C1 - Character in column 1 is A
- C2 - Character in column 1 is B
- C3 - Character in column 2 is digit!

Effects:

- E1 - Update made (C1 OR C2) AND C3
- E2 - Displays Message X (NOT C1 AND NOT C2)
- E3 - Displays Message Y (NOT C3)

Where AND, OR, NOT are the logical gates.



Effect E1- Update made- The logic for the existence of effect E1 is "**(C1 OR C2) AND C3**". For **C1 OR C2**, any one from C1 and C2 should be true. For logic **AND C3** (Character in column 2 should be a digit), C3 must be true. In other words, for the existence of effect E1 (Update made) any one from C1 and C2 but the C3 must be true. We can see in graph cause C1 and C2 are connected through OR logic and effect E1 is connected with AND logic.

Effect E2 - Displays Message X - The logic for the existence of effect E2 is "**NOT C1 AND NOT C2**" that means both C1 (Character in column 1 should be A) and C2 (Character in column 1 should be B) should be false. In other words, for the existence of effect E2 the character in column 1 should not be either A or B. We can see in the graph, **C1 OR C2** is connected through NOT logic with effect E2.

Effect E3 - Displays Message Y- The logic for the existence of effect E3 is "**NOT C3**" that means cause C3 (Character in column 2 is a digit) should be false. In other words, for the existence of effect E3, the character in column 2 should not be a digit. We can see in the graph, **C3** is connected through NOT logic with effect E3.

So, it is the cause-effect graph for the given situation. A tester needs to convert causes and effects into logical statements and then design cause-effect graph. If function gives output (effect) according to the input (cause) so, it is considered as defect free, and if not doing so, then it is sent to the development team for the correction.

Conclusion

Summary of the steps:

- Draw the circles for effects and Causes.
- Start from effect and then pick up what is the cause of this effect.
- Draw mutually exclusive causes (exclusive causes which are directly connected via one effect and one cause) at last.
- Use logic gates to draw dynamic test cases.

State Transition Technique

The general meaning of state transition is, different forms of the same situation, and according to the meaning, the state transition method does the same. It is used to capture the behavior of the software application when different input values are given to the same function.

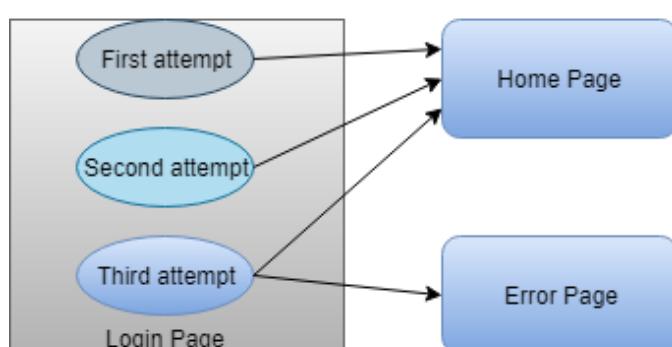
We all use the ATMs, when we withdraw money from it, it displays account details at last. Now we again do another transaction, then it again displays account details, but the details displayed after the second transaction are different from the first transaction, but both details are displayed by using the same function of the ATM. So the same function was used here but each time the output was different, this is called state transition. In the case of testing of a software application, this method tests whether the function is following state transition specifications on entering different inputs.

This applies to those types of applications that provide the specific number of attempts to access the application such as the login function of an application which gets locked after the specified number of incorrect attempts. Let's see in detail, in the login function we use email and password, it gives a specific number of attempts to access the application, after crossing the maximum number of attempts it gets locked with an error message.

The screenshot shows a purple-themed login interface. At the top center, it says "Login Here". Below that is a form area with two input fields. The first field is labeled "Email" and has a placeholder "Enter Your Email". The second field is labeled "Password" and has a placeholder "10 characters". At the bottom of the form are two blue buttons: "Back" on the left and "Login" on the right.

Let see in the diagram:

There is a login function of an application which provides a maximum three number of attempts, and after exceeding three attempts, it will be directed to an error page.



State transition table

STATE	LOGIN	VALIDATION	REDIRECTED
S1	First Attempt	Invalid	S2
S2	Second Attempt	Invalid	S3
S3	Third Attempt	Invalid	S5
S4	Home Page		
S5	Error Page		

In the above state transition table, we see that state S1 denotes first login attempt. When the first attempt is invalid, the user will be directed to the second attempt (state S2). If the second attempt is also invalid, then the user will be directed to the third attempt (state S3). Now if the third and last attempt is invalid, then the user will be directed to the error page (state S5).

But if the third attempt is valid, then it will be directed to the homepage (state S4).

Let's see state transition table if third attempt is valid:

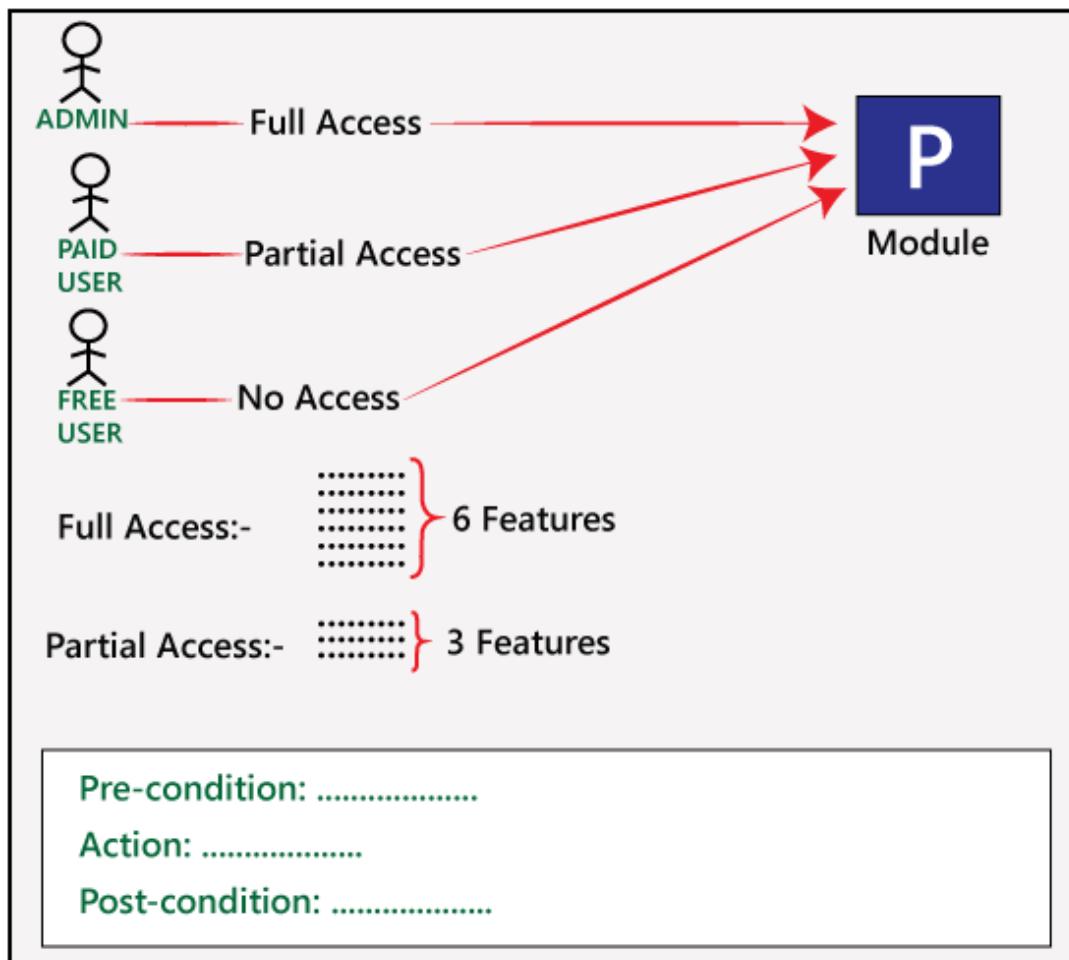
STATE	LOGIN	VALIDATION	REDIRECTED
S1	First Attempt	Invalid	S2
S2	Second Attempt	Invalid	S3
S3	Third Attempt	Valid	S4
S4	Home Page		
S5	Error Page		

By using the above state transition table we can perform testing of any software application. We can make a state transition table by determining desired output, and then exercise the software system to examine whether it is giving desired output or not.

Use Case Technique

The use case is functional testing of the black box testing used to identify the test cases from the beginning to the end of the system as per the usage of the system. By using this technique, the test team creates a test scenario that can exercise the entire software based on the functionality of each function from start to end.

It is a graphic demonstration of business needs, which describe how the end-user will cooperate with the software or the application. The use cases provide us all the possible techniques of how the end-user uses the application as we can see in the below image, that how the **use case** will look like:



In the above image, we can see that a sample of a use case where we have a requirement related to the customer requirement specification (CRS).

For **module P** of the software, we have six different features.

And here, **Admin** has access to all the **six features**, the **Paid user** has access to the **three features** and for the **Free user**, there is **no access** provided to any of the features.

Like for **Admin**, the different conditions would be as below:

Pre-condition→ Admin must be generated

Action→ Login as Paid user

Post-condition→ 3 features must be present

And for **Free user**, the different condition would be as below:

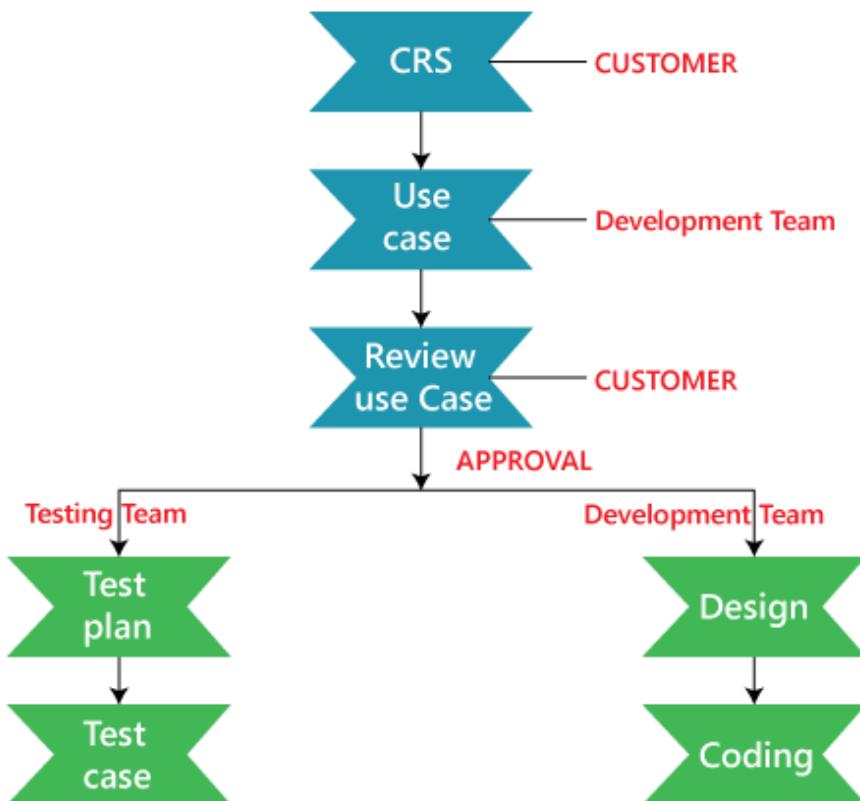
Pre-condition→ free user must be generated

Action→ Login as a free user

Post-condition→ no features

Who writes the use case?

The client provides the customer requirement specification for the application, then the development team will write the **use case** according to the CRS, and the use case is sent to the customer for their review.



If the client approves it, then the approved **use case** is sent to the development team for further design and coding process and these approved use case is also sent to the testing team, so they can start writing the test plan and later on start writing the test cases for the different features of the software.

In the below scenario, there is a tester who represents the user to use the functions of a system one by one. In this scenario, there is an actor who represents the user to use the functions of a software system.

This describes step-by-step functionality of the software application which can be understood with an example, assume that there is a software application of online money transfer. The various steps for transferring money are as follows:

- The user does login for the authentication of the actual user.
- The system checks ID and password with the database to ensure that whether it is a valid user or not.
- If the verification succeeds, the server connects the user to the account page, otherwise returns to the login page.
- In the account page, there are several options because the examiner is checking the money transfer option; the user goes into the money transfer option.
- After successful completion of this step, the user enters the account number in which he wants to transfer money. The user also need to enter other details like bank name, amount, IFSC code, home branch, etc.

In the last step, if there is a security feature that includes verification of the ATM card number and PIN, then enter the ATM card number, PIN and other required details.

If the system is successfully following all the steps, then there is no need to design test cases for this function. By describing the steps to use, it is easy to design test cases for software systems.

Difference between use case and prototype

Use case	Prototype
With the help of the use case, we get to know how the product should work. And it is a graphical representation of the software and its multiple features and also how they should work.	In this, we will not see how the end-user interacts with the application because it is just a dummy (particular image of the software) of the application.

How developers develop the use cases

The developers use the standard symbols to write a use case so that everyone will understand easily. They will use the **Unified modeling language** (UML) to create the use cases.

There are various tools available that help to write a use case, such as **Rational Rose**. This tool has a predefined UML symbols, we need to drag and drop them to write a use case, and the developer can also use these symbols to develop the use case.

Advantage of Use Case Technique

The use case technique gives us some features which help us to create an application.

Following are the benefits of using the use case technique while we are developing the product:

- The use case is used to take the functional needs of the system.
- These are the classification of steps, which describe the connections between the user and its actions.
- It starts from an elementary view where the system is created first and primarily used for its users.
- It is used to determine the complete analyses, which help us to achieve the complication, and then it focuses on the one detailed features at a time.

Functional Testing

Before proceeding to functional testing, we should know about the testing, what testing is?

What is testing?

In simple terms, the testing is to compare the actual result with the expected result. Testing is done to identify whether all the function is working as expectations.

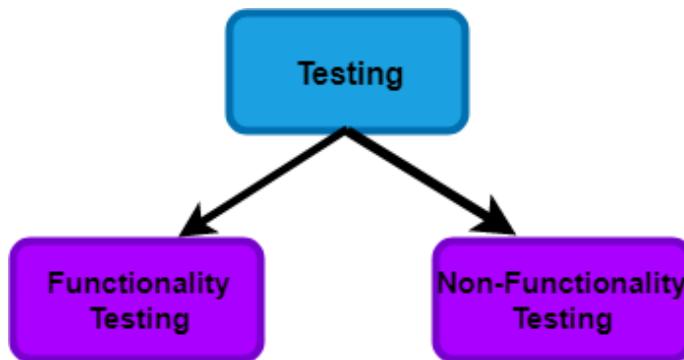
What is Software Testing?

Software testing is a technique to check whether the actual result matches the expected result and to ensure that the software has not any defect or bug.

Software testing ensures that the application has not any defect or the requirement is missing to the actual need. Either manual or automation testing can do software testing.

Software testing also defines as verification of application under test (AUT).

There are two types of testing:



Functional Testing:

It is a type of software testing which is used to verify the functionality of the software application, whether the function is working according to the requirement specification. In functional testing, each function tested by giving the value, determining the output, and verifying the actual output with the expected value. Functional testing performed as black-box testing which is presented to confirm that the functionality of an application or system behaves as we are expecting. It is done to verify the functionality of the application.

Functional testing also called as black-box testing, because it focuses on application specification rather than actual code. Tester has to test only the program rather than the system.

Goal of functional testing

The purpose of the functional testing is to check the primary entry function, necessarily usable function, the flow of screen GUI. Functional testing displays the error message so that the user can easily navigate throughout the application.

What is the process of functional testing?

Testers follow the following steps in the functional testing:

- Tester does verification of the requirement specification in the software application.
- After analysis, the requirement specification tester will make a plan.
- After planning the tests, the tester will design the test case.
- After designing the test, case tester will make a document of the traceability matrix.
- The tester will execute the test case design.
- Analysis of the coverage to examine the covered testing area of the application.
- Defect management should do to manage defect resolving.

Functional Testing



What to test in functional testing? Explain

The main objective of functional testing is checking the functionality of the software system. It concentrates on:

- **Basic Usability:** Functional Testing involves the usability testing of the system. It checks whether a user can navigate freely without any difficulty through screens.
- **Accessibility:** Functional testing test the accessibility of the function.
- **Mainline function:** It focuses on testing the main feature.
- **Error Condition:** Functional testing is used to check the error condition. It checks whether the error message displayed.

Explain the complete process to perform functional testing.

There are the following steps to perform functional testing:

- There is a need to understand the software requirement.
- Identify test input data
- Compute the expected outcome with the selected input values.
- Execute test cases
- Comparison between the actual and the computed result

Identify test input(input data)

Compute the expected outcome with the selected input values

Execute test cases

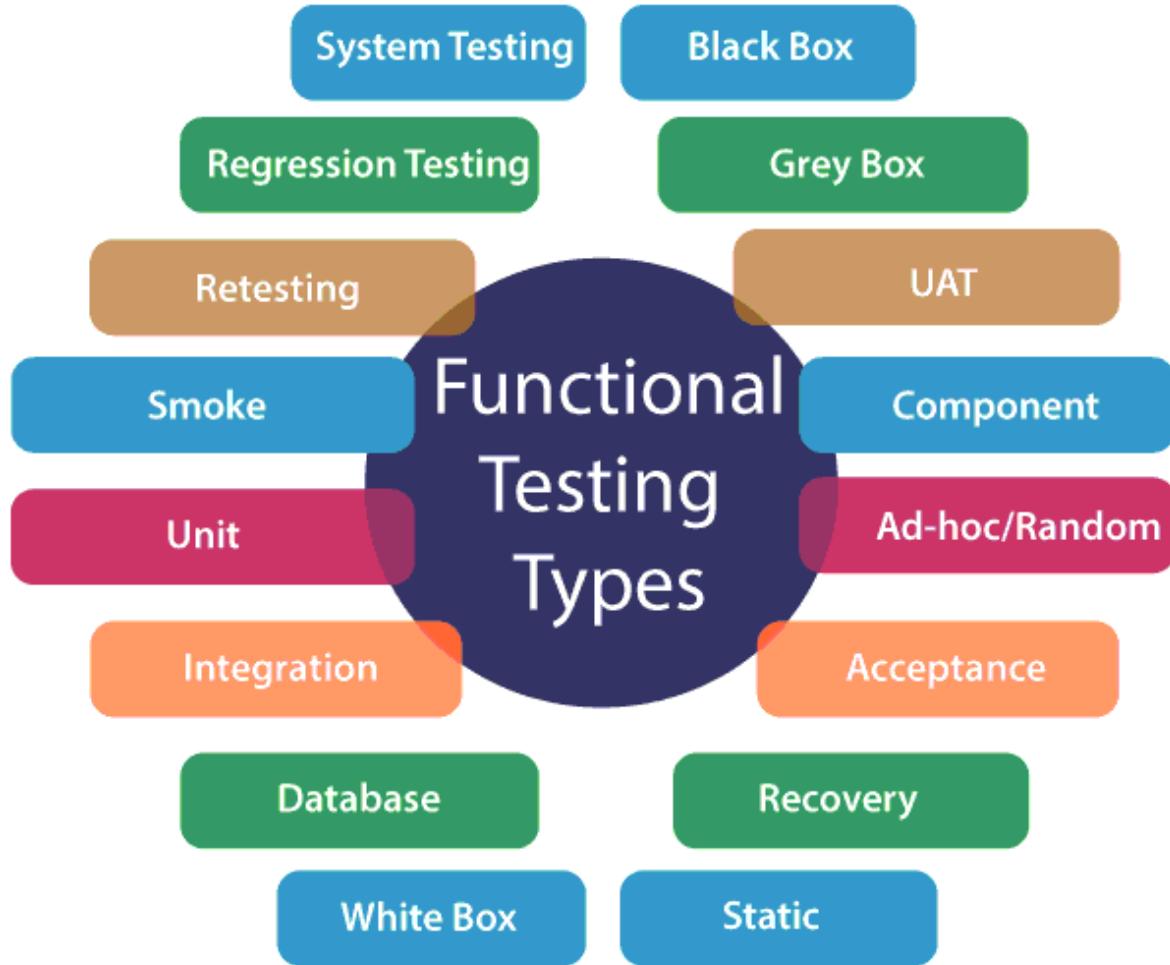
Comparison of actual and computed expected result

Explain the types of functional testing.

The main objective of functional testing is to test the functionality of the component.

Functional testing is divided into multiple parts.

Here are the following types of functional testing.



Unit Testing: Unit testing is a type of software testing, where the individual unit or component of the software tested. Unit testing, examine the different part of the application, by unit testing functional testing also done, because unit testing ensures each module is working correctly.

The developer does unit testing. Unit testing is done in the development phase of the application.

Smoke Testing: Functional testing by smoke testing. Smoke testing includes only the basic (feature) functionality of the system. Smoke testing is known as "**Build Verification Testing**." Smoke testing aims to ensure that the most important function work.

For example, Smoke testing verifies that the application launches successfully will check that GUI is responsive.

Sanity Testing: Sanity testing involves the entire high-level business scenario is working correctly. Sanity testing is done to check the functionality/bugs fixed. Sanity testing is little advance than smoke testing.

For example, login is working fine; all the buttons are working correctly; after clicking on the button navigation of the page is done or not.

Regression Testing: This type of testing concentrate to make sure that the code changes should not side effect the existing functionality of the system. Regression testing specifies when bug arises in the system after fixing the bug, regression testing concentrate on that all parts are working or not. Regression testing focuses on is there any impact on the system.

Integration Testing: **Integration testing** combined individual units and tested as a group. The purpose of this testing is to expose the faults in the interaction between the integrated units.

Developers and testers perform integration testing.

White box testing: **White box testing** is known as Clear Box testing, code-based testing, structural testing, extensive testing, and glass box testing, transparent box testing. It is a software testing method in which the internal structure/design/ implementation tested known to the tester.

The white box testing needs the analysis of the internal structure of the component or system.

Black box testing: It is also known as behavioral testing. In this testing, the internal structure/ design/ implementation not known to the tester. This type of testing is functional testing. Why we called this type of testing is black-box testing, in this testing tester, can't see the internal code.

For example, A tester without the knowledge of the internal structures of a website tests the web pages by using the web browser providing input and verifying the output against the expected outcome.

User acceptance testing: It is a type of testing performed by the client to certify the system according to requirement. The final phase of testing is user acceptance testing before releasing the software to the market or production environment. UAT is a kind of black-box testing where two or more end-users will involve.

Retesting: **Retesting** is a type of testing performed to check the test cases that were unsuccessful in the final execution are successfully pass after the defects fixed. Usually, tester assigns the bug when they find it while testing the product or its component. The bug allocated to a developer, and he fixes it. After fixing, the bug is assigned to a tester for its verification. This testing is known as retesting.

Database Testing: Database testing is a type of testing which checks the schema, tables, triggers, etc. of the database under test. Database testing may involve creating complex queries to load/stress test the database and check its responsiveness. It checks the data integrity and consistency.

Example: let us consider a banking application whereby a user makes a transaction. Now from database testing following, things are important. They are:

- Application store the transaction information in the application database and displays them correctly to the user.
- No information lost in this process
- The application does not keep partially performed or aborted operation information.
- The user information is not allowed individuals to access by the

Ad-hoc testing: Ad-hoc testing is an informal testing type whose aim is to break the system. This type of software testing is unplanned activity. It does not follow any test design to create the test cases. Ad-hoc testing is done randomly on any part of the application; it does not support any structured way of testing.

Recovery Testing: **Recovery testing** is used to define how well an application can recover from crashes, hardware failure, and other problems. The purpose of recovery testing is to verify the system's ability to recover from testing points of failure.

Static Testing: **Static testing** is a software testing technique by which we can check the defects in software without actually executing it. Static testing is done to avoid errors in the early stage of the development as it is easier to find failure in the early stages. Static testing used to detect the mistakes that may not found in dynamic testing.

Why we use static testing?

Static testing helps to find the error in the early stages. With the help of static testing, this will reduce the development timescales. It reduces the testing cost and time. Static testing also used for development productivity.

Component Testing: **Component Testing** is also a type of software testing in which testing is performed on each component separately without integrating with other parts. Component testing is also a type of black-box testing. Component testing also referred to as Unit testing, program testing, or module testing.

Grey Box Testing: **Grey Box Testing** defined as a combination of both white box and black-box testing. Grey Box testing is a testing technique which performed with limited information about the internal functionality of the system.



What are the functional testing tools?

The functional testing can also be executed by various apart from manual testing. These tools simplify the process of testing and help to get accurate and useful results.

It is one of the significant and top-priority based techniques which were decided and specified before the development process.

The tools used for functional testing are:

Tools	Features/ Characteristics
Sahi	<ul style="list-style-type: none">○ It is an open-source and automation testing tool, released under Apache License open source license, used for testing of the web application.○ Sahi is written in Java and JavaScript and considered for most of the testing techniques.○ It runs as a proxy server; it is browser-independent.
SoapUI	<ul style="list-style-type: none">○ It is an open-source functional testing tool, used for web application testing.○ It is simple and easy to design.○ It supports multiple environments, i.e., at any instance, the target environment may be set up.
Watir	<ul style="list-style-type: none">○ Watir, is an abbreviated form of web application testing in ruby, is an open-source tool for automating web browser./li>○ It uses a ruby scripting language, which is concise and easy to use./li>○ Watir supports multiple browsers on various platform.
Selenium	<ul style="list-style-type: none">○ The open-source tool, used for functional testing on both web application and applications of the desktop.

	<ul style="list-style-type: none"> ◦ It automates browsers and web application for testing purpose. ◦ It gives the flexibility to customize the automated test case ◦ Provides the advantage of writing test scripts, as per the requirements, using web driver.
Canoo WebTest	<ul style="list-style-type: none"> ◦ An open-source tool for performing functional testing of the web application. ◦ Platform independent ◦ Easy and fast ◦ Easy to extend to meet growing and incoming requirements.
Cucumber	<ul style="list-style-type: none"> ◦ Cucumber is an open-source testing tool written in Ruby language. This tool works best for test-driven development. It is used to test many other languages like java, c#, and python. Cucumber for testing using some programming.

What are the advantages of Functional Testing?

Advantages of functional testing are:

- It produces a defect-free product.
- It ensures that the customer is satisfied.
- It ensures that all requirements met.
- It ensures the proper working of all the functionality of an application/software/product.
- It ensures that the software/ product work as expected.
- It ensures security and safety.
- It improves the quality of the product.

Example: Here, we are giving an example of banking software. In a bank when money transferred from bank A to bank B. And the bank B does not receive the correct amount, the fee is applied, or the money not converted into the correct currency, or incorrect transfer or bank A does not receive statement advice from bank B that the payment has received. These issues are critical and can be avoided by proper functional testing.

What are the disadvantages of functional testing?

Disadvantages of functional testing are:

- Functional testing can miss a critical and logical error in the system.
- This testing is not a guarantee of the software to go live.
- The possibility of conducting redundant testing is high in functional testing.

Wrap Up

Here, we can easily conclude that to build a strong foundation of a top-class software product, functional testing is essential. It acts as a foundation of the structure, and it is a crucial part of every test routine.

Non-Functional Testing

Non-functional testing is a type of software testing to test non-functional parameters such as reliability, load test, performance and accountability of the software. The primary purpose of non-functional testing is to test the reading speed of the software system as per non-functional parameters. The parameters of non-functional testing are never tested before the functional testing.

Non-functional testing is also very important as functional testing because it plays a crucial role in customer satisfaction.

For example, non-functional testing would be to test how many people can work simultaneously on any software.

Why Non-Functional Testing

Functional and Non-functional testing both are mandatory for newly developed software. Functional testing checks the correctness of internal functions while Non-Functional testing checks the ability to work in an external environment.

It sets the way for software installation, setup, and execution. The measurement and metrics used for internal research and development are collected and produced under non-functional testing.

Non-functional testing gives detailed knowledge of product behavior and used technologies. It helps in reducing the risk of production and associated costs of the software.

Parameters to be tested under Non-Functional Testing



Performance Testing

Performance Testing eliminates the reason behind the slow and limited performance of the software. Reading speed of the software should be as fast as possible.

For Performance Testing, a well-structured and clear specification about expected speed must be defined. Otherwise, the outcome of the test (Success or Failure) will not be obvious.

Load Testing

Load testing involves testing the system's loading capacity. Loading capacity means more and more people can work on the system simultaneously.

Security Testing

Security testing is used to detect the security flaws of the software application. The testing is done via investigating system architecture and the mindset of an attacker. Test cases are conducted by finding areas of code where an attack is most likely to happen.

Portability Testing

The portability testing of the software is used to verify whether the system can run on different operating systems without occurring any bug. This test also tests the working of software when there is a same operating system but different hardware.

Accountability Testing

Accountability test is done to check whether the system is operating correctly or not. A function should give the same result for which it has been created. If the system gives expected output, it gets passed in the test otherwise failed.

Reliability Testing

Reliability test assumes that whether the software system is running without fail under specified conditions or not. The system must be run for a specific time and number of processes. If the system is failed under these specified conditions, reliability test will be failed.

Efficiency Testing

Efficiency test examines the number of resources needed to develop a software system, and how many of these were used. It also includes the test of these three points.

- Customer's requirements must be satisfied by the software system.
- A software system should achieve customer specifications.
- Enough efforts should be made to develop a software system.

Advantages of Non-functional testing

- It provides a higher level of security. Security is a fundamental feature due to which system is protected from cyber-attacks.
- It ensures the loading capability of the system so that any number of users can use it simultaneously.
- It improves the performance of the system.
- Test cases are never changed so do not need to write them more than once.
- Overall time consumption is less as compared to other testing processes.

Disadvantages of Non-Functional Testing

- Every time the software is updated, non-functional tests are performed again.
- Due to software updates, people have to pay to re-examine the software; thus software becomes very expensive.

Unit Testing

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

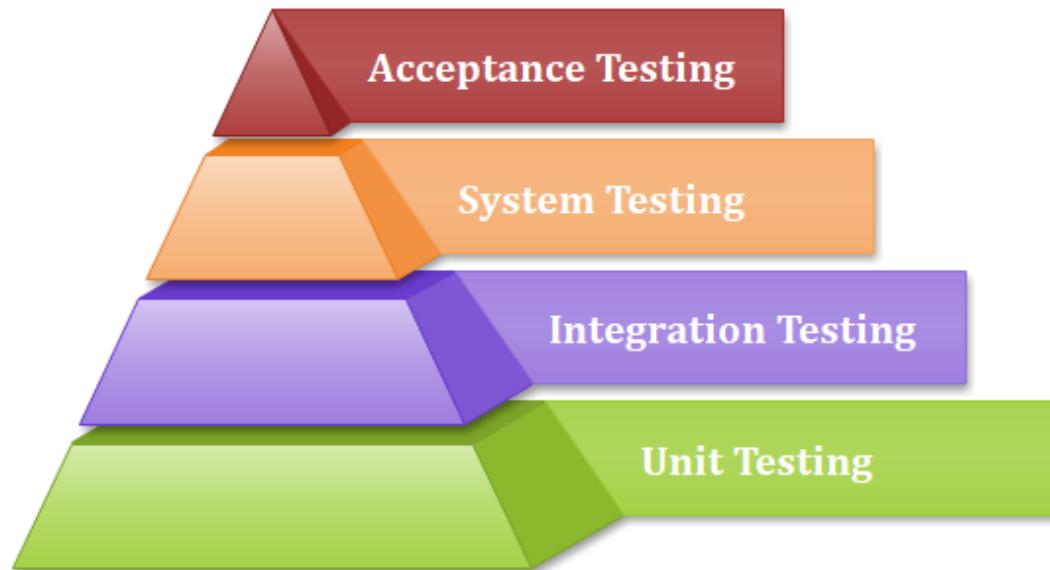
A unit is a single testable part of a software system and tested during the development phase of the application software.

The purpose of unit testing is to test the correctness of isolated code. A unit component is an individual function or code of the application. White box testing approach used for unit testing and usually done by the developers.

Whenever the application is ready and given to the Test engineer, he/she will start checking every component of the module or module of the application independently or one by one, and this process is known as **Unit testing** or **components testing**.

Why Unit Testing?

In a testing level hierarchy, unit testing is the first level of testing done before integration and other remaining levels of the testing. It uses modules for the testing process which reduces the dependency of waiting for Unit testing frameworks, stubs, drivers and mock objects are used for assistance in unit testing.



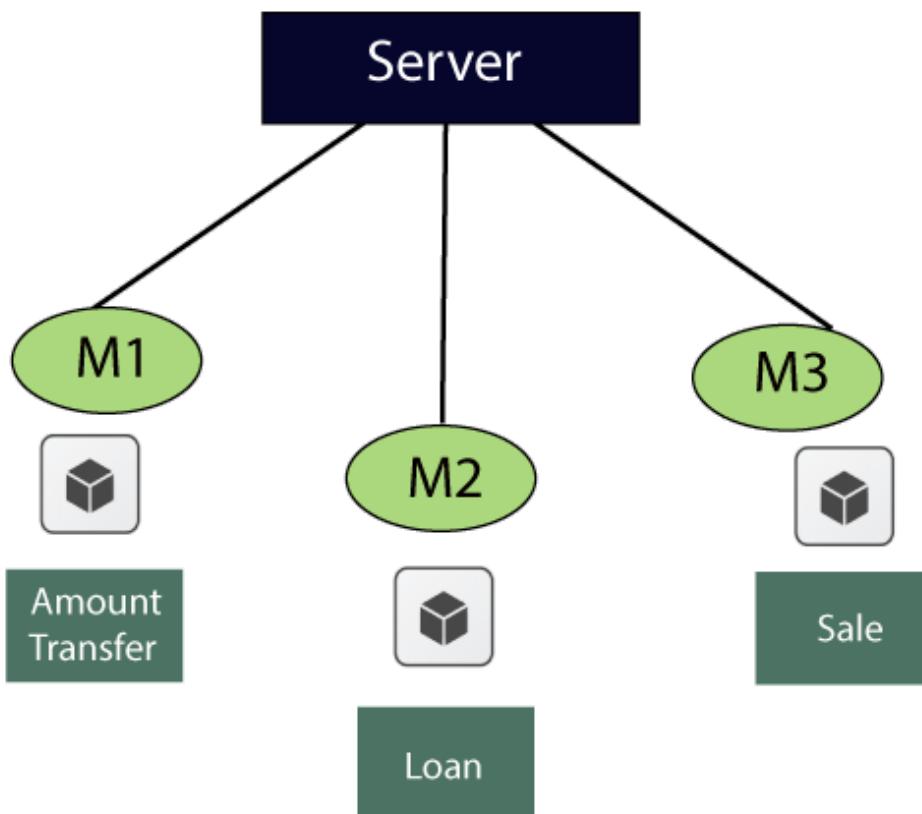
Generally, **the** software goes under four level of testing: Unit Testing, Integration Testing, System Testing, and Acceptance Testing but sometimes due to time consumption software testers does minimal unit testing but skipping of unit testing may lead to higher defects during Integration Testing, System Testing, and Acceptance Testing or even during Beta Testing which takes place after the completion of software application.

Some crucial reasons are listed below:

- Unit testing helps tester and developers to understand the base of code that makes them able to change defect causing code quickly.
- Unit testing helps in the documentation.
- Unit testing fixes defects very early in the development phase that's why there is a possibility to occur a smaller number of defects in upcoming testing levels.
- It helps with code reusability by migrating code and test cases.

Example of Unit testing

Let us see one sample example for a better understanding of the concept of unit testing:



For the **amount transfer**, requirements are as follows:

1.	Amount transfer
1.1	From account number (FAN) → Text Box
1.1.1	FAN → accept only 4 digit
1.2	To account no (TAN) → Text Box
1.2.1	TAN → Accept only 4 digit
1.3	Amount → Text Box
1.3.1	Amount → Accept maximum 4 digit
1.4	Transfer → Button
1.4.1	Transfer → Enabled
1.5	Cancel → Button
1.5.1	Cancel → Enabled

Below are the application access details, which is given by the customer

- URL → login Page
- Username/password/OK → home page
- To reach Amount transfer module follow the below

Loans → sales → Amount transfer

While performing unit testing, we should follow some rules, which are as follows:

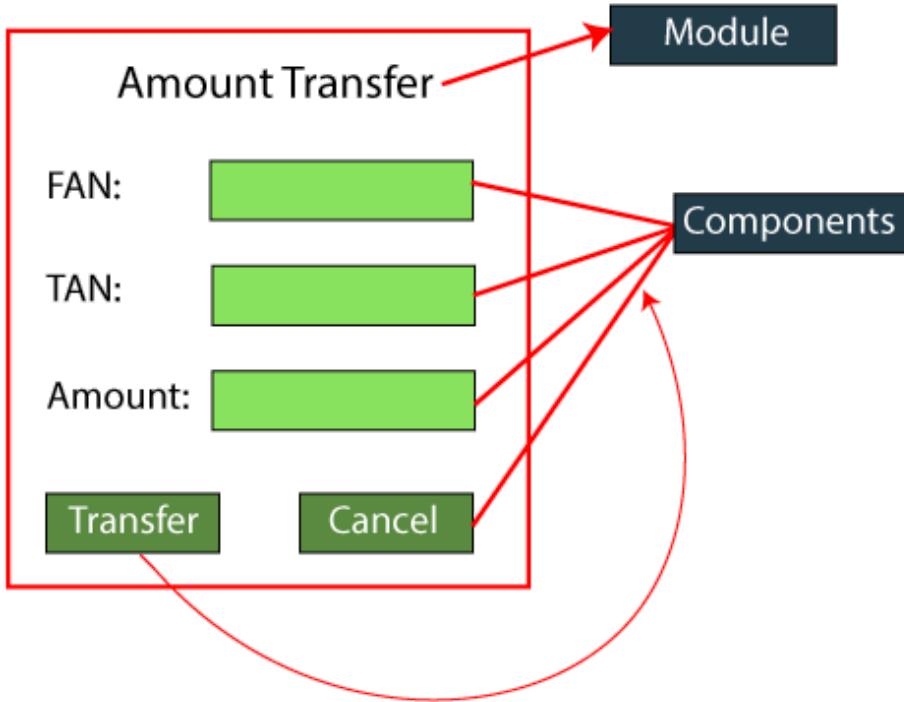
- To start unit testing, at least we should have one module.
- Test for positive values
- Test for negative values
- No over testing

- No assumption required

When we feel that the **maximum test coverage** is achieved, we will stop the testing.

Now, we will start performing the unit testing on the different components such as

- **From account number(FAN)**
- **To account number(TAN)**
- **Amount**
- **Transfer**
- **Cancel**



For the FAN components

Values	Description
1234	accept
4311	Error message → account valid or not
blank	Error message → enter some values
5 digit/ 3 digit	Error message → accept only 4 digit
Alphanumeric	Error message → accept only digit
Blocked account no	Error message
Copy and paste the value	Error message → type the value
Same as FAN and TAN	Error message

For the TAN component

- Provide the values just like we did in **From account number** (FAN) components

For Amount component

- Provide the values just like we did in FAN and TAN components.

For Transfer component

- Enter valid FAN value
- Enter valid TAN value
- Enter the correct value of Amount
- Click on the Transfer button → amount transfer successfully(confirmation message)

For Cancel Component

- Enter the values of FAN, TAN, and amount.
- Click on the Cancel button → all data should be cleared.

Unit Testing Tools

We have various types of unit testing tools available in the market, which are as follows:

- NUnit
- JUnit
- PHPunit
- Parasoft Jtest
- EMMA

For more information about Unit testing tools, refers to the below link:

<https://www.javatpoint.com/unit-testing-tools>

Unit Testing Techniques:

Unit testing uses all white box testing techniques as it uses the code of software application:

- Data flow Testing
- Control Flow Testing
- Branch Coverage Testing
- Statement Coverage Testing
- Decision Coverage Testing

How to achieve the best result via Unit testing?

Unit testing can give best results without getting confused and increase complexity by following the steps listed below:

- Test cases must be independent because if there is any change or enhancement in requirement, the test cases will not be affected.
- Naming conventions for unit test cases must be clear and consistent.
- During unit testing, the identified bugs must be fixed before jump on next phase of the SDLC.
- Only one code should be tested at one time.
- Adopt test cases with the writing of the code, if not doing so, the number of execution paths will be increased.
- If there are changes in the code of any module, ensure the corresponding unit test is available or not for that module.

Advantages and disadvantages of unit testing

The pros and cons of unit testing are as follows:

Advantages

- Unit testing uses module approach due to that any part can be tested without waiting for completion of another parts testing.
- The developing team focuses on the provided functionality of the unit and how functionality should look in unit test suits to understand the unit API.
- Unit testing allows the developer to refactor code after a number of days and ensure the module still working without any defect.

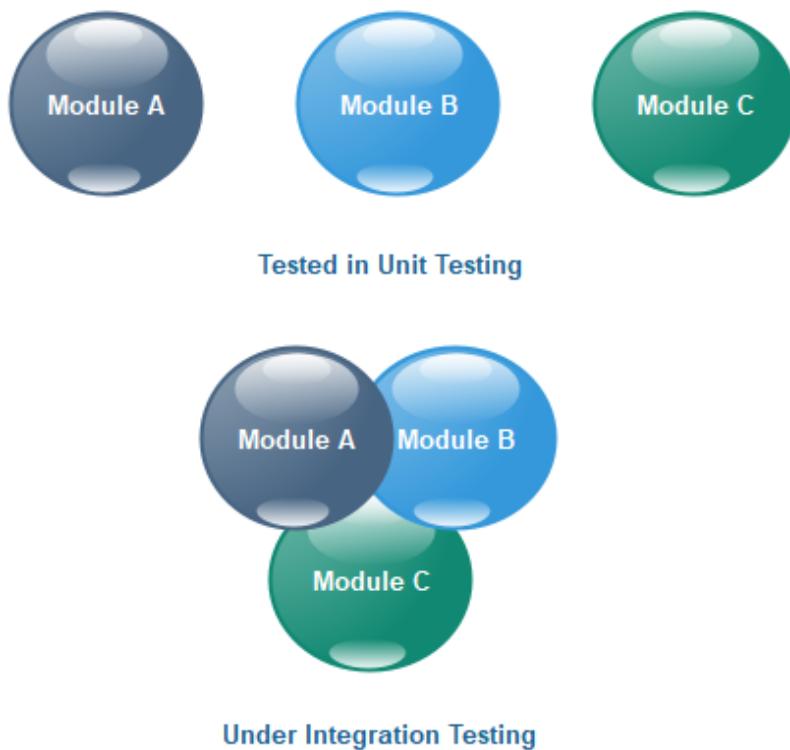
Disadvantages

- It cannot identify integration or broad level error as it works on units of the code.
- In the unit testing, evaluation of all execution paths is not possible, so unit testing is not able to catch each and every error in a program.
- It is best suitable for conjunction with other testing activities.

Integration testing

Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.

Unit testing uses modules for testing purpose, and these modules are combined and tested in integration testing. The Software is developed with a number of software modules that are coded by different coders or programmers. The goal of integration testing is to check the correctness of communication among all the modules.



Once all the components or modules are working independently, then we need to check the data flow between the dependent modules is known as **integration testing**.

Let us see one sample example of a banking application, as we can see in the below image of amount transfer.

A screenshot of a mobile banking application interface titled "Amount Transfer". The screen has a purple header and a white body. It contains three input fields: "FAN:" with a placeholder box, "TAN:" with a placeholder box, and "AMOUNT:" with a placeholder box. At the bottom, there are two buttons: "Transfer" in blue and "Cancel" in red.

- First, we will login as a user **P** to amount transfer and send Rs200 amount, the confirmation message should be displayed on the screen as **amount transfer successfully**. Now logout as P and login as user **Q** and go to amount balance page and check for a balance in that account = Present balance + Received Balance. Therefore, the integration test is successful.
- Also, we check if the amount of balance has reduced by Rs200 in P user account.

- Click on the transaction, in P and Q, the message should be displayed regarding the data and time of the amount transfer.

Guidelines for Integration Testing

- We go for the integration testing only after the functional testing is completed on each module of the application.
- We always do integration testing by picking module by module so that a proper sequence is followed, and also we don't miss out on any integration scenarios.
- First, determine the test case strategy through which executable test cases can be prepared according to test data.
- Examine the structure and architecture of the application and identify the crucial modules to test them first and also identify all possible scenarios.
- Design test cases to verify each interface in detail.
- Choose input data for test case execution. Input data plays a significant role in testing.
- If we find any bugs then communicate the bug reports to developers and fix defects and retest.
- Perform **positive and negative integration testing**.

Here **positive** testing implies that if the total balance is Rs15, 000 and we are transferring Rs1500 and checking if the amount transfer works fine. If it does, then the test would be a pass.

And **negative testing** means, if the total balance is Rs15, 000 and we are transferring Rs20, 000 and check if amount transfer occurs or not, if it does not occur, the test is a pass. If it happens, then there is a bug in the code, and we will send it to the development team for fixing that bug.

Note: Any application in this world will do functional testing compulsory, whereas integration testing will be done only if the modules are dependent on each other. Each integration scenarios should compulsorily have source→ data→destination. Any scenarios can be called as integration scenario only if the data gets saved in the destination.

For example: In the Gmail application, the **Source** could be **Compose**, **Data** could be **Email** and the **Destination** could be the **Inbox**.

Example of integration testing

Let us assume that we have a **Gmail** application where we perform the integration testing.

First, we will do **functional testing** on the **login page**, which includes the various components such as **username**, **password**, **submit**, and **cancel** button. Then only we can perform integration testing.

The different integration scenarios are as follows:

Compose Mail

[Inbox](#)
[Compose mail](#)
[Sent Items](#)
[Trash](#)
[Spam](#)
[Contact](#)
[Folders](#)
[Logout](#)

To

From

Subject

TEXT FIELD

Save To Draft
 Add To Contact

Scenarios1:

- First, we login as **P** users and click on the **Compose** mail and performing the functional testing for the specific components.
- Now we click on the **Send** and also check for **Save Drafts**.
- After that, we send a **mail** to **Q** and verify in the **Sent Items** folder of **P** to check if the send mail is there.
- Now, we will **log out** as **P** and login as **Q** and move to the **Inbox** and verify that if the mail has reached.

Secanrios2: We also perform the integration testing on **Spam** folders. If the particular contact has been marked as spam, then any mail sent by that user should go to the spam folder and not in the inbox.

Note: We will perform functional testing for all features, such as to send items, inbox, and so on.

As we can see in the below image, we will perform the **functional testing** for all the **text fields and every feature**. Then we will perform **integration testing** for the related functions. We first test the **add user, list of users, delete user, edit user**, and then **search user**.

Add Users

[Add User](#)
[Delete User](#)
[List User](#)
[Edit User](#)
[Product Sales](#)
[Product Purchases](#)
[Search Users](#)
[Help](#)

User Name

Password

Designation

Team Lead
 Manager
.....
.....

Email

Telephone

Address

Note:

- There are some features, we might be performing only the **functional testing**, and there are some features where we are performing both **functional** and **integration testing** based on the feature's requirements.
- **Prioritizing is essential**, and we should perform it at all the phases, which means we will open the application and select which feature needs to be tested first. Then go to that feature and choose which component must be tested first. Go to those components and determine what values to be entered first. And don't apply the same rule everywhere because testing logic varies from feature to feature.
- While performing testing, we should test one feature entirely and then only proceed to another function.
- Among the two features, we must be performing **only positive integrating testing** or both **positive and negative integration** testing, and this also depends on the features need.

Reason Behind Integration Testing

Although all modules of software application already tested in unit testing, errors still exist due to the following reasons:

1. Each module is designed by individual software developer whose programming logic may differ from developers of other modules so; integration testing becomes essential to determine the working of software modules.
2. To check the interaction of software modules with the database whether it is an erroneous or not.
3. Requirements can be changed or enhanced at the time of module development. These new requirements may not be tested at the level of unit testing hence integration testing becomes mandatory.
4. Incompatibility between modules of software could create errors.
5. To test hardware's compatibility with software.
6. If exception handling is inadequate between modules, it can create bugs.

Integration Testing Techniques

Any testing technique (Blackbox, Whitebox, and Greybox) can be used for Integration Testing; some are listed below:

Black Box Testing

- State Transition technique
- Decision Table Technique
- Boundary Value Analysis
- All-pairs Testing
- Cause and Effect Graph
- Equivalence Partitioning
- Error Guessing

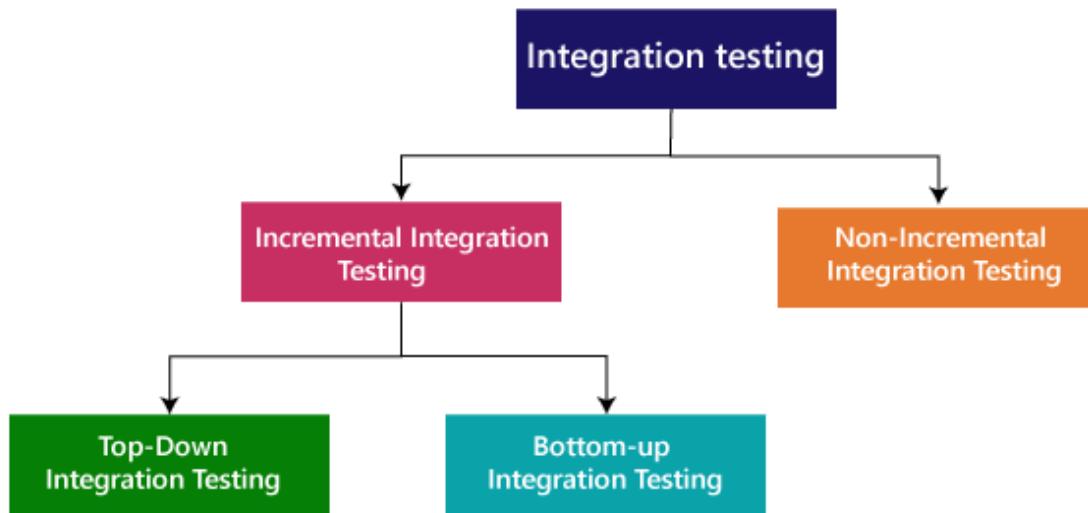
White Box Testing

- Data flow testing
- Control Flow Testing
- Branch Coverage Testing
- Decision Coverage Testing

Types of Integration Testing

Integration testing can be classified into two parts:

- **Incremental integration testing**
- **Non-incremental integration testing**

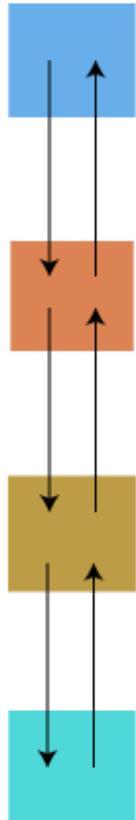


Incremental Approach

In the Incremental Approach, modules are added in ascending order one by one or according to need. The selected modules must be logically related. Generally, two or more than two modules are added and tested to determine the correctness of functions. The process continues until the successful testing of all the modules.

OR

In this type of testing, there is a strong relationship between the dependent modules. Suppose we take two or more modules and verify that the data flow between them is working fine. If it is, then add more modules and test again.



For example: Suppose we have a Flipkart application, we will perform incremental integration testing, and the flow of the application would like this:

Flipkart → Login → Home → Search → Add cart → Payment → Logout

Incremental integration testing is carried out by further methods:

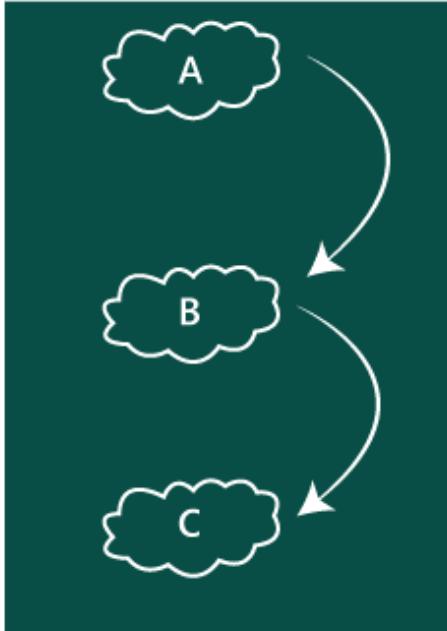
- Top-Down approach

- Bottom-Up approach

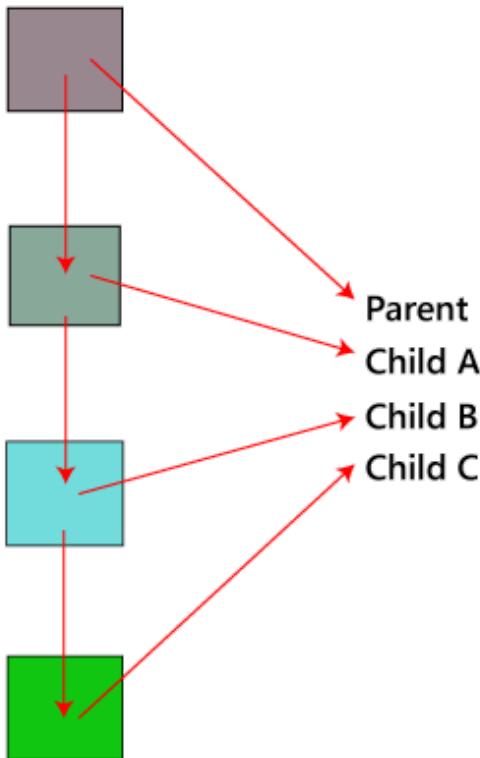
Top-Down Approach

The top-down testing strategy deals with the process in which higher level modules are tested with lower level modules until the successful completion of testing of all the modules. Major design flaws can be detected and fixed early because critical modules tested first. In this type of method, we will add the modules incrementally or one by one and check the data flow in the same order.

Top-Down Approach



In the top-down approach, we will be ensuring that the module we are adding is the **child of the previous one like Child C is a child of Child B** and so on as we can see in the below image:



Advantages:

- Identification of defect is difficult.
- An early prototype is possible.

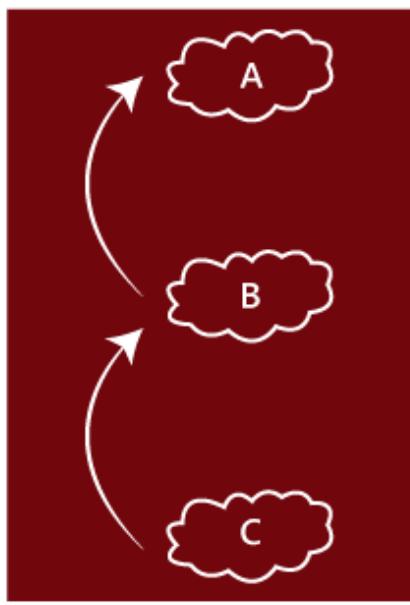
Disadvantages:

- Due to the high number of stubs, it gets quite complicated.
- Lower level modules are tested inadequately.
- Critical Modules are tested first so that fewer chances of defects.

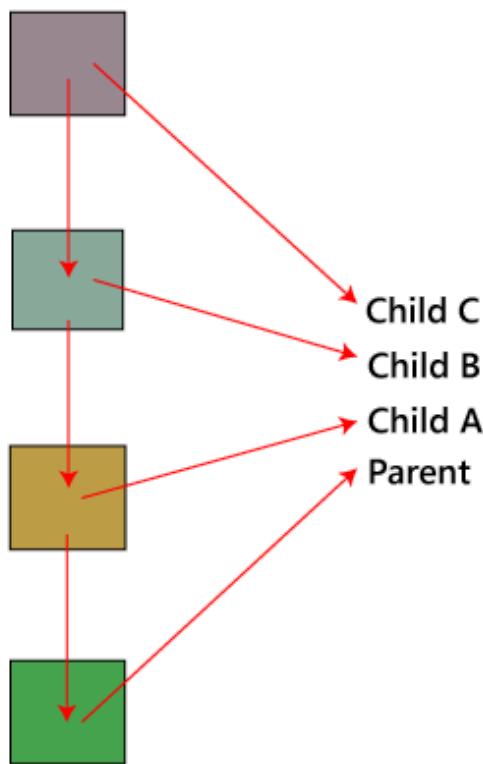
Bottom-Up Method

The bottom to up testing strategy deals with the process in which lower level modules are tested with higher level modules until the successful completion of testing of all the modules. Top level critical modules are tested at last, so it may cause a defect. Or we can say that we will be adding the modules from **bottom to the top** and check the data flow in the same order.

Bottom-up Approach



In the bottom-up method, we will ensure that the modules we are adding **are the parent of the previous one** as we can see in the below image:



Advantages

- Identification of defect is easy.
- Do not need to wait for the development of all the modules as it saves time.

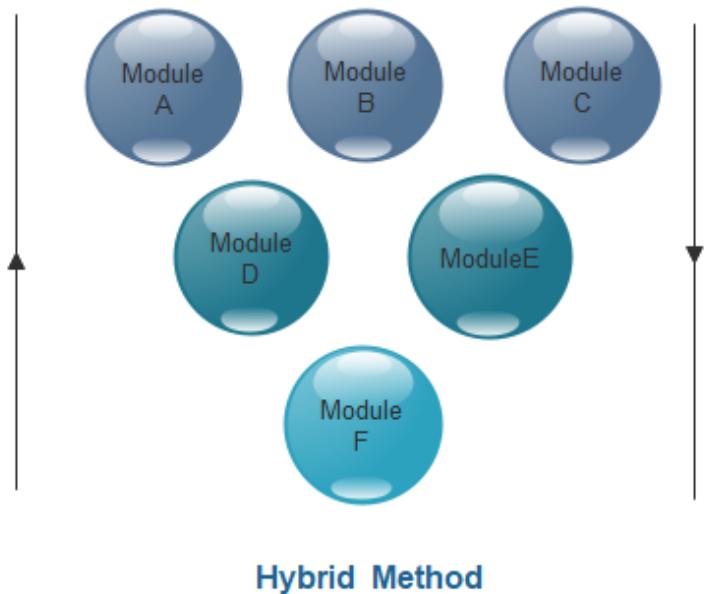
Disadvantages

- Critical modules are tested last due to which the defects can occur.
- There is no possibility of an early prototype.

In this, we have one addition approach which is known as **hybrid testing**.

Hybrid Testing Method

In this approach, both **Top-Down** and **Bottom-Up** approaches are combined for testing. In this process, top-level modules are tested with lower level modules and lower level modules tested with high-level modules simultaneously. There is less possibility of occurrence of defect because each module interface is tested.



Advantages

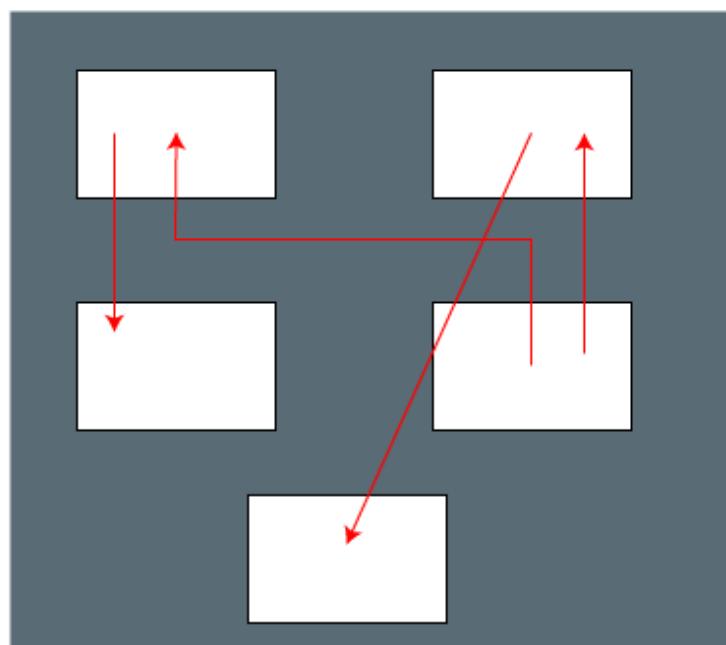
- The hybrid method provides features of both Bottom Up and Top Down methods.
- It is most time reducing method.
- It provides complete testing of all modules.

Disadvantages

- This method needs a higher level of concentration as the process carried out in both directions simultaneously.
- Complicated method.

Non- incremental integration testing

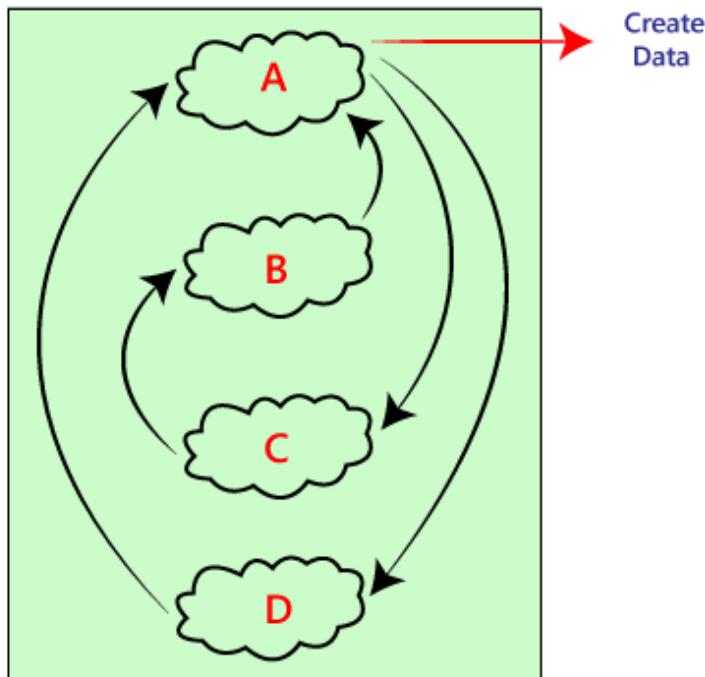
We will go for this method, when the data flow is very complex and when it is difficult to find who is a parent and who is a child. And in such case, we will create the data in any module bang on all other existing modules and check if the data is present. Hence, it is also known as the **Big bang method**.



Big Bang Method

In this approach, testing is done via integration of all modules at once. It is convenient for small software systems, if used for large software systems identification of defects is difficult.

Since this testing can be done after completion of all modules due to that testing team has less time for execution of this process so that internally linked interfaces and high-risk critical modules can be missed easily.



Advantages:

- It is convenient for small size software systems.

Disadvantages:

- Identification of defects is difficult because finding the error where it came from is a problem, and we don't know the source of the bug.
- Small modules missed easily.
- Time provided for testing is very less.
- We may miss to test some of the interfaces.

Let us see examples for our better understanding of the non-incremental integrating testing or big bang method:

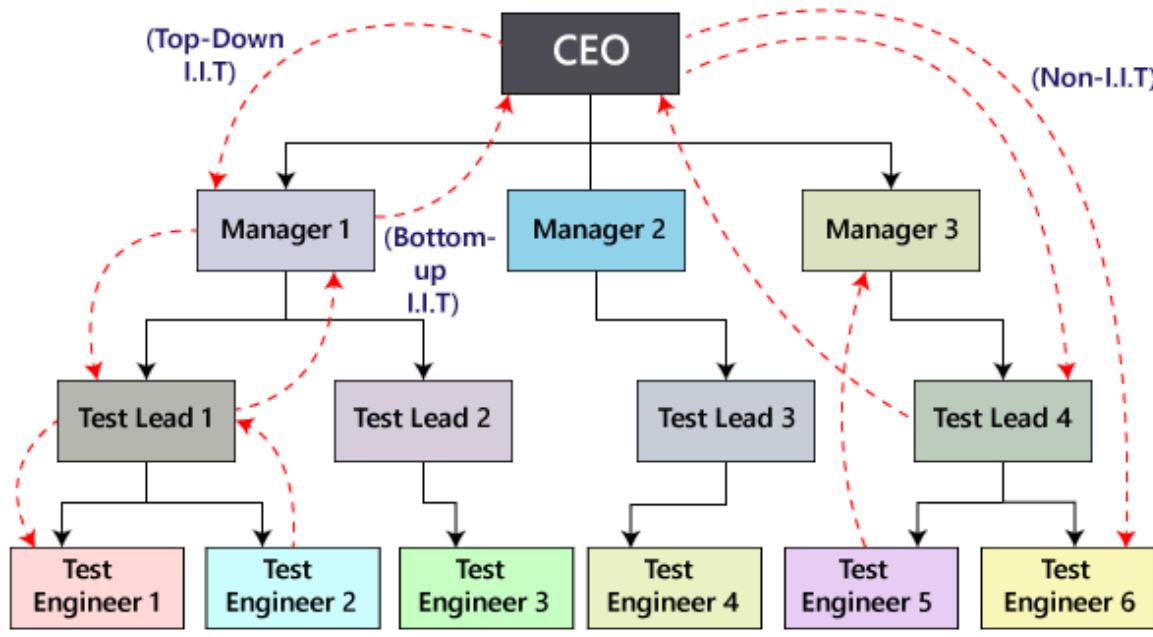
Example1

In the below example, the development team develops the application and sends it to the CEO of the testing team. Then the CEO will log in to the application and generate the username and password and send a mail to the manager. After that, the CEO will tell them to start testing the application.

Then the manager manages the username and the password and produces a username and password and sends it to the **test leads**. And the **test leads** will send it to the **test engineers** for further testing purposes. This order from the CEO to the test engineer is **top-down incremental integrating testing**.

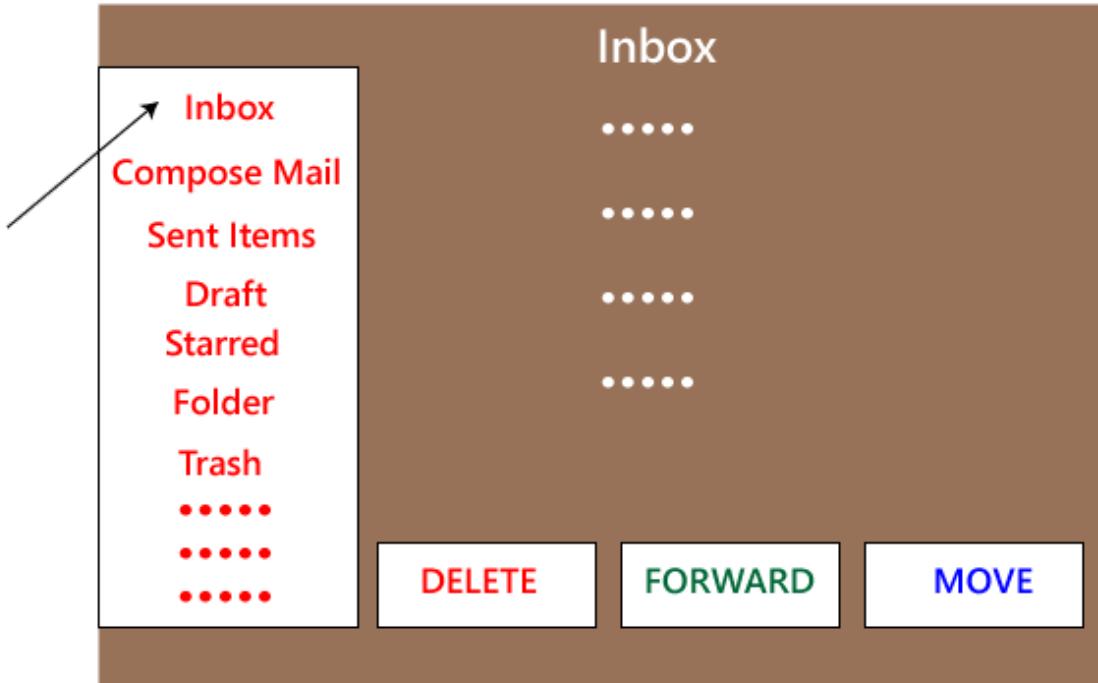
In the same way, when the test engineers are done with testing, they send a report to the **test leads**, who then submit a report to the **manager**, and the manager will send a report to the **CEO**. This process is known as **Bottom-up incremental integration testing** as we can see in the below image:

Note: The combination incremental integration testing (I.I.T) and non-incremental integration testing is known as sandwich testing.



Example2

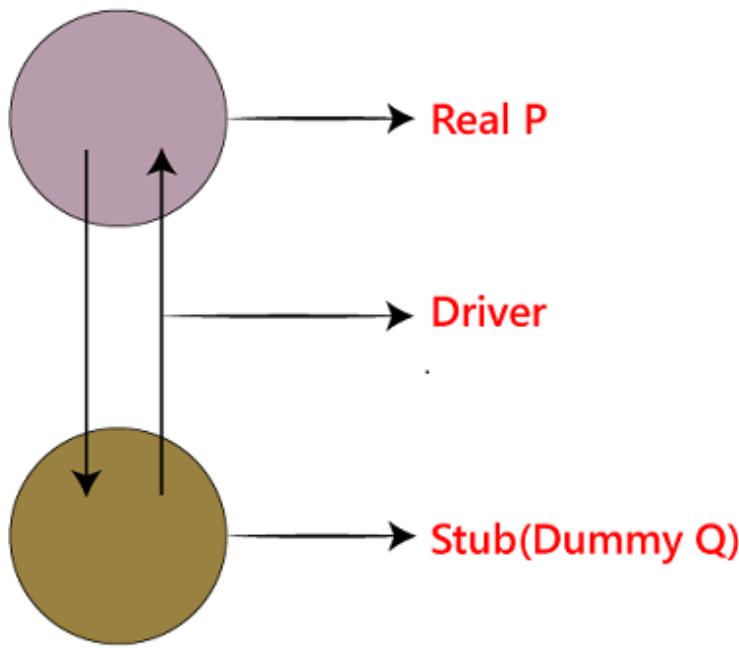
The below example demonstrates a home page of **Gmail's Inbox**, where we click on the **Inbox** link, and we are moved to the inbox page. Here we have to do **non- incremental integration testing** because there is no parent and child concept.



Note

Stub and driver

The **stub** is a dummy module that receives the data and creates lots of probable data, but it performs like a real module. When a data is sent from module P to Stub Q, it receives the data without confirming and validating it, and produce the estimated outcome for the given data.



The function of a driver is used to verify the data from P and sends it to stub and also checks the expected data from the stub and sends it to P.

The **driver** is one that sets up the test environments and also takes care of the communication, evaluates results, and sends the reports. We never use the stub and driver in the testing process.

In **White box testing**, **bottom-up integration testing** is ideal because writing drivers is accessible. And in **black box testing**, no preference is given to any testing as it depends on the application.

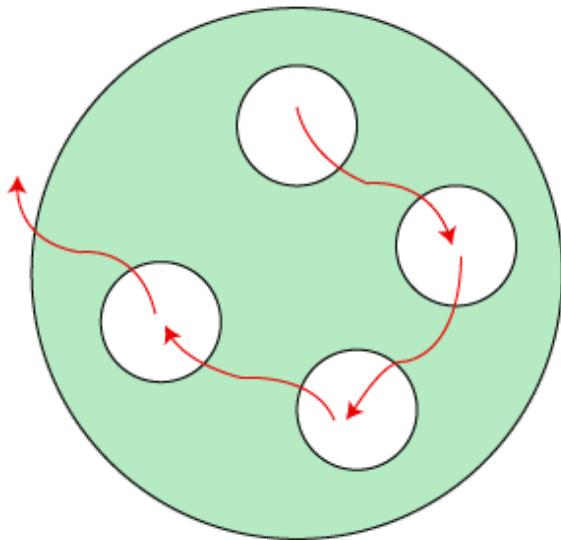
System Testing

System Testing includes testing of a fully integrated software system. Generally, a computer system is made with the integration of software (any software is only a single element of a computer system). The software is developed in units and then interfaced with other software and hardware to create a complete computer system. In other words, a computer system consists of a group of software to perform the various tasks, but only software cannot perform the task; for that software must be interfaced with compatible hardware. System testing is a series of different type of tests with the purpose to exercise and examine the full working of an integrated software computer system against requirements.



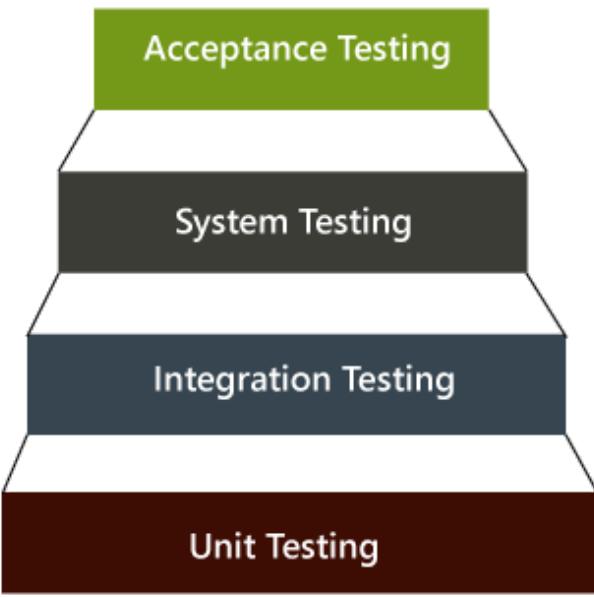
To check the end-to-end flow of an application or the software as a user is known as **System testing**. In this, we navigate (go through) all the necessary modules of an application and check if the end features or the end business works fine, and test the product as a whole system.

It is **end-to-end testing** where the testing environment is similar to the production environment.



There are four levels of **software testing**: **unit testing**, **integration testing**, **system testing** and **acceptance testing**, all are used for the testing purpose. Unit Testing used to test a single software; Integration Testing used to test a group of units of software, System Testing used to test a whole system and Acceptance Testing used to test the acceptability of business requirements. Here we are discussing system testing which is the third level of testing levels.

Hierarchy of Testing Levels



There are mainly two widely used methods for software testing, one is **White box testing** which uses internal coding to design test cases and another is **black box testing** which uses GUI or user perspective to develop test cases.

- White box testing
- Black box testing

System testing falls under Black box testing as it includes testing of the external working of the software. Testing follows user's perspective to identify minor defects.

System Testing includes the following steps.

- Verification of input functions of the application to test whether it is producing the expected output or not.
- Testing of integrated software by including external peripherals to check the interaction of various components with each other.
- Testing of the whole system for End to End testing.
- Behavior testing of the application via a user's experience

Example of System testing

Suppose we open an application, let say www.rediff.com, and there we can see that an advertisement is displayed on the top of the homepage, and it remains there for a few seconds before it disappears. These types of Ads are done by the Advertisement Management System (AMS). Now, we will perform system testing for this type of field.

The below application works in the following manner:

- Let's say that Amazon wants to display a promotion ad on January 26 at precisely 10:00 AM on the Rediff's home page for the country India.
- Then, the sales manager logs into the website and creates a request for an advertisement dated for the above day.
- He/she attaches a file that likely an image files or the video file of the AD and applies.
- The next day, the AMS manager of Rediffmail login into the application and verifies the awaiting Ad request.
- The AMS manager will check those Amazons ad requests are pending, and then he/she will check if the space is available for the particular date and time.
- If space is there, then he/she evaluate the cost of putting up the Ad at 15\$ per second, and the overall Ad cost for 10 seconds is approximate 150\$.
- The AMS manager clicks on the payment request and sends the estimated value along with the request for payment to the Amazon manager.

- Then the amazon manager login into the Ad status and confirms the payment request, and he/she makes the payment as per all the details and clicks on the **Submit** and **Pay**
- As soon as Rediff's AMs manager gets the amount, he/she will set up the Advertisement for the specific date and time on the Rediffmail's home page.

The various system test scenarios are as follows:

Scenario1: The first test is the general scenario, as we discussed above. The test engineer will do the system testing for the underlying situation where the Amazon manager creates a request for the Ad and that Ad is used at a particular date and time.

Scenario2: Suppose the Amazon manager feels that the AD space is too expensive and cancels the request. At the same time, the Flipkart requests the Ad space on January 26 at 10:00 AM. Then the request of Amazon has been canceled. Therefore, Flipkart's promotion ad must be arranged on January 26 at 10 AM.

After all, the request and payment have been made. Now, if Amazon changes their mind and they feel that they are ready to make payment for January 26 at 10 AM, which should be given because Flipkart has already used that space. Hence, another calendar must open up for Amazon to make their booking.

Scenario3: in this, first, we login as AMS manger, then click on Set Price page and set the price for AD space on logout page to 10\$ per second.

Then login as Amazon manager and select the date and time to put up and Ad on the logout page. And the payment should be 100\$ for 10 seconds of an Ad on Rediffmail logout page.

Name	Status	Total Amount
Amazon	Available	150\$
Flipkart	Available	100\$
....
....
Credit Card No.	<input type="text"/>	
....
....
....
SUBMIT	CANCEL	

REQUEST FOR ADVERTISEMENT

Product	<input type="text" value="Amazon"/>
Country	<input type="text" value="India"/> <input type="button" value="▼"/>
	China Russia U.S.A
Page	<input type="text" value="Inbox"/> <input type="button" value="▼"/>
	Sent Items Compose Mail Logout
Duration	<input type="text" value="10 Second"/>
Date	<input type="text" value="26th January 10:00AM"/>
Attach Advertisement [Text,Audio,Image,Video]	
SUBMIT CANCEL	

PENDING ADVERTISEMENT		
NAME	STATUS	SELECT
....	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
Pending Ads	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
....	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
Set Rates	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
Amazon	Pending	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
Flipkart	Pending	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
....	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
....	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
....	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
....	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
CHECK AVAILABILITY	PAYMENT REQUEST	APPROXIMATE
		SETUP

SET PRICE

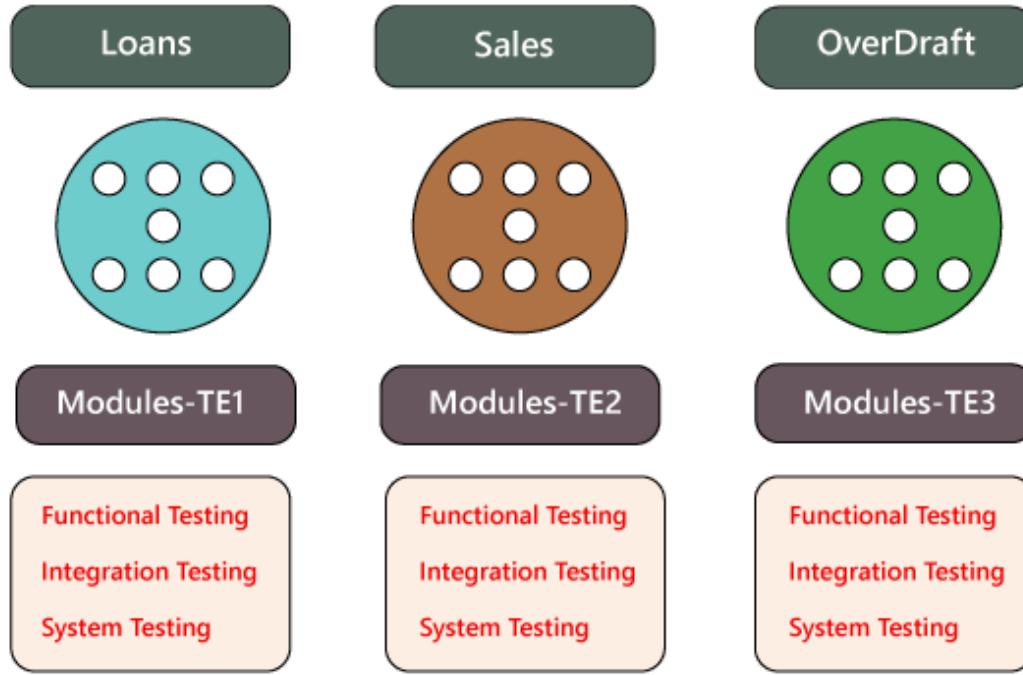
Country	<input type="text" value="India"/> <input type="button" value="▼"/>
Page	<input type="text" value="Logout"/> <input type="button" value="▼"/>
Duration	<input type="text" value="10 Seconds"/>
Amount	<input type="text" value="10\$"/>
SUBMIT CANCEL	

Amazon Advertisement
Available at 10:00AM on
January 26th

Note: Generally, every test engineer does the functional, integration, and system testing on their assigned module only.

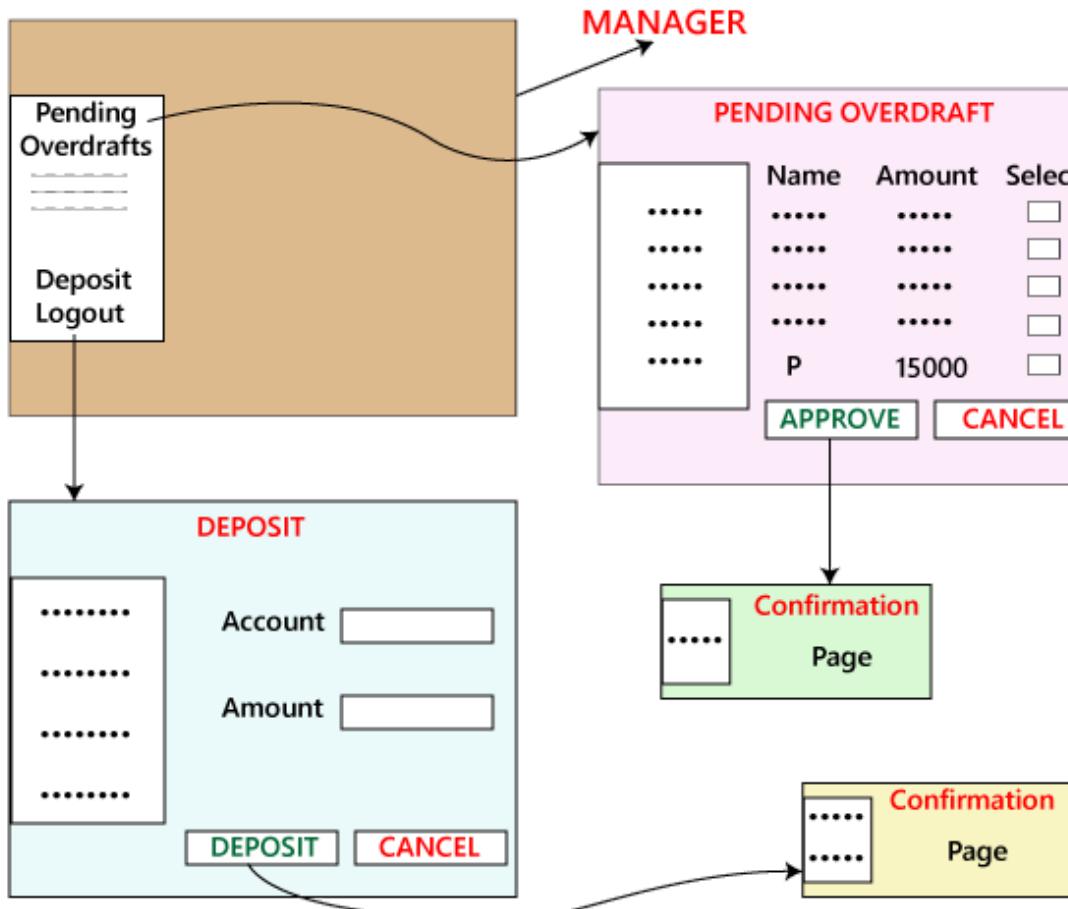
As we can see in the below image, we have three different modules like **Loans**, **Sales**, and **Overdraft**. And these modules are going to be tested by their assigned test engineers only because if data flow between these modules or scenarios, then we need to clear that in which module it is going and that test engineer should check that thing.

Let us assume that here we are performing system testing on the interest estimation, where the customer takes the Overdraft for the first time as well as for the second time.



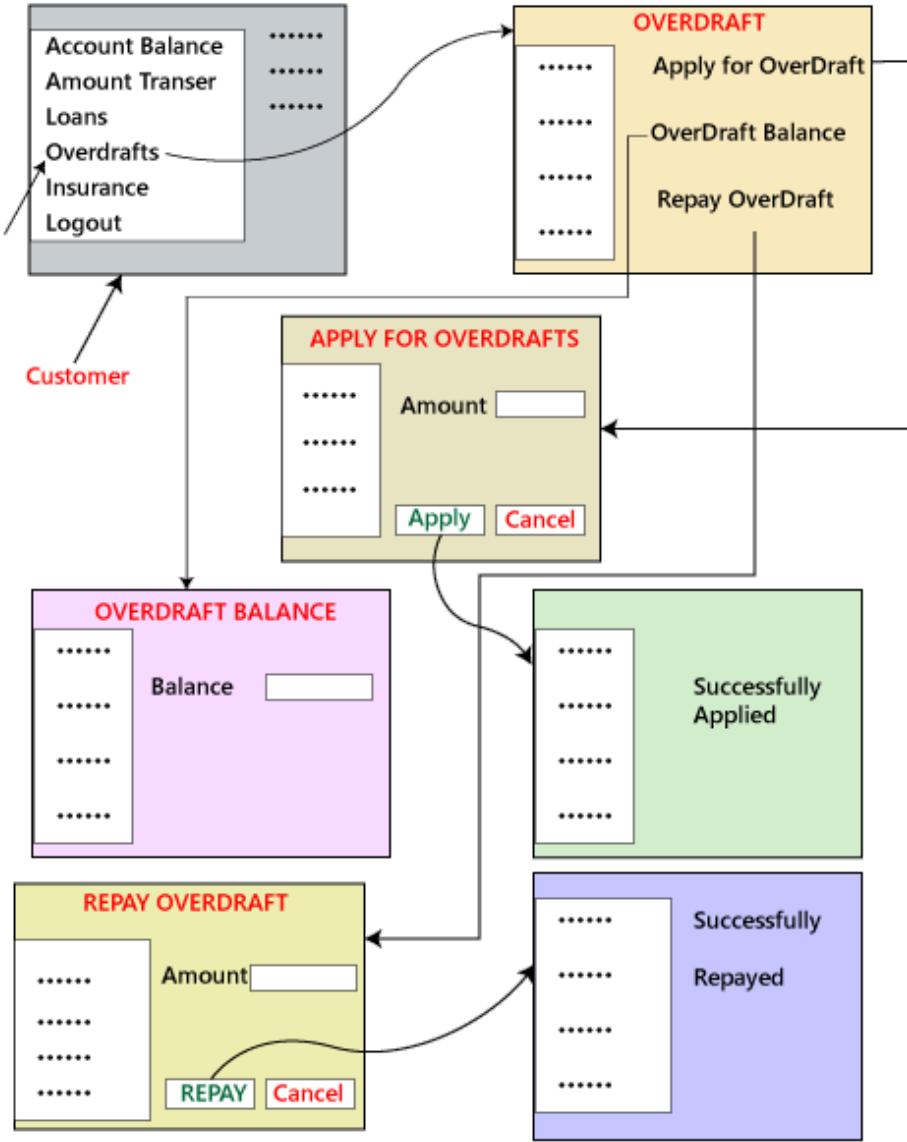
In this particular example, we have the following scenarios:

Scenarios 1



- First, we will log in as a User; let see P, and apply for Overdraft Rs15000, click on apply, and logout.
- After that, we will log in as a Manager and approve the Overdraft of P, and logout.
- Again we will log in as a P and check the Overdraft balance; Rs15000 should be deposited and logout.
- Modify the server date to the next 30 days.
- Login as P, check the Overdraft balance is $15000 + 300 + 200 = 15500$, than logout
- Login as a Manager, click on the Deposit, and Deposit Rs500, logout.

- Login as P, Repay the Overdraft amount, and check the Overdraft balance, which is Rs zero.
- Apply for Overdraft in Advance as a two-month salary.
- Approve by the Manager, amount credit and the interest will be there to the processing fee for 1st time.
- Login user → Homepage [Loan, Sales, Overdraft] → Overdraft page [Amount Overdraft, Apply Overdraft, Repay Overdraft] → Application
- Login manager → Homepage [Loan, Sales, Overdraft] → Overdraft page [Amount Overdraft, Apply Overdraft, Repay Overdraft, Approve Overdraft] → Approve Page → Approve application.
- Login as user P → Homepage [Loan, Sales, Overdraft] → Overdraft page [Amount Overdraft, Apply Overdraft, Repay Overdraft] → Approved Overdraft → Amount Overdraft
- Login as user P → Homepage [Loan, Sales, Overdraft] → Overdraft page [Amount Overdraft, Apply Overdraft, Repay Overdraft] → Repay Overdraft → with process fee + interest amount.



Scenario 2

Now, we test the alternative scenario where the bank provides an offer, which says that a customer who takes Rs45000 as Overdraft for the first time will not charge for the Process fee. The processing fee will not be refunded when the customer chooses another overdraft for the third time.

We have to test for the third scenario, where the customer takes the Overdraft of Rs45000 for the first time, and also verify that the Overdraft repays balance after applying for another overdraft for the third time.

Scenario 3

In this, we will reflect that the application is being used generally by all the clients, all of a sudden the bank decided to reduce the processing fee to Rs100 for new customer, and we have test Overdraft for new clients and check whether it is accepting only for Rs100.

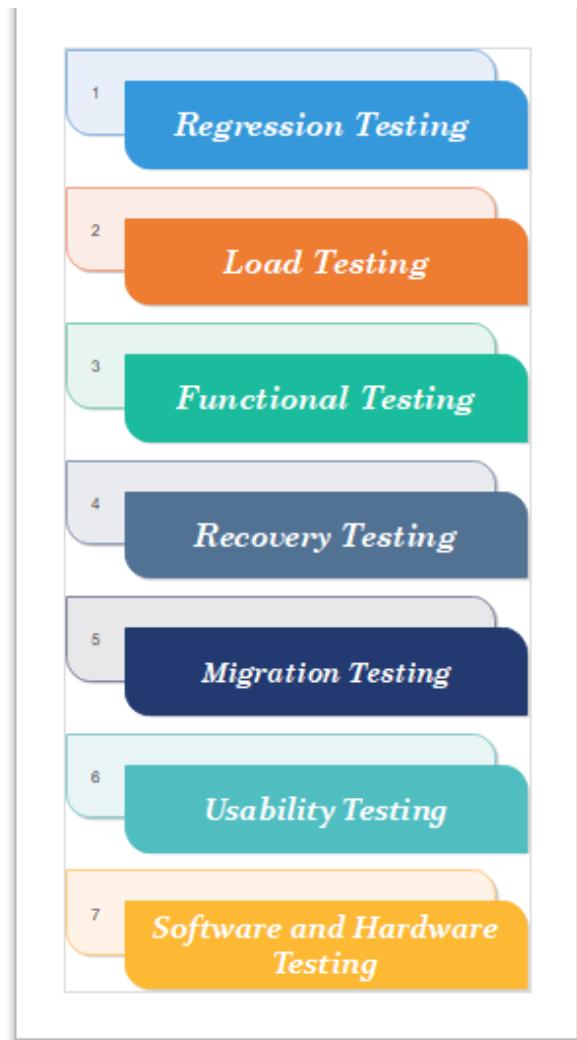
But then we get conflicts in the requirement, assume the client has applied for Rs15000 as Overdraft with the current process fee for Rs200. Before the Manager is yet to approve it, the bank decreases the process fee to Rs100.

Now, we have to test what process fee is charged for the Overdraft for the pending customer. And the testing team cannot assume anything; they need to communicate with the Business Analyst or the Client and find out what they want in those cases.

Therefore, if the customers provide the first set of requirements, we must come up with the maximum possible scenarios.

Types of System Testing

System testing is divided into more than 50 types, but software testing companies typically uses some of them. These are listed below:



Regression Testing

Regression testing is performed under system testing to confirm and identify that if there's any defect in the system due to modification in any other part of the system. It makes sure, any changes done during the development process have not introduced a new defect and also gives assurance; old defects will not exist on the addition of new software over the time.

For more information about regression testing refers to the below link:

<https://www.javatpoint.com/regression-testing>

Load Testing

Load testing is performed under system testing to clarify whether the system can work under real-time loads or not.

Functional Testing

Functional testing of a system is performed to find if there's any missing function in the system. Tester makes a list of vital functions that should be in the system and can be added during functional testing and should improve quality of the system.

Recovery Testing

Recovery testing of a system is performed under system testing to confirm reliability, trustworthiness, accountability of the system and all are lying on recouping skills of the system. It should be able to recover from all the possible system crashes successfully.

In this testing, we will test the application to check how well it recovers from the crashes or disasters.

Recovery testing contains the following steps:

- Whenever the software crashes, it should not vanish but should write the **crash log message or the error log message** where the reason for crash should be mentioned. **For example:** **C://Program Files/QTP/Cresh.log**
- It should kill its own procedure before it vanishes. Like, in Windows, we have the Task Manager to show which process is running.
- We will introduce the bug and crash the application, which means that someone will lead us to how and when will the application crash. Or **By experiences**, after few months of involvement on working the product, we can get to know how and when the application will crash.
- Re-open the application; the application must be reopened with earlier settings.

For example: Suppose, we are using the Google Chrome browser, if the power goes off, then we switch on the system and re-open the Google chrome, we get a message asking whether we want to **start a new session** or **restore the previous session**. For any developed product, the developer writes a recovery program that describes, why the software or the application is crashing, whether the crash log messages are written or not, etc.

Migration Testing

Migration testing is performed to ensure that if the system needs to be modified in new infrastructure so it should be modified without any issue.

Usability Testing

The purpose of this testing to make sure that the system is well familiar with the user and it meets its objective for what it supposed to do.

For more information about usability testing refers to the below link:

<https://www.javatpoint.com/usability-testing>

Software and Hardware Testing

This testing of the system intends to check **hardware** and **software** compatibility. The hardware configuration must be compatible with the software to run it without any issue. Compatibility provides flexibility by providing interactions between hardware and software.

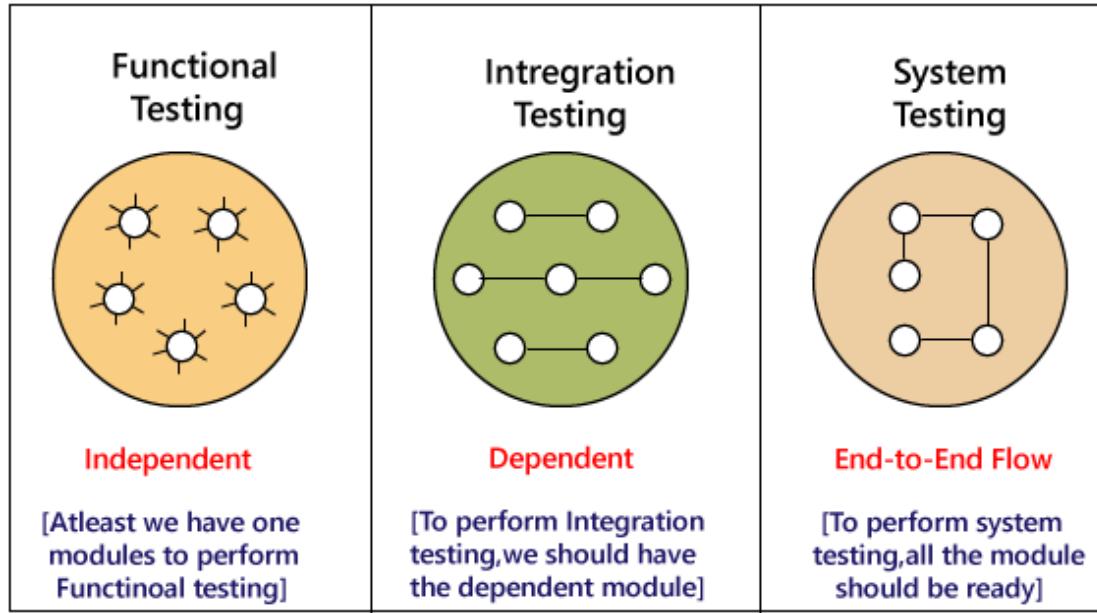
Why is System Testing Important?

- System Testing gives hundred percent assurance of system performance as it covers end to end function of the system.
- It includes testing of System software architecture and business requirements.
- It helps in mitigating live issues and bugs even after production.
- System testing uses both existing system and a new system to feed same data in both and then compare the differences in functionalities of added and existing functions so, the user can understand benefits of

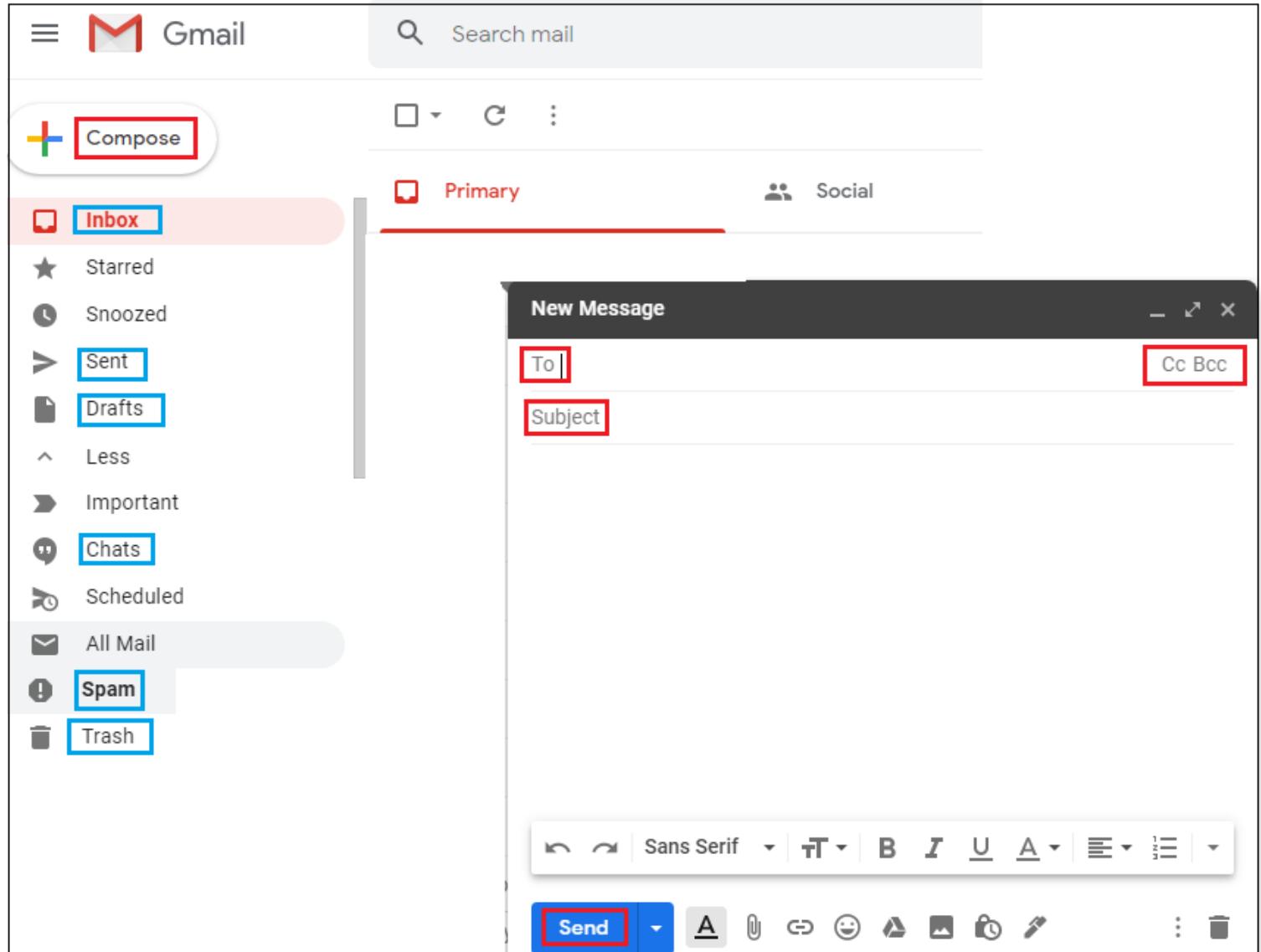
new added functions of the system.

Testing Any Application

Here, we are going to test the **Gmail** application to understand how **functional, integration, and System testing** works.



Suppose, we have to test the various modules such as **Login, Compose, Draft, Inbox, Sent Item, Spam, Chat, Help, Logout** of **Gmail** application.



We do **Functional Testing** on all Modules First, and then only we can perform integration testing and system testing.

In functional testing, at least we have one module to perform functional testing. So here we have the Compose Module where we are performing the functional testing.

Compose

The different components of the Compose module are **To**, **CC**, **BCC**, **Subject**, **Attachment**, **Body**, **Sent**, **Save to Draft**, **Close**.

- First, we will do functional testing on the **To**

Input	Results
Positive inputs	
mike@gmail.com	Accept
Mike12@gmail.com	Accept
Mike@yahoo.com	Accept
Negative inputs	
Mike@yahoocom	Error
Mike@yahoo.com	Error

- For **CC & BCC** components, we will take the same input as **To component**.
- For **Subject** component, we will take the following inputs and scenarios:

Input	Results
Positive inputs	
Enter maximum character	Accept
Enter Minimum character	Accept
Blank Space	Accept
URL	Accept
Copy & Paste	Accept
Negative inputs	
Crossed maximum digits	Error
Paste images / video / audio	Error

- Maximum character**
- Minimum character**
- Flash files (GIF)**
- Smiles**
- Format**
- Blank**
- Copy & Paste**
- Hyperlink**
- Signature**

- For the **Attachment** component, we will take the help of the below scenarios and test the component.
 - **File size at maximum**
 - **Different file formats**
 - **Total No. of files**
 - **Attach multiple files at the same time**
 - **Drag & Drop**
 - **No Attachment**
 - **Delete Attachment**
 - **Cancel Uploading**
 - **View Attachment**
 - **Browser different locations**
 - **Attach opened files**
- For **Sent** component, we will write the entire field and click on the **Sent** button, and the Confirmation message; **Message sent successfully** must be displayed.
- For **Saved to Drafts** component, we will write the entire field and click on **aved to drafts**, and the Confirmation message must be displayed.
- For the **Cancel** component, we will write all fields and click on the Cancel button, and the **Window will be closed** or moved to **save to draft** or all fields must be refreshed.

Once we are done performing functional testing on compose module, we will do the Integration testing on Gmail application's various modules:

Login

- First, we will enter the username and password for login to the application and Check the username on the Homepage.

Compose

- Compose mail, send it and check the mail in Sent Item [sender]
- Compose mail, send it and check the mail in the receiver [Inbox]
- Compose mail, send it and check the mail in self [Inbox]
- Compose mail, click on Save as Draft, and check-in sender draft.
- Compose mail, send it invalid id (valid format), and check for undelivered message.
- Compose mail, close and check-in Drafts.

Inbox

- Select the mail, reply, and check in sent items or receiver Inbox.
- Select the mail in Inbox for reply, Save as Draft and check in the Draft.
- Select the mail then delete it, and check in Trash.

Sent Item

- Select the mail, Sent Item, Reply or Forward, and check in Sent item or receiver inbox.
- Select mail, Sent Item, Reply or Forward, Save as Draft, and verify in the Draft.
- Select mail, delete it, and check in the Trash.

Draft

- Select the email draft, forward and check Sent item or Inbox.
- Select the email draft, delete and verify in Trash.

Chat

- Chat with offline users saved in the inbox of the receiver.
- Chat with the user and verify it in the chat window.
- Chat with a user and check in the chat history.

Note: During testing, we need to wait for a particular duration of time because system testing can only be performed when all the modules are ready and performed functional and integration testing.

Performance Testing

In this section, we will learn about performance testing, why we need it, types of performance testing, and the performance testing process.

Following are the topics, which we will understand in this section:

What is performance testing?

It is the most important part of non-functional testing.

“ Checking the behavior of an application by applying some load is known as performance testing. ”

Generally, this testing defines how quickly the server responds to the user's request.

While doing performance testing on the application, we will concentrate on the various factors like **Response time**, **Load**, and **Stability** of the application.

Response time: Response time is the time taken by the server to respond to the client's request.

Load: Here, Load means that when **N-number** of users using the application simultaneously or sending the request to the server at a time.

Stability: For the stability factor, we can say that, when N-number of users using the application simultaneously for a particular time.

When we use performance testing?

We will do performance testing once the software is stable and moved to the production, and it may be accessed by the multiple users concurrently, due to this reason, some performance issues may occur. To avoid these performance issues, the tester performs one round of performance testing.

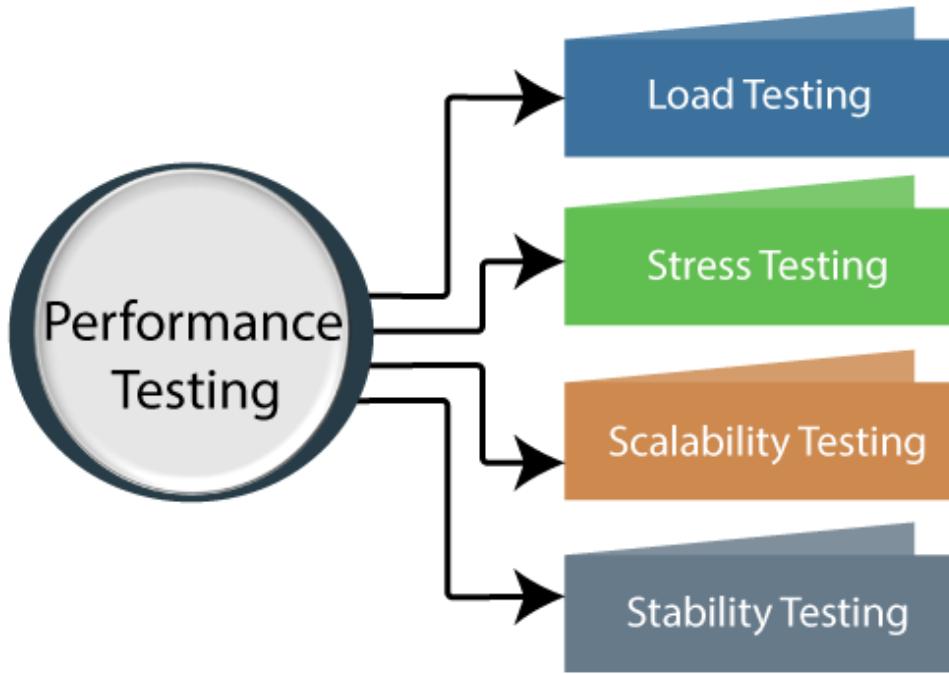
Since it is non-functional testing which doesn't mean that we always use performance testing, we only go for performance testing when the application is functionally stable.

Note: Performance testing cannot be done manually since its costly and accurate result can't be maintained.

Types of Performance Testing

Following are the types of performance testing:

- **Load testing**
- **Stress testing**
- **Scalability testing**
- **Stability testing**

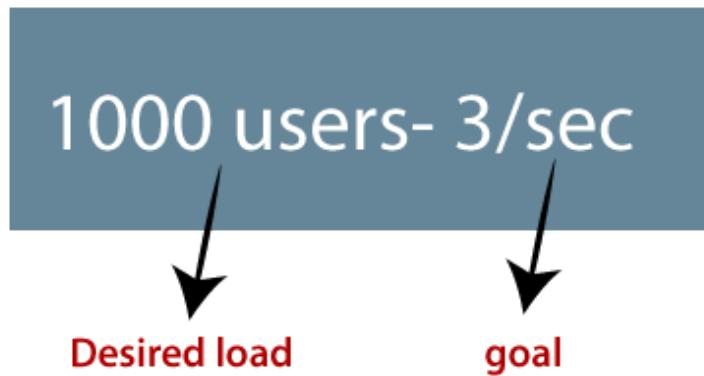


Let us discuss one by one to give you a complete understanding of **Load**, **Stress**, **Scalability**, and **Stability** performance testing.

Load testing

The load testing is used to check the performance of an application by applying some load which is either less than or equal to the desired load is known as load testing.

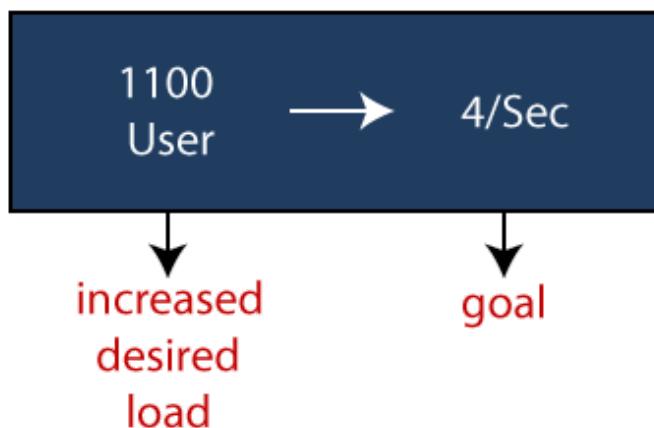
For example: In the below image, **1000 users** are the **desired load**, which is given by the customer, and **3/second** is the **goal** which we want to achieve while performing a load testing.



Stress Testing

The stress testing is testing, which checks the behavior of an application by applying load greater than the desired load.

For example: If we took the above example and increased the desired load 1000 to 1100 users, and the goal is 4/second. While performing the stress testing in this scenario, it will pass because the load is greater (100 up) than the actual desired load.



Scalability Testing

Checking the performance of an application by increasing or decreasing the load in particular scales (no of a user) is known as **scalability testing**. Upward scalability and downward scalability testing are called scalability testing.

Scalability testing is divided into two parts which are as follows:

- **Upward scalability testing**
- **Downward scalability testing**

Upward scalability testing

It is testing where we **increase the number of users on a particular scale** until we get a crash point. We will use upward scalability testing to find the maximum capacity of an application.

Downward scalability testing

The downward scalability testing is used when the load testing is not passed, then start **decreasing the no. of users in a particular interval** until the goal is achieved. So that it is easy to identify the bottleneck (bug).

Stability Testing

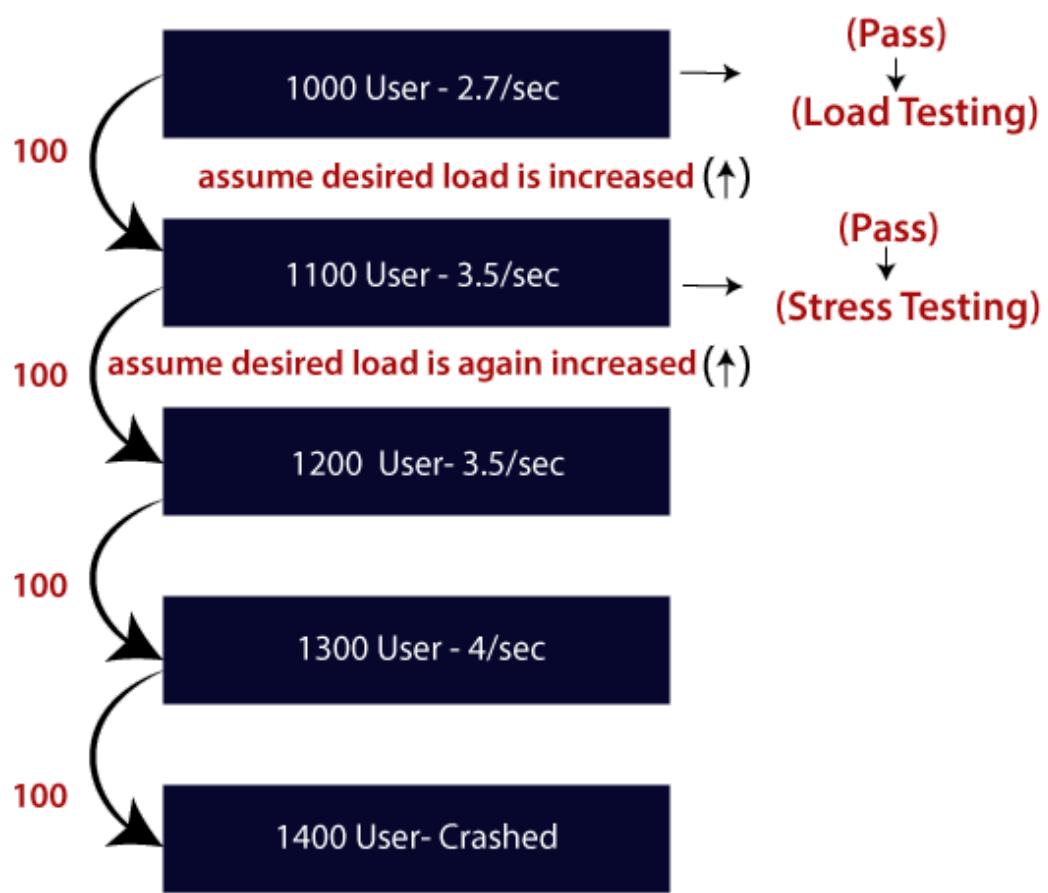
Checking the performance of an application by **applying the load for a particular duration of time** is known as **Stability Testing**.

Performance testing example

Let us take one example where we will **test the behavior of an application where the desired load is either less than 1000 or equal to 1000 users**.

In the below image, we can see that the **100 up** users are increased continuously to check the **maximum load**, which is also called **upward scalability testing**.

- **Scenario1:** When we have the 1000 users as desired load, and the 2.7/sec is goal time, these scenarios will pass while performing the load test because in load testing, we will concentrate on the no. of users, and as per the requirement it is equal to 1000 user.
- **Scenario2:** In the next scenario, we will increase the desired load by 100 users, and goal time will go up to 3.5\sec. This scenario will pass if we perform stress testing because here, the actual load is greater than (1100) the desired load (1000).
- **Scenario3:** In this, if we increase the desired load three times as **1200 → 3.5\sec**: [it is not less than or equal to the desired load that's why it will **Fail**] **1300 → 4\sec**: [it is not less than or equal to the desired load. i.e., **Fail**] **1400 → Crashed**



Note1: Volume and soak testing is a type of testing but not performance testing.

Volume testing

Volume testing is testing, which helps us to check the behavior of an application by inserting a massive volume of the load in terms of data is known as volume testing, and here, we will concentrate on the number of data rates than the number of users.

Note2: Volume is a capacity while Load is a quantity, i.e., load testing means no. of users, and volume testing means amount of data.

Soak testing

In this type of testing, we will check the behavior of an application on the environment, which is unsupportive for a long duration of time is known as soak testing.

Generally, soak testing is a negative type of testing since we already know that the server or environment is not supportive.

Performance testing process

The performance testing cannot be done manually since:

- We need a lot of resources, and it became a costlier approach.
- And the accuracy cannot maintain when we track response time manually.

The Performance testing process will be completed in the following steps:

- Identify performance scenarios
- Plan and design performance test script
- Configure the test environment & distribute the load

- Execute test scripts
- Result
- Analysis result
- Identify the Bottleneck
- Re-run test



If we perform a **positive flow** of the performance testing process, it could follow the below process:

Identify performance scenarios

Firstly, we will identify the performance scenarios based on these below factors:

Most commonly scenarios: It means that we can find the performance scenarios based on the scenarios, which commonly used like in the **Gmail application**; we will perform **login, inbox, send items, and compose a mail and logout**.

Most critical scenarios: Critical scenarios mean regularly used and important for the business-like in Gmail application **login, compose, inbox, and logout**.

Huge data transaction: If we have huge data means that n-number of the users using the application at the same time.

Once we identify the performance scenarios, we will move to the next step.

Plan and design performance test script

In this step, we will install the tools in the Test Engineer Machine and access the test server and then we write some script according to the test scenarios and run the tool.

Once we are done with writing the script, we will go to the next step.

Configure the test environment & distribute the load

After writing the test scripts, we will arrange the testing environment before the execution. And also, manage the tools, other resources and distribute the load according to the "Usage Pattern" or mention the duration and stability.

Execute test scripts

Once we are done with distributing the load, we will execute, validate, and monitor the test scripts.

Result

After executing the test scripts, we will get the test result. And check that the result meeting the goal in the given response time or not, and the response time could be maximum, average, and minimum.

If the response is not meeting the required time response, then we will go for the **negative flow** where will perform the below steps:

Analysis result

First, we will analyze the test result whether it meets with the response time or not.

Identify the Bottleneck

After that, we will identify the **bottleneck (bug or performance issue)**. And the bottleneck could occur because of these aspects like the **problem in code, hardware issue (hard disk, RAM Processor), network issues, and the software issue (operating system)**. And after finding the bottleneck, we will perform **tuning (fix or adjustment)** to resolve this bottleneck.

Re-run test

Once we fix the bottlenecks, re-run the test scripts and checks the result whether it meets the required goal or not.

The problem occurs in performance testing

While performing performance testing on the application, some problems may occur, and these problems are also called the **performance issue**.

The performance issues are as follows:

- **Response time issue**
- **Scalability issue**
- **Bottleneck**
- **Speed issue**

Response time issue

The response time means how quickly the server respond to the client's request. If the user's request does not complete in the given response time, it might have possible that the user may be lost his/her interest in the particular software or application. That's why the application or software should have a perfect response time for responding user's request quickly.

Scalability issue

The scalability issues occur when the application cannot take the n-numbers of users and expected user requests at the same time. That's why we will do **upward scalability testing** (check the maximum capacity of the application) and **downward scalability testing** (when expected time is not matched with the actual time).

Bottleneck

The Bottleneck is the informal name of bug, which occurs when the application is limited by a single component and creates a bad impact on the system performance.

The main causes of bottlenecks are **software issues (issue related to the operating system)**, **hardware issues (issues related to the hard disk, RAM and the processor)**, and **coding issue**, etc.

Following are the most common performance bottlenecks:

- Memory utilization
- Disk usage
- CPU utilization
- Operating System limitations
- Network utilization

Speed issues

When we perform performance testing on the application, the application should be faster in speed to get the user's interest and attention because if the application's speed is slow, it may lose the user interest in the application.

Performance test tools

We have various types of performance testing tools available in the market, where some are commercial tools and open-source tool.

Commercial tools: LoadRunner[HP], WebLOAD, NeoLoad

Open-source tool: JMeter

LoadRunner

It is one of the most powerful tools of performance testing, which is used to support the performance testing for the extensive range of protocols, number of technologies, and application environments.

It quickly identifies the most common causes of performance issues. And also accurately predict the application scalability and capacity.

JMeter

The Apache JMeter software is an open- source tool, which is an entirely a Java application designed to load the functional test behavior and measure the performance.

Generally, it was designed for testing the Web Applications but now expanded to other test functions also.

Apache JMeter is used to test performance for both static and dynamic resources and dynamic web applications. It can be used to reproduce the heavy load on a server, network or object, group of servers to test its strength or to analyze overall performance under different load types.

WebLOAD

WebLOAD testing tool used to test the load testing, performance testing, and stress test web applications.

The WebLOAD tool combines performance, scalability, and integrity as a single process for the verification of web and mobile applications.

NeoLoad

Neotys develop a testing tool which is called NeoLoad. The NeoLoad is used to test the performance test scenarios. With the help of NeoLoad, we can find the bottleneck areas in the web and the mobile app development process.

The NeoLoad testing tool is faster as compared to traditional tools.

Apart from them, some other tools are **Electric load**, **web stress tool**, **LoadUI Pro**, **StresStimulus**, **LoadView**, **LoadNinja**, and **RedLine13**, which helps to test the performance of the software or an application.

Usability Testing

Nowadays, we have n-numbers of application available in application store in order to help the people in their works.

And where they can gives a negative response or a poor rating, which leads a particular product towards their ends before it is downloaded or installed by a limited number of end-users.

In short, we can say that one bad review can damage all the resources skill, extended hours of planning, enthusiasm to develop the product, and so on.

That is why **Usability testing** comes into the picture to resolve these types of issues, as usability testing has a vibrant significance and is executed by the test engineers throughout the **STLC (Software Testing Life Cycle)**.

To help us understand its significance in STLC, in this section, we are going to discuss all about Usability Testing, which includes the following list of essential topics:

- **What is Usability Testing?**
- **Why do we need to perform Usability Testing?**
- **Features of Usability Testing**
- **Parameters covered by Usability Testing**
- **Various strategies of Usability Testing**
- **Usability Testing Process**
- **Examples of Usability Testing**
- **Usability Testing Checklist**
- **Bugs in Usability Testing**
- **Advantages of Usability Testing**
- **Disadvantages of Usability Testing**



What is Usability Testing?

Usability Testing is a significant **type of software testing** technique, which is comes under the **non-functional testing**.

It is primarily used in user-centered interaction design on order to check the usability or ease of using a software product. The implementation of usability testing requires an understanding of the application, as it is extensive testing.

Generally, usability testing is performed from an end-user viewpoint to verify if the system is efficiently working or not.

“ Checking the user-friendliness, efficiency, and accuracy of the application is known as Usability Testing. ”

The primary purpose of executing the usability testing is to check that the application should be easy to use for the end-user who is meant to use it, whereas sustaining the client's specified functional and business requirements.

When we use usability testing, it makes sure that the developed software is straightforward while using the system without facing any problem and makes end-user life easier.

In other words, we can say that Usability testing is one of the distinct testing techniques that identify the defect in the end-user communication of software product. And that's why it is also known as **User Experience (UX) Testing**.

It helps us to fix several usability problems in a specific website or application, even making sure its excellence and functionality.

The execution of usability testing certifies all the necessary features of a product, from testing the effortlessness of navigating a website and to validate its flow and the content in order to propose the best user experience.

Typically, the usability testing is executed by real-life users, not by the development team, as we are already aware of that the development team is the one who has created the product. Consequently, they fail to identify the more minor defects or bugs related to the user experience.

Note: It can be implemented in the Designing phase of the [software development life cycle \(SDLC\)](#) in order to help us getting more clarity of the user's needs.

In **Usability Testing**, the **user-friendliness** can be described with the help of the following characteristics:

- **Easy to understand**
- **Easy to access**
- **Look and feel**
- **Faster to Access**
- **Effective Navigation**
- **Good Error Handling**



Let us see them one by one for our better understanding:

Easy to understand

- All the features of software or applications must be visible to the end-users.

Easy to Access

- A user-friendly application should be accessible by everyone.

Easy to Access

- The look and feel of the application should be excellent and attractive to get the user's interest.
- The GUI of the software should be good because if the GUI is not well, the user may be lost his/her interest while using the application or the software.
- The quality of the product is up to the mark as given by the client.

Faster to Access

- The software should be faster while accessing, which means that the application's response time is quick.
- If the response time is slow, it might happen that the user got irritated. We have to ensure that our application will be loaded within 3 to 6 seconds of the response time.

Effective Navigation

- Effective navigation is the most significant aspect of the software. Some of the following aspects for effective navigation:
- Good Internal Linking
- Informative header and footer
- Good search feature

Good Error Handling

- Handling error at a coding level makes sure that the software or the application is bug-free and robust.
- By showing the correct error message will help to enhance the user experience and usability of the application.

Why do we need to perform Usability Testing?

We need usability testing because usability testing is to build a system with great user experience. Usability is not only used for **software development** or website development, but it is also used for product designing.

And Customers must be comfortable with your application with the following parameters.

- The flow of an Application should be good
- Navigation steps should be clear
- Content should be simple
- The layout should be clear
- Response time

And we can also test the different features in usability testing given as follows:

- How easy it is using the application
- How easy to learn application

Features of Usability Testing

The implementation of usability testing helps us to increase the end-user experience of the particular application and software. With the help of usability testing, the software development team can quickly detect several usability errors in the system and fix them quickly.

Some other vital features of usability testing are as follows:

- It is an essential type of **non-functional testing** technique, which comes under the **black-box testing** technique in **software testing**.
- Usability testing is performed throughout the **system** and **acceptance testing** levels.
- Generally, usability testing is implemented in the early stage of **the Software Development Life Cycle** (SDLC).
- The execution of usability testing offers more visibility on the prospects of the end-users.
- The usability testing makes sure that the software product meets its planned purpose.
- It also helps us to find many usability errors in the specified software product.
- Usability testing mainly tests the user-friendliness, usefulness, traceability, usability, and desirability of the final product.
- It offers direct input on how real-users use the software/application.
- The usability testing includes the systematic executions of the product's usability under a measured environment.

Parameters covered by Usability Testing

In order to test the **quality**, **usability**, **user-friendliness**, and other significant factors of the software, usability testing plays an important role. And it also helps us in order to supports the organizations for delivering a more extensive services to their target audience.

However, the impact of usability testing is inadequate to these aspects, and also covered the following various constraints or parameters that help us enhance the software's productivity.

Parameters covered by Usability Testing



1. Efficiency

2. Memorability

3. Accuracy

4. Learnability

5. Satisfaction

6. Errors

Let's see them separately in order to enhance our knowledge of usability testing:

1. Efficiency

The first constraint covered by the execution of usability testing is **Efficiency**. Here, the efficiency parameter explains the end-user who is an expert and can take the minimum amount of time to execute his/her fundamental or, we can say, undeveloped task.

2. Memorability

The second constrain that is covered by the implementation of usability testing is **Memorability**. The Memorability of an application can be beneficial or not beneficial. But, the question arises, how can we have decided the Memorability of an application is good or bad?

The below points will give the perfect answer to the above arise the question:

- When we are not asking for an application for some time and returning to the application or trying to do the simple task without any help, we can say that the **Memorability of an application is beneficial**.
- Or, if we cannot execute a simple task without any help after a duration, we can say that the **Memorability of an application is not beneficial**.

3. Accuracy

The next parameter covered by performing the Usability testing is **Accuracy**. The usability testing ensures that no inappropriate/irrelevant data or information exists in the product. And also, able to discover the broken links in the particular product that helps us develop the accuracy of the final product.

4. Learnability

Another constraint that is encompassed by usability testing is **Learnability**. In this constraint, the end-user takes a minimum amount of time to learn the fundamental task.

5. Satisfaction

The execution of usability testing ensures the **customer's satisfaction** as we know that the satisfied customer can easily or freely use the application.

6. Errors

The last and most important parameter covered by the usability testing is **Errors detection**. At this point, we try to help the end-users fix those errors they made earlier and accomplish their tasks all over again.

Various Strategies of Usability Testing/Usability Testing Methods

Like others type of software testing contains several approaches, usability testing also involved various strategies or methods. Some of the most frequently used usability testing methods are as follows:



- **A/B Testing**
- **Hallway Testing**
- **Laboratory Usability Testing**
- **Expert Review**
- **Automated Expert Review**
- **Synchronous Remote Usability Testing**
- **Asynchronous Remote Usability Testing**

Let us summarise them one by one for our better understanding:

1. A/B Testing

The first usability testing approach is **A/B Testing**, which includes creating a similar image of the product without an essential aspect from the original, which can directly affect the user performance.

A comparative analysis understands the A/B testing, and we can go through with some of the other elements such as **colour, text, or difference of the interface**.

2. Hallway Testing

The next method of usability testing is **Hallway Testing**. It is one of the most successful and cost-saving approaches compared to the other usability testing methods.

In hallway testing, some random people test the application without having any earlier knowledge of the product instead of skilled professionals. As a result, we will get more precise outcomes and reliable responses for further enhancement, if any of those random people test the application more efficiently.

The primary purpose behind hallway testing is to find the most crucial environments for the bugs because those bugs can make the simple features unproductive and lethargic to work with.

3. Laboratory Usability Testing

The third strategy of usability testing is **Laboratory Usability Testing**. The Laboratory usability testing is performed in the existence of the viewers. Generally, it is implemented by the team in an individual lab room.

In this method, the viewers are concerned about checking the performance of the test engineers regularly and reporting the results of testing to the related team.

4. Expert Review

Another general approach to usability testing is **Expert Review**. The Expert Review method includes the benefits of a professionals teams who have in-depth knowledge or experience in the specified field of performing usability tests.

Usability testing is consistent as the professional's knowledge is worth the expenditure when the product has a crucial feature. The organization needs to find out the user's response before releasing the product.

The specialist in a specified field is requested to test the product, give the response, and then submit the outcomes. To submit the outcomes, the expert review can also be performed remotely.

The expert review of usability testing is implemented rapidly and takes less time than the other type of usability testing because the professionals can easily identify the loopholes and discover the flaws in the product.

And that makes the particular process costly because the company needs to appoint a skilled person. So, sometimes the clients avoid this option.

5. Automated Expert Review

The next essential approach of **Usability Testing** is **Automated Expert Review**. As the name recommends, automated **expert review** is executed by writing automation scripts.

To execute this usability testing approach, an organization needs to appoint a resource who is well aware of writing automation scripts and developing an automation framework.

The automation test engineers write the test scripts, and when the scripts are triggered, we can easily implement the test cases. After the implementation of the test, the results are recorded and submitted.

The automated expert review is one of the successful types of usability testing because there is less human involvement, automated scripts, and fewer chances of missing any issues.

In simple words, we can specify that it is just a program-based review of all the usability constraints. However, the problem of this method is the absence of insightful reviewing when executed by persons, which makes it a slower method of testing.

It is a primarily used method of usability testing as it is not that costly compared to the **Expert Review**.

6. Remote Usability Testing

The next method of usability testing is **Remote Usability testing**. As the name indicates, remote usability testing takes place by people located at remote locations, which means those situated in various states or sometimes in some other countries to achieve their testing objectives.

The remote usability testing is executed remotely and also able to report the issues if identify any. In this approach, the response can be documented and submitted by random people, not by the skilled ones.

From time to time, remote testing is implemented using video conferencing. And this approach is less expensive in comparison with other types of usability testing approaches.

The remote usability testing can be divided into the following two parts, which are as discussed below:

- **Synchronous Remote Usability Testing**
- **Asynchronous remote Usability Testing**

Synchronous Remote Usability Testing

The first part of remote usability testing **Synchronous Remote usability testing**. After comprehensive research on the issues related to execute the usability testing through distant locations, the synchronous remote usability testing approach was put forward.

We can use the **WebEx** tools for the video conferencing of remote web sharing. However, It needs the effectiveness of a real presence to make this collective testing process a success.

Asynchronous remote Usability Testing

The second method of the Remote usability testing approach is **Asynchronous remote usability testing**.

Asynchronous remote usability testing approach help us to easily divide the user response into various demographic and performance types.

It is the most frequent method, which uses user logs, response for user interface, and testing in the user environment itself.

In maximum situations, usability testing resolves many bugs closely related to the output of performance testing processes.

Usability Testing Process

The Process of usability testing is completed into few significant steps. This process will assist us in providing and creating different results for all the issues identified during the execution of testing.

In real-time, usability testing tests the application's behavior from the user's perspective even though it is a time-consuming process, providing the tester the most precise outcomes from actual testing.

And that gives us an idea of the errors/flaws in our product and helps us distinctly before installing it on the server.

The usability testing process follows a precise set of steps to help the team get the detailed and helpful response from the end-users.

Therefore, the process of usability testing completed into the following steps, as we can see in the following image:



Step1: Planning

The first step of usability testing is **Planning**, where the team makes the test plan and generates some document samples that help the testing team complete the usability testing tasks. It is one of the most essential and crucial stages in the usability testing process.

The objective of the usability test is governed in the planning step. Here, the aim is not to have the volunteers sit in front of our application and recording their activities, but we need to fix the crucial features and elements of the system.

We need to give the tasks to our test engineers who are familiar with these crucial features. And the usability testing method, number, and demographics of usability test engineers, test report formats are also fixed throughout the planning phase.

Step2: Team Recruiting

Once the planning phase is completed, we will move to the next step of usability testing, which is **team recruiting**.

As the name suggests itself, here, we will hire or recruit the end-user delegates and the participants or the test engineers as per the budget and density of the product.

These representatives or test engineers are prepared to sit across the test sessions and validate the correctness and usability of the product.

Primarily, the selection of these test engineers is based on the necessity of testing along with the number of persons mentioned in the test plan.

As soon as the hiring of a test engineer is achieved, the team is appointed to the particular responsibilities and jobs.

Step3: Test Execution

Once the planning and team recruiting steps have been completed successfully. We will proceed to the next step, which is **Test Execution**.

In the test execution step, the test engineers execute the usability testing and implement their assigned responsibilities. In this situation, the users' needs to test the product to find irregularity, if any, and also record them correctly.

Step4: Test Result Documentation

The **test result documentation** step includes the results based on the **test execution** step and then proceeds for further analysis.

Step5: Data Analysis

Once the test result documentation is done, we will move to the next step of the usability testing process, i.e., **Data Analysis**.

The response or the feedback is obtained from usability testing evaluation in the data analysis phase. And the outcomes are classified, and the patterns are acknowledged.

In this step, the data from usability tests are wholly evaluated to get expressive implications and help us provide actionable suggestions to enhance the overall usability of our product.

Step6: Reporting

After performing all the above steps successfully, we will finally reach the last step of the usability testing process which is named as **Reporting**.

In this, we can report and share the outcomes and suggested modifications with the development team, designer, and another participant of the particular project and all the related documents along with audio, databases, screen recording, etc.

Examples of Usability Testing

Let us see some examples, where we understand the use of usability testing.

Example 1

Suppose we have two applications, namely **P and Q**, that are different but perform the same job, and we will see which one is user-friendly.

Below are some of the significant parameters or constraints we look into for testing, and most of the parameters are not measurable.

- **Look & feel**
- **Navigation should be simple**
- **Speed**
- **Compatibility**
- **Help**
- **Location of components**
- **Features**

In this example, we learn the **Application P** in **4 hours**, but we take 6 hours in order to understand the **application Q**.

Let us see other different situations here to get more clarity of the above example:

- Since we understand the **application P** in 4 hours, it becomes user-friendly if we compare it to the **application Q**.
- Suppose **look and feel** is not suitable for **application P**. In such scenarios yet, we understand **application P** in 4 hours; we cannot state that application P is user-friendly.
- Thus, we look into various parameters before we say user-friendliness of a software.

Note: What is LOOK and FEEL?

In usability testing, the term **look and feel** are most commonly used term. The look & feel is used to describe that the application should be pleasant looking.

Let say we have **blue color text in the red color background**; indeed, we don't feel like using it and make a feel to the end-user to use it.

Example 2

We are taking one **banking application** where we produce the application for the manager.

Note: Here, the Manager is the end-user.

Now, if the end-user (manager) starts using the application in front of the test engineers

Suppose two test engineer sits at the back of the end-user while he/she is using the application and takes the report of the defect as a developer to check whether the end-user is using the application in a right way or not.

And the end-user (manager) will check the application step by step because he/she knows that the Test engineer is watching him/her.

Note: Generally, the professional test engineers do not perform usability testing because they know where exactly the particular feature will fail and how it works. Therefore, a test engineer becomes user-friendly with the application. So only the end-user should do the usability testing for better results.

Sometimes Test engineer has to do usability testing for the following reasons:

- There is no money to spend on usability testing.
- Do not want to outsource to another company.

Example 3

In this example, the Director of the company goes and collects the software (suppose a gaming software) and distributes it to the various end-users like employees, friends, etc.

Now, these end-users will use particular game software and give their feedback to the Director.

This Director looks into their feedback, and see the major feedback, then consolidates all the feedback and makes one report.

If a particular feature for all the end-users has been reported, then that should be considered, or if the feature has been reported only by 1 or 2 end-user, then it becomes minor.

Once the consolidation of the major and minor bugs is done, they will be fixed based on the requirement of the director.

If it is a major bug, then it will fix first, or if it is minor, then it could be delayed or fixed in the next release.

Note: All the applications cannot be given to the end-users because it depends on the applications or the software needs.

Usability Testing Checklist

The usability testing checklist contains all the documents which are related to usability testing. We don't write test cases in usability testing as we use the standard Usability testing checklist, and we are just testing the look and feel of the application.

Note: While creating a usability checklist, we should develop a standard checklist that can be performed for all pages. And there is another case where the customer provides the checklist for the application.

To make the usability testing more successful, we will prepare the standard checklist, which means that "**what are the points to check**". Or if we do not make a checklist, we may miss some features in the application.

- **Create a checklist**
- **Review checklist**
- **Execute a checklist / Approve checklist**
- **Derive checklist report (Execution Report)**

Let see **one example** where we are creating a checklist for an application:

If we take the one **E-commerce application** and prepare the checklist, it would be like as below:

- All the images should have the alt tag (Tooltip).
- The Login feature should have the Forget password link.

- All the pages should have a link to the homepage of the application.
- Should be able to access all the components.

Like this, we can drive as many checklists as possible based on the product or the application.

Bugs in Usability Testing

A usual error in usability testing is organizing a study too late in the design process. If we wait until our product is released, we won't have the time or money to resolve any issues. And we have wasted a lot of effort creating our product the wrong way.

Furthermore, we may encounter some more bugs while testing any software or application. And those bugs could be the **Path holes** and **Potential bugs**.

Path holes and Potential bugs: Path holes and potential bugs are those, which are visible to the developers and the test engineers while performing usability testing.

The Advantages of Usability Testing

Some of the significant benefits of using the usability testing are as discussed below:

- The execution of usability testing helps us to validate the usability of the software.
- Its enhanced user satisfaction with the software product and also ensured to deliver a good quality product.
- The implementation of usability testing will improve the adequacy and consistency of the software product.
- With the help of usability testing, we can discover the usability issues before delivering the final product.
- The end-user is always eager to use the application.
- The execution of usability testing helps us to identify the possible bugs and defects in the software.
- It helps us to make the software more efficient and applicable.
- The usage of usability testing will help us in receiving related and precise user responses.
- It improves the adequacy and consistency of the software product.

The Disadvantage of Usability Testing

Some of most common drawbacks of implementing usability testing are as discussed below:

- As we know that budgeting is the most crucial factor while performing any software testing. Here, usability testing costing also plays an essential role. It requires many resources to establish a usability test lab, and sometimes hiring or recruiting a usability test engineer could be costly.
- As we understood from the above discussion of the usability testing that it is implemented by the end-users, sometimes it is a bit harder to identify the volunteers who can work as test engineers.
- Primarily, usability testing is not 100% representative of the actual condition.

Conclusion

After seeing all the vital **usability testing topics**, we can conclude that it is extensive testing process, which requires a higher level of understanding of this field as well as a creative mind.

Implementing usability testing is necessary for organizations worldwide as it is one of the most effective approaches of **software testing** that helps the test engineers and developers in order to sustain the **usability, correctness, consistency, performance, and other essential characteristics of the software**.

Hence, if the usability testing is performed throughout the initial stage of software development, we can guarantee the ease of use of our applications and deliver a significant product which fulfils the user expectation.

Compatibility testing

In this section, we will learn about what is compatibility testing, why we use it, when we should perform it, types of compatibility testing, compatibility testing process, and compatibility bug and tools.

What is compatibility testing?

It is part of non-functional testing.

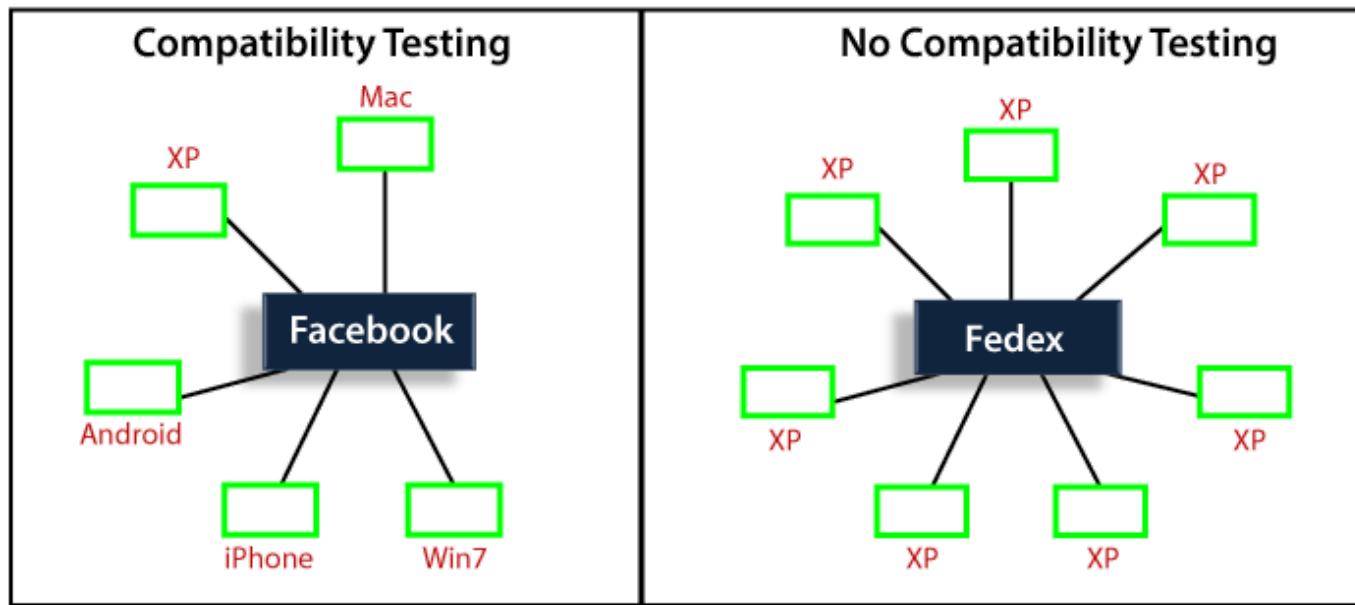
“ Checking the functionality of an application on different software, hardware platforms, network, and browsers is known as compatibility testing. ”

Why we use compatibility testing?

Once the application is stable, we move it to the production, it may be used or accessed by multiple users on the different platforms, and they may face some compatibility issues, to avoid these issues, we do one round of compatibility testing.

When should we perform Compatibility testing?

Generally, we go for compatibility testing, only when the application or software is functionally stable.



Note: It is not done for each application; we will do it only for that application where we don't have control over the platform used by users.

Types of Compatibility testing

Following are the types of compatibility testing:

- **Software**
- **Hardware**
- **Network**
- **Mobile**

Software

Here, software means different operating systems (Linux, Window, and Mac) and also check the software compatibility on the various versions of the operating systems like Win98, Window 7, Window 10, Vista, Window XP, Window 8, UNIX, Ubuntu, and Mac.

And, we have two types of version compatibility testing, which are as follows:

- **Forward Compatibility Testing:** Test the software or application on the new or latest versions. **For example:** Latest Version of the platforms (software) **Win 7 → Win 8 → Win 8.1 → Win 10**
- **Backward Compatibility Testing:** Test the software or application on the old or previous versions. **For example:** Window XP → Vista → Win 7 → Win 8 → Win 8.1

And different browsers like **Google Chrome**, **Firefox**, and **Internet Explorer**, etc.

Hardware

The application is compatible with different sizes such as RAM, hard disk, processor, and the graphic card, etc.

Mobile

Check that the application is compatible with mobile platforms such as iOS, Android, etc.

Network

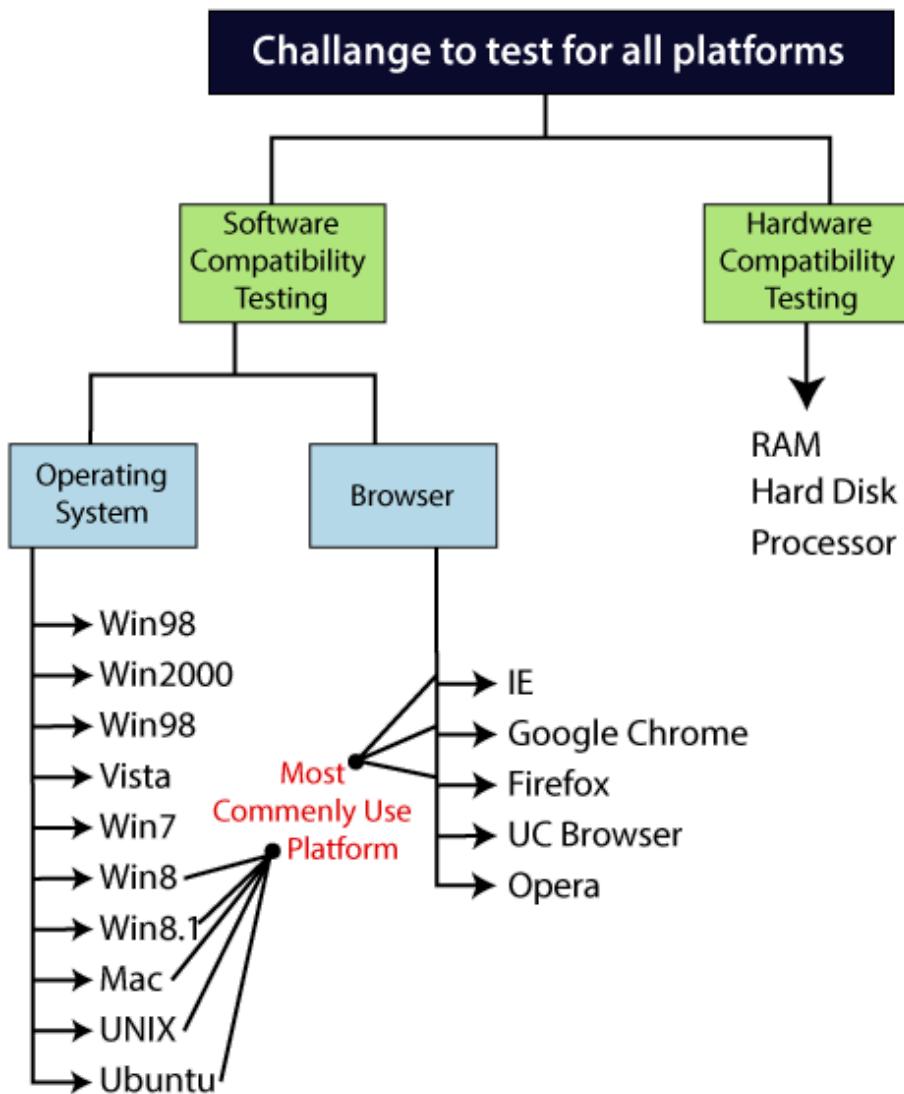
Checking the compatibility of the software in the different network parameters such as operating speed, bandwidth, and capacity.

What is the most challenging part to test compatibility testing?

The most challenging thing while performing compatibility testing is to decide the necessary needs which are to be tested.

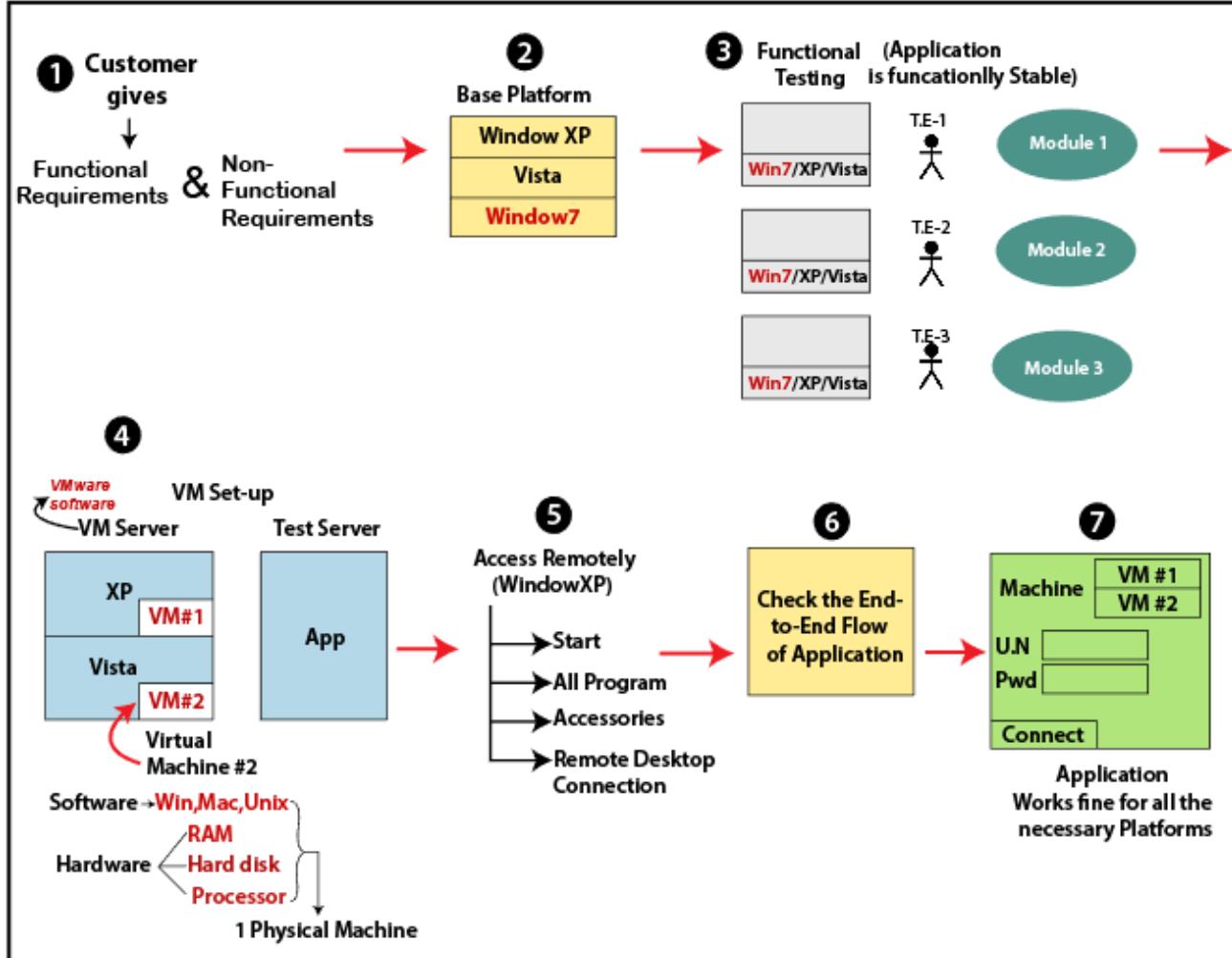
We can't test on all existing platforms since it will be a time-consuming process.

Hence, we only select those platforms which are commonly used by the end-users.



Compatibility testing process

Here, we are performing an O.S (Operating system) compatibility testing process.



- Firstly, the customer will give the functional requirement as well as a non-functional requirement.
- After getting the non-functional requirement, the one base platform will decide according to the most commonly used platform.
- After that, the test engineer will start functional testing on the base platform until the application is functional stable.
- We have to test the application on a different platform, so for this, we have a VMware software.

Note1: VMware software: with the help of VMware, we can divide one physical machine into multiple Virtual machines, which can be accessed simultaneously.

- For compatibility testing, we will use VM Server where we install all necessary operating system and browser, and access that server using Remote Desktop connection.
- We access VMware set up remotely, and after that, the test engineer will do one round of compatibility testing on the platform and check the end-to-end flow.
- The end-to-end flow will go on until the application is stable, and applications work fine for all necessary platforms and handover to the customer.

Note2:

- For Browser compatibility testing, we don't have to go for the VMware setup, since multiple browsers can be installed in a single machine as well as they also are accessed simultaneously.
- In the case of various versions of the same browser, we have to go with VMware set up since multiple versions of the browser cannot be accessed simultaneously from a single system.
- We can do Software compatibility testing on different platforms.

Compatibility testing bug/issue

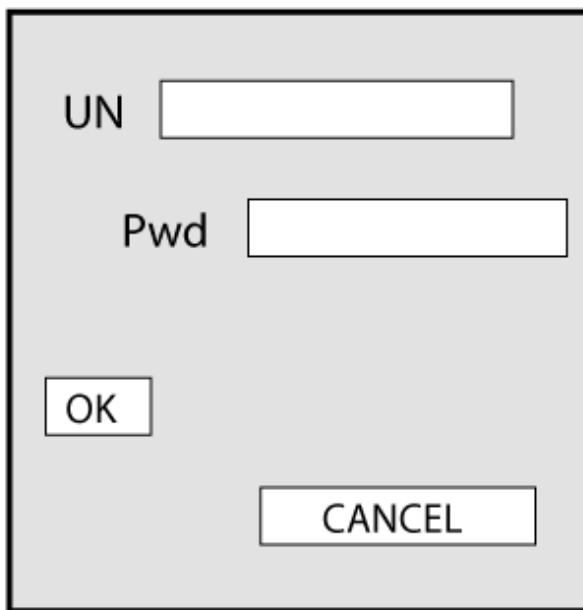
These bugs are those that are happening in one platform, but not occur in another platform.

Generally, the compatibility bugs are user interface issues, some of the U.I problems are as follows:

- Alignment issue
- Overlap issue
- Scattered issue
- Look and feel issue

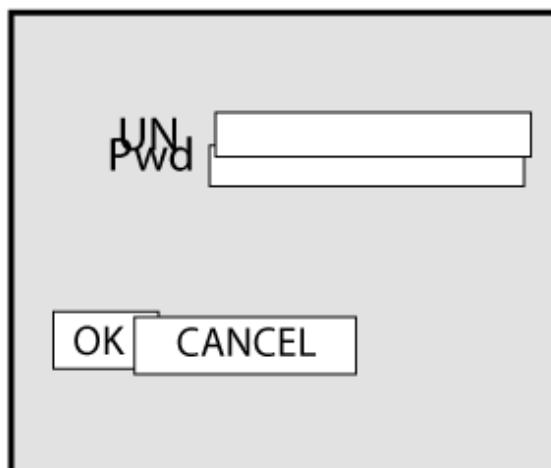
Alignment issue

The alignment issue is that in which the element of the page is not aligned in a proper format as we can see in the below image:



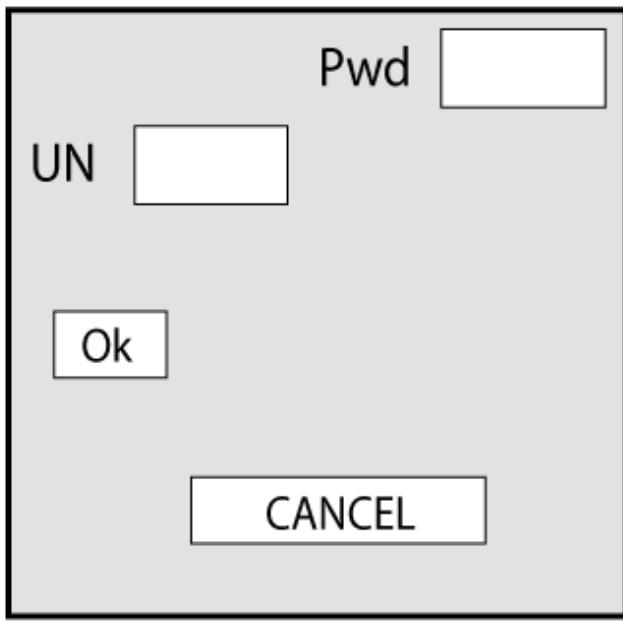
Overlap issue

When one attribute is overlapping to another attribute, it may happen when we are trying to open the application on the different platforms, browsers as we can see in the below image:



Scattered issue

When the test engineer performing compatibility testing on the application, and that application is not compatible with all browsers, and platform that's why the scattered issues may occur as we can see in the below image:



Testing the functionality, integration, and end-to-end flow on the different platforms are what we do in the compatibility testing.

Compatibility issue: When the application feature is not working in one operating system, but working fine in other operating systems. It happens because a program written by the developer is not working in only one platform, but working fine on all other platforms.

Functionality issue: when a feature is not working in all operating system/ platform. And it is also known as functionality defect issue.

For compatibility testing, the test execution report looks something like this:

Text Execution Report for Compatibility Testing

Step No.	Test Case Name	Windows Vista		Windows XP		Windows 7	
		Status	Comments	Status	Comments	Status	Comments
1	Fail	Pass	Fail
2	Fail	Fail	Pass
3	Pass	Pass	Pass
....
....
....
7	Fail	Pass
....	Pass	Pass
237	Pass	Pass

Various Browsers

WINDOWS XP					
Mozilla FireFox		Opera		Internet Explorer	
Status	Comments	Status	Comments	Status	Comments

Compatibility testing tools

Some of the most commonly used compatibility testing tools are as follows:

- **LambdaTest**
- **BrowserStack**
- **BrowseEMail**
- **TestingBot**

LambdaTest

It is an open-source browser compatibility testing tool in the cloud. With the help of this tool, we can test our web application on almost any mobile browsers and desktop browsers. LambdaTest has a screenshot feature, which allows us to take the full-page screenshots of our web pages.

In this tool, we can test our application on the real browsers, and the user has a large number of mobile and desktop browsers option to check the compatibility of the application.

BrowserStack

This tool helps us to test the websites and mobile applications compatibility over multiple browsers and platforms.

In this, we can test a web application in various browsers and mobile applications such as android and iOS in all the mobile devices.

The main product of BrowserStack tools are Live, Automate, App Live, and App Automate, with the help of these tools, we can maintain the cost.

These tools help us to reduce the time, price, and maintenance overhead associated with testing.

BrowseEMAIL

This tool can run the application on different operating systems such as Linux, Windows, and macOS and, it is a cross-browser testing tool.

It is used to test the application on all the mobile browsers and desktop, and we can directly use it on our local machine and in our local network.

And we can perform the regression and visual testing without any network delays, and we can also record and play the automated tests against a lot of desktop and mobile browsers.

TestingBot

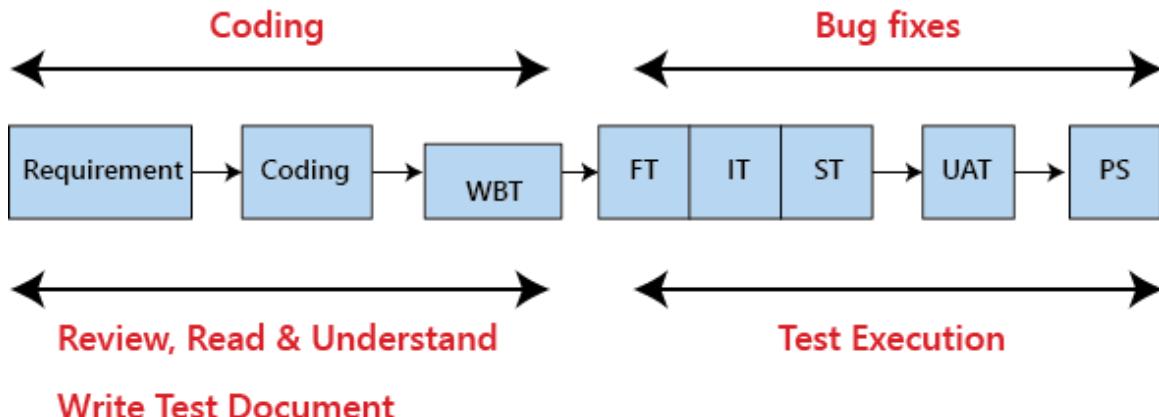
This tool is used to perform the test on various browsers such as Firefox, Chrome, Edge, IE, Safari, and so on. In this, we can compare the screenshots from the multiple browsers and platforms and test the responsive layout of the application.

It will provide a large number of browser versions for the instant use of automation testing.

Testing Documentation

Testing documentation is the documentation of artifacts that are created during or before the testing of a software application. Documentation reflects the importance of processes for the customer, individual and organization.

Projects which contain all documents have a high level of maturity. Careful documentation can save the time, efforts and wealth of the organization.



There is the necessary reference document, which is prepared by every test engineer before starting the test execution process. Generally, we write the test document whenever the developers are busy in writing the code.

Once the test document is ready, the entire test execution process depends on the test document. The primary objective for writing a test document is to decrease or eliminate the doubts related to the testing activities.

Types of test document

In software testing, we have various types of test document, which are as follows:

- Test scenarios
- Test case
- Test plan
- Requirement traceability matrix(RTM)
- Test strategy
- Test data
- Bug report
- Test execution report



Test Scenarios

It is a document that defines the multiple ways or combinations of testing the application. Generally, it is prepared to understand the flow of an application. It does not consist of any inputs and navigation steps.

For more information about test scenario, refers to the below link:

<https://www.javatpoint.com/test-scenario>

Test case

It is an in-details document that describes step by step procedure to test an application. It consists of the complete navigation steps and inputs and all the scenarios that need to be tested for the application. We will write the test case to maintain the consistency, or every tester will follow the same approach for organizing the test document.

For more information about test case, refers to the below link:

<https://www.javatpoint.com/test-case>

Test plan

It is a document that is prepared by the managers or test lead. It consists of all information about the testing activities. The test plan consists of multiple components such as **Objectives**, **Scope**, **Approach**, **Test Environments**, **Test methodology**, **Template**, **Role & Responsibility**, **Effort estimation**, **Entry and Exit criteria**, **Schedule**, **Tools**, **Defect tracking**, **Test Deliverable**, **Assumption**, **Risk**, and **Mitigation Plan or Contingency Plan**.

For more information about the test plan, refers to the below link:

<https://www.javatpoint.com/test-plan>

Requirement Traceability Matrix (RTM)

The Requirement traceability matrix [RTM] is a document which ensures that all the test case has been covered. This document is created before the test execution process to verify that we did not miss writing any test case for the particular requirement.

For more information about RTM, refers to the below link:

<https://www.javatpoint.com/traceability-matrix>

Test strategy

The test strategy is a high-level document, which is used to verify the test types (levels) to be executed for the product and also describe that what kind of technique has to be used and which module is going to be tested. The Project Manager can approve it. It includes the multiple components such as documentation formats, objective, test processes, scope, and customer communication strategy, etc. we cannot modify the test strategy.

Test data

It is data that occurs before the test is executed. It is mainly used when we are implementing the test case. Mostly, we will have the test data in the Excel sheet format and entered manually while performing the test case.

The test data can be used to check the expected result, which means that when the test data is entered, the expected outcome will meet the actual result and also check the application performance by entering the incorrect input data.

Bug report

The bug report is a document where we maintain a summary of all the bugs which occurred during the testing process. This is a crucial document for both the developers and test engineers because, with the help of bug reports, they can easily track the defects, report the bug, change the status of bugs which are fixed successfully, and also avoid their repetition in further process.

Test execution report

It is the document prepared by test leads after the entire testing execution process is completed. The test summary report defines the constancy of the product, and it contains information like the modules, the number of written test cases, executed, pass, fail, and their percentage. And each module has a separate spreadsheet of their respective module.

Why documentation is needed

If the testing or development team gets software that is not working correctly and developed by someone else, so to find the error, the team will first need a document. Now, if the documents are available then the team will quickly find out the cause of the error by examining documentation. But, if the documents are not available then the tester need to do black box and white box testing again, which will waste the time and money of the organization. More than that, Lack of documentation becomes a problem for acceptance.

Example

Let's take a real-time example of Microsoft, Microsoft launch every product with proper user guidelines and documents, which are very explanatory, logically consistent and easy to understand for any user. These are all the reasons behind their successful products.

Benefits of using Documentation

- Documentation clarifies the quality of methods and objectives.
- It ensures internal coordination when a customer uses software application.
- It ensures clarity about the stability of tasks and performance.
- It provides feedback on preventive tasks.
- It provides feedback for your planning cycle.

- It creates objective evidence for the performance of the quality management system.
- If we write the test document, we can't forget the values which we put in the first phase.
- It is also a time-saving process because we can easily refer to the text document.
- It is also consistent because we will test on the same value.

The drawback of the test document

- It is a bit tedious because we have to maintain the modification provided by the customer and parallel change in the document.
- If the test documentation is not proper, it will replicate the quality of the application.
- Sometimes it is written by that person who does not have the product knowledge.
- Sometimes the cost of the document will be exceeding its value.

Test Scenario

The test scenario is a detailed document of test cases that cover end to end functionality of a software application in liner statements. The liner statement is considered as a scenario. The test scenario is a high-level classification of testable requirements. These requirements are grouped on the basis of the functionality of a module and obtained from the use cases.

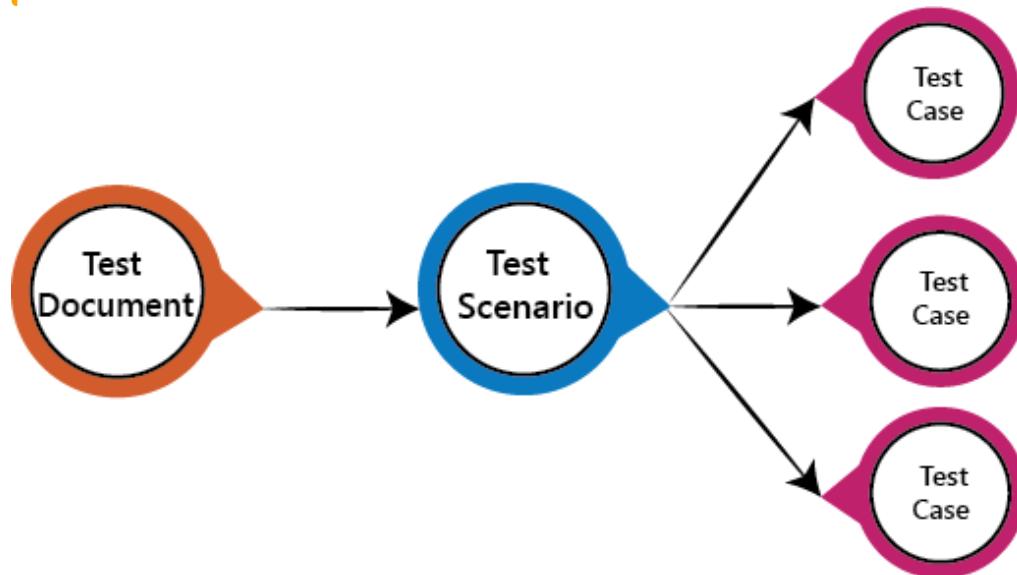
In the test scenario, there is a detailed testing process due to many associated test cases. Before performing the test scenario, the tester has to consider the test cases for each scenario.

In the test scenario, testers need to put themselves in the place of the user because they test the software application under the user's point of view. Preparation of scenarios is the most critical part, and it is necessary to seek advice or help from customers, stakeholders or developers to prepare the scenario.

Note:

The test scenarios can never be used for the text execution process because it does not consist of navigation steps and input.

These are the high-level documents that talks about all the possible combination or multiple ways or combinations of using the application and the primary purpose of the test scenarios are to understand the overall flow of the application.



How to write Test Scenarios

As a tester, follow the following steps to create Test Scenarios-

- Read the requirement document such as BRS (Business Requirement Specification), SRS (System Requirement Specification) and FRS (Functional Requirement Specification) of the software which is under the test.
- Determine all technical aspects and objectives for each requirement.
- Find all the possible ways by which the user can operate the software.
- Ascertain all the possible scenario due to which system can be misused and also detect the users who can be hackers.
- After reading the requirement document and completion of the scheduled analysis make a list of various test scenarios to verify each function of the software.
- Once you listed all the possible test scenarios, create a traceability matrix to find out whether each and every requirement has a corresponding test scenario or not.

- Supervisor of the project reviews all scenarios. Later, they are evaluated by other stakeholders of the project.

Features of Test Scenario

- The test scenario is a liner statement that guides testers for the testing sequence.
- Test scenario reduces the complexity and repetition of the product.
- Test scenario means talking and thinking about tests in detail but write them in liner statements.
- It is a thread of operations.
- Test scenario becomes more important when the tester does not have enough time to write test cases, and team members agree with a detailed liner scenario.
- The test scenario is a time saver activity.
- It provides easy maintenance because the addition and modification of test scenarios are easy and independent.

Note:

Some rules have to be followed when we were writing test scenarios:

- Always list down the most commonly used feature and module by the users.
- We always start the scenarios by picking module by module so that a proper sequence is followed as well as we don't miss out on any module level.
- Generally, scenarios are module level.
- Delete scenarios should always be the last option else, and we will waste lots of time in creating the data once again.
- It should be written in a simple language.
- Every scenario should be written in one line or a maximum of two lines and not in the paragraphs.
- Every scenario should consist of Dos and checks.

Example of Test scenarios

Here we are taking the **Gmail application** and writing test scenarios for different modules which are most commonly used such as **Login, Compose, Inbox, and Trash**

Test scenarios on the Login module

- Enter the valid login details (Username, password), and check that the home page is displayed.
- Enter the invalid Username and password and check for the home page.
- Leave Username and password blank, and check for the error message displayed.
- Enter the valid Login, and click on the cancel, and check for the fields reset.
- Enter invalid Login, more than three times, and check that account blocked.
- Enter valid Login, and check that the **Username** is displayed on the home screen.

Test scenarios on Compose module

- Checks that all users can enter email ides in the **To, Cc, and Bcc**.

- Check that the entire user can enter various email ids in To, Cc, and Bcc.
- Compose a mail, send it, and check for the confirmation message.
- Compose a mail, send it, and check in the sent item of the sender and the inbox.
- Compose a mail, send it, and check for invalid and valid email id (valid format), check the mail in sender inbox.
- Compose main, discard, and then check for conformation message and check-in draft.
- Compose mail click on save as draft and check for the confirmation message
- Compose mail click on close and check for conformation save as drafts.

Test scenarios on Inbox module

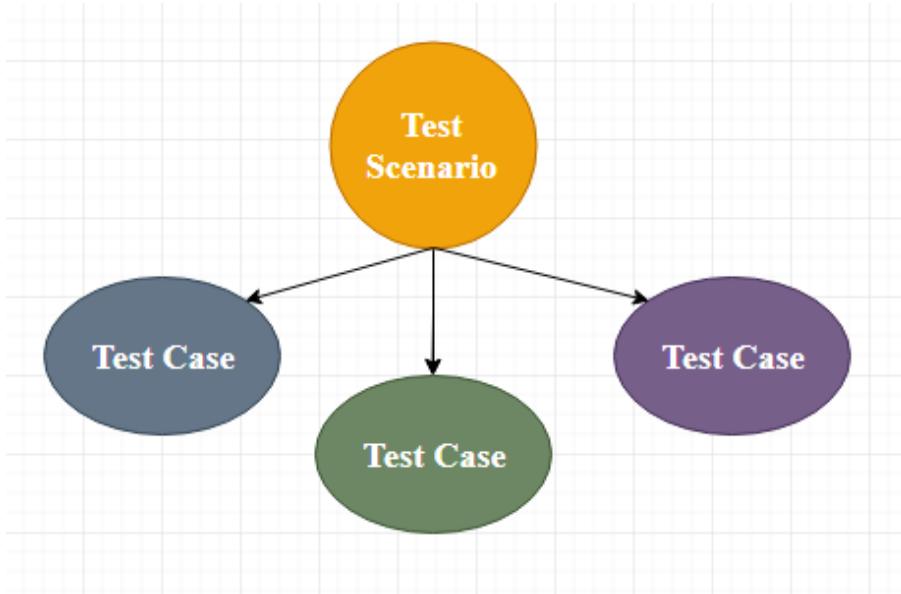
- Click on the inbox, and verify all received mail are displayed and highlighted in the inbox.
- Check that a latest received mail has been displayed to the sender email id correctly.
- Select the mail, reply and forward send it; check in the sent item of sender and inbox of the receiver.
- Check for any attached attachments to the mail that are downloaded or not.
- Check that attachment is scanned correctly for any viruses before download.
- Select the mail, reply and forward save as draft, and check for the confirmation message and checks in the Draft section.
- Check all the emails are marked as read are not highlighted.
- Check all mail recipients in **Cc** are visible to all users.
- Checks all email recipients in **Bcc** are not visible to the users.
- Select mail, delete it, and then check in the **Trash** section.

Test scenario on Trash module

- Open trash, check all deleted mail present.
- Restore mail from Trash; check-in the corresponding module.
- Select mail from trash, delete it, and check mail is permanently deleted.

Test Case

The test case is defined as a group of conditions under which a tester determines whether a software application is working as per the customer's requirements or not. Test case designing includes preconditions, case name, input conditions, and expected result. A test case is a first level action and derived from test scenarios.



It is an in-detailed document that contains all possible inputs (positive as well as negative) and the navigation steps, which are used for the test execution process. Writing of test cases is a one-time attempt that can be used in the future at the time of regression testing.

Test case gives detailed information about testing strategy, testing process, preconditions, and expected output. These are executed during the testing process to check whether the software application is performing the task for that it was developed or not.

Test case helps the tester in defect reporting by linking defect with test case ID. Detailed test case documentation works as a full proof guard for the testing team because if developer missed something, then it can be caught during execution of these full-proof test cases.

To write the test case, we must have the requirements to derive the inputs, and the test scenarios must be written so that we do not miss out on any features for testing. Then we should have the test case template to maintain the uniformity, or every test engineer follows the same approach to prepare the test document.

Generally, we will write the test case whenever the developer is busy in writing the code.

When do we write a test case?

We will write the test case when we get the following:

- When the customer gives the business needs then, the developer starts developing and says that they need 3.5 months to build this product.
- And In the meantime, the testing team will **start writing the test cases**.
- Once it is done, it will send it to the Test Lead for the review process.
- And when the developers finish developing the product, it is handed over to the testing team.
- The test engineers never look at the requirement while testing the product document because testing is constant and does not depend on the mood of the person rather than the quality of the test engineer.

Note: When writing the test case, the actual result should never be written as the product is still being in the development process. That's why the actual result should be written only after the execution of the test cases.

Why we write the test cases?

We will write the test for the following reasons:

- **To require consistency in the test case execution**
- **To make sure a better test coverage**
- **It depends on the process rather than on a person**
- **To avoid training for every new test engineer on the product**

To require consistency in the test case execution: we will see the test case and start testing the application.

To make sure a better test coverage: for this, we should cover all possible scenarios and document it, so that we need not remember all the scenarios again and again.

It depends on the process rather than on a person: A test engineer has tested an application during the first release, second release, and left the company at the time of third release. As the test engineer understood a module and tested the application thoroughly by deriving many values. If the person is not there for the third release, it becomes difficult for the new person. Hence all the derived values are documented so that it can be used in the future.

To avoid giving training for every new test engineer on the product: When the test engineer leaves, he/she leaves with a lot of knowledge and scenarios. Those scenarios should be documented so that the new test engineer can test with the given scenarios and also can write the new scenarios.

Note: when the developers are developing the first product for the First release, the test engineer writes the test cases. And in the second release, when the new features are added, the test engineer writes the test cases for that also, and in the next release, when the elements are modified, the test engineer will change the test cases or writes the new test cases as well.

Test case template

The primary purpose of writing a test case is to achieve the efficiency of the application.

Header

Test Case Name/ID :- **Release - Version - Application Name - Module**

Test Case Type:- **F.T.C I.T.C S.T.C**

Requirement Number:-

Module:-

Severity:- **Critical/Major/Minor**

Status:-

Release:-

Version:-

Pre-condition:-

Test Data:-

Summary:-

Body

Step No.	Description	Inputs	Expected Result	Actual Result	Status	Comments
...
...

Footer

{ Author:-

Reviewd By:-

Date:-

Approved By:-

As we know, the **actual result** is written after the test case execution, and most of the time, it would be same as the **expected result**. But if the test step will fail, it will be different. So, the actual result field can be skipped, and in the **Comments** section, we can write about the bugs.

And also, the **Input field** can be removed, and this information can be added to the **Description field**.

The above template we discuss above is not the standard one because it can be different for each company and also with each application, which is based on the test engineer and the test lead. But, for testing one application, all the test engineers should follow a usual template, which is formulated.

The test case should be written in simple language so that a new test engineer can also understand and execute the same.

In the above sample template, the header contains the following:

Step number

It is also essential because if step number 20 is failing, we can document the bug report and hence prioritize working and also decide if it's a critical bug.

Test case type

It can be functional, integration or system test cases or positive or negative or positive and negative test cases.

Release

One release can contain many versions of the release.

Pre-condition

These are the necessary conditions that need to be satisfied by every test engineer before starting the test execution process. Or it is the data configuration or the data setup that needs to be created for the testing.

For example: In an application, we are writing test cases to add users, edit users, and delete users. The per-condition will be seen if user A is added before editing it and removing it.

Test data

These are the values or the input we need to create as per the per-condition.

For example, Username, Password, and account number of the users.

The test lead may be given the test data like username or password to test the application, or the test engineer may themselves generate the username and password.

Severity

The severity can be **major, minor, and critical**, the severity in the test case talks about the importance of that particular test cases. All the test execution process always depends on the severity of the test cases.

We can choose the severity based on the module. There are many features include in a module, even if one element is critical, we claim that test case to be critical. It depends on the functions for which we are writing the test case.

For example, we will take the Gmail application and let us see the severity based on the modules:

Modules	Severity
Login	Critical
Help	Minor
Compose mail	Critical

Setting	Minor
Inbox	Critical
Sent items	Major
Logout	Critical

And for the banking application, the severity could be as follows:

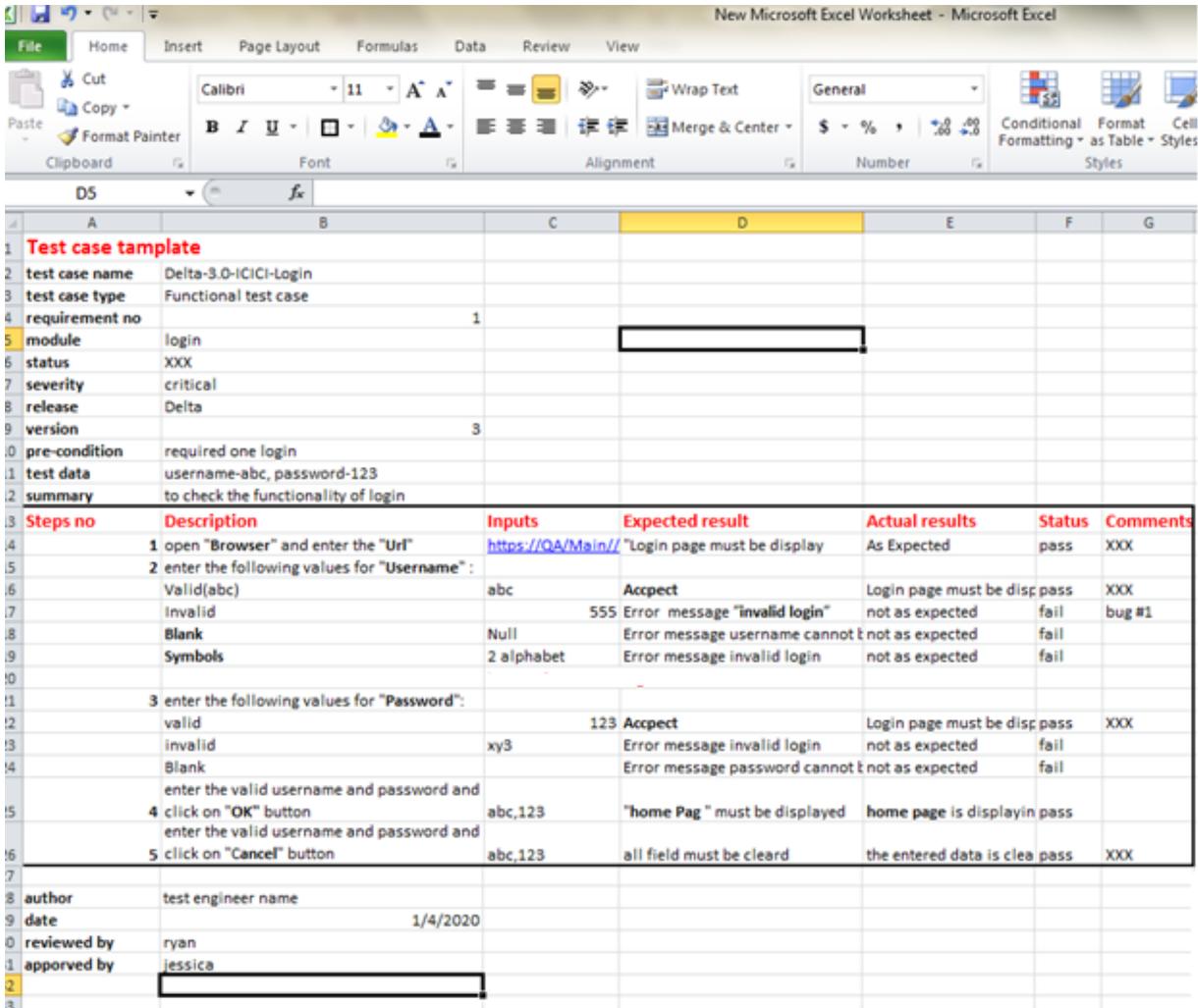
Modules	Severity
Amount transfer	critical
Feedback	minor

Brief description

The test engineer has written a test case for a particular feature. If he/she comes and reads the test cases for the moment, he/she will not know for what feature has written it. So, the brief description will help them in which feature test case is written.

Example of a test case template

Here, we are writing a test case for the **ICICI application's Login** module:



The screenshot shows an Excel spreadsheet titled "New Microsoft Excel Worksheet - Microsoft Excel". The spreadsheet contains a test case template for the ICICI application's Login module. The data is organized into several sections:

- Test case template:** A section at the top containing general information about the test case.
- Test Case Details:** Rows 2 through 10 provide specific details:
 - test case name: Delta-3.0-ICICI-Login
 - test case type: Functional test case
 - requirement no: 1
 - module: login
 - status: XXX
 - severity: critical
 - release: Delta
 - version: 3
 - pre-condition: required one login
 - test data: username-abc, password-123
 - summary: to check the functionality of login
- Steps no:** A table starting from row 11, mapping steps to descriptions, inputs, expected results, actual results, status, and comments.
- Author, Date, Reviewed by, Approved by:** Information at the bottom of the sheet.

Types of test cases

We have a different kind of test cases, which are as follows:

- **Function test cases**
- **Integration test cases**
- **System test cases**

The functional test cases

Firstly, we check for which field we will write test cases and then describe accordingly.

In functional testing or if the application is data-driven, we require the input column else; it is a bit time-consuming.

Rules to write functional test cases:

- In the expected results column, try to use **should be** or **must be**.
- Highlight the Object names.
- We have to describe only those steps which we required the most; otherwise, we do not need to define all the steps.
- To reduce the excess execution time, we will write steps correctly.
- Write a generic test case; do not try to hard code it.

Let say it is the amount transfer module, so we are writing the functional test cases for it and then also specifies that it is not a login feature.

The form is titled "AMOUNT TRANSFER". It contains three input fields:

- "From Account Number" (FAN) with placeholder text "...."
- "To Account Number" (TAN) with placeholder text "...."
- "Amount" with placeholder text "...."

At the bottom, there are two buttons: "TRANSFER" on the left and "CANCEL" on the right.

The functional test case for amount transfer module is in the below Excel file:

Functional Test case template						
Test case name	beta-1.0-ICIO-amount transfer					
Test case type	Functional test case					
Requirement no	6					
Module	amount transfer					
Status	XXX					
Severity	Critical					
Release	Beta					
Version	1					
Pre-condition	sender login two account number Balance-> exist					
Test data	Username:xyz, Password:1234 1231, 4321 3000-9000					
Summary	to check the functionality of amount transfer					
Steps no	Description	Inputs	Expected result	Actual results	Status	Comments
1	Open "Browser" and enter the "Url"	https://QA/Main/	"Login page" must be display	As Expected	pass	XXX
2	Enter the following values for "Username" and "Password" and click on the "OK" button	xyz, 1234	"Home page" must be displayed	As Expected	pass	XXX
3	Click on the "Amount Transfer"	Null			pass	XXX
4	Enter the following for From Account number(FAN):					
	valid	1234	Accept	As Expected	pass	
	invalid	1124	Error message invalid account		fail	
	blank	—	Error message FAN value cannot be blank		fail	
	—	—				
	—	—	test maximum coverage			
5	Enter the following values for "TO account number"(TAN)					
	valid	4321	Accept	As expected	pass	XXX
	invalid	6655	Error message invalid account		fail	
	Blank	—	Error message TAN value cannot be blank			
	—	—				
	—	—	test maximum coverage			
6	enter the value for "Amount"					
	valid	5000, 5001,9000,84	Accept	As expected	pass	XXX
	invalid	4999,9001	Error message amount should be between (5000-9000)		fail	
7	Enter the value for "FAN, TAN, Amount" click on the "Transfer" button					
	FAN	1234				
	TAN	4321	"Confirmation Message" amount transfer sucessfully must be displayed	As expected	pass	XXX
	Amount	6000				
	Enter the value for "FAN, TAN, Amount" click on the "Cancel" button					
	FAN	1234				
	TAN	4321	All field must be cleared	As expected	pass	XXX
	Amount	6000				
Author	Sem					
Date	4/1/2020					
Reviewed by	jessica					
Approved by	ryan					

Integration test case

In this, we should not write something which we already covered in the functional test cases, and something we have written in the integration test case should not be written in the system test case again.

Rules to write integration test cases

- Firstly, understand the product
- Identify the possible scenarios
- Write the test case based on the priority

When the test engineer writing the test cases, they may need to consider the following aspects:

If the test cases are in details:

- They will try to achieve maximum test coverage.
- All test case values or scenarios are correctly described.
- They will try to think about the execution point of view.
- The template which is used to write the test case must be unique.

Note: when we involve fewer numbers of steps or coverage is more, it should be the best test case, and when these test cases are given to anyone, they will understand easily.

System test cases

We will write the system test cases for the end-to-end business flows. And we have the entire modules ready to write the system test cases.

The process to write test cases

The method of writing a test case can be completed into the following steps, which are as below:



System study

In this, we will understand the application by looking at the requirements or the SRS, which is given by the customer.

Identify all scenarios:

- When the product is launched, what are the possible ways the end-user may use the software to identify all the possible ways.
- I have documented all possible scenarios in a document, which is called test design/high-level design.
- The test design is a record having all the possible scenarios.

Write test cases

Convert all the identified scenarios to test claims and group the scenarios related to their features, prioritize the module, and write test cases by applying test case design techniques and use the standard test case template, which means that the one which is decided for the project.

Review the test cases

Review the test case by giving it to the head of the team and, after that, fix the review feedback given by the reviewer.

Test case approval

After fixing the test case based on the feedback, send it again for the approval.

Store in the test case repository

After the approval of the particular test case, store in the familiar place that is known as the test case repository.

To get complete information about test case review process refers to the below link:

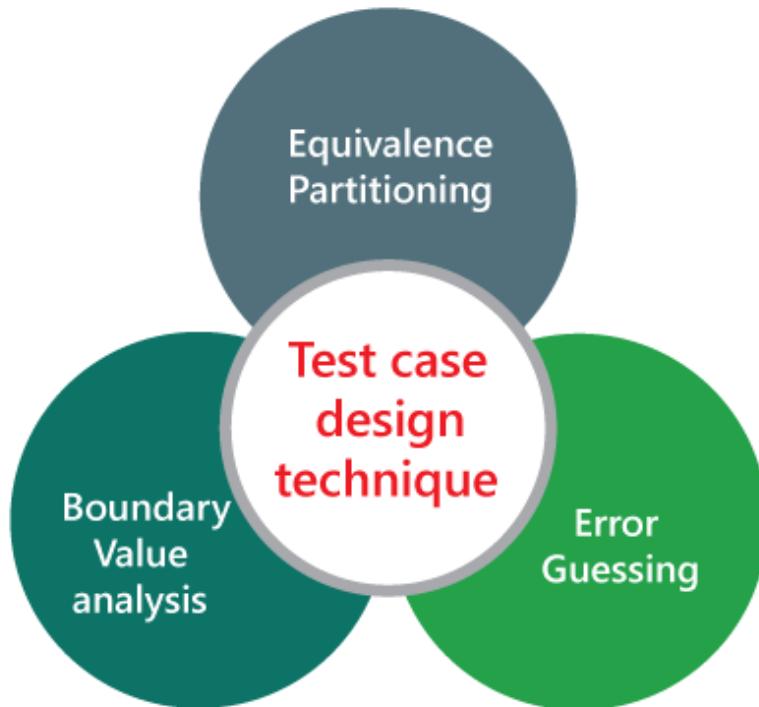
<https://www.javatpoint.com/test-case-review-process>

Error Guessing Technique

The test case design technique or methods or approaches that need to be followed by every test engineer while writing the test cases to achieve the maximum test coverage. If we follow the test case design technique, then it became process-oriented rather than person-oriented.

The test case design technique ensures that all the possible values that are both positive and negative are required for the testing purposes. In software testing, we have three different test case design techniques which are as follows:

- Error Guessing
- Equivalence Partitioning
- Boundary Value Analysis[BVA]



In this section, we will understand the first test case design technique that is **Error guessing techniques**.

Error guessing is a technique in which there is no specific method for identifying the error. It is based on the experience of the test analyst, where the tester uses the experience to guess the problematic areas of the software. It is a type of black box testing technique which does not have any defined structure to find the error.

In this approach, every test engineer will derive the values or inputs based on their understanding or assumption of the requirements, and we do not follow any kind of rules to perform error guessing technique.

The accomplishment of the error guessing technique is dependent on the ability and product knowledge of the tester because a good test engineer knows where the bugs are most likely to be, which helps to save lots of time.

How does the error guessing technique be implemented?

The implementation of this technique depends on the experience of the tester or analyst having prior experience with similar applications. It requires only well-experienced testers with quick error guessing technique. This technique is used to find errors that may not be easily captured by formal black box testing techniques, and that is the reason, it is done after all formal techniques.

The scope of the error guessing technique entirely depends on the tester and type of experience in the previous testing involvements because it does not follow any method and guidelines. Test cases are prepared by the analyst to identify conditions. The conditions are prepared by identifying most error probable areas and then test cases are designed for them.

The main purpose of this technique is to identify common errors at any level of testing by exercising the following tasks:

- Enter blank space into the text fields.
- Null pointer exception.
- Enter invalid parameters.
- Divide by zero.
- Use maximum limit of files to be uploaded.
- Check buttons without entering values.

The increment of test cases depends upon the ability and experience of the tester.

Purpose of Error guessing

The main purpose of the error guessing technique is to deal with all possible errors which cannot be identified as informal testing.

- The main purpose of error guessing technique is to deal with all possible errors which cannot be identified as informal testing.
- It must contain the all-inclusive sets of test cases without skipping any problematic areas and without involving redundant test cases.
- This technique accomplishes the characteristics left incomplete during the formal testing.

Depending on the tester's intuition and experience, all the defects cannot be corrected. There are some factors that can be used by the examiner while using their experience -

- Tester's intuition
- Historical learning
- Review checklist
- Risk reports of the software
- Application UI
- General testing rules
- Previous test results
- Defects occurred in the past
- Variety of data which is used for testing
- Knowledge of AUT

Examples of Error guessing method

Example1

A function of the application requires a mobile number which must be of 10 characters. Now, below are the techniques that can be applied to guess error in the mobile number field:

- What will be the result, if the entered character is other than a number?
- What will be the result, if entered characters are less than 10 digits?
- What will be the result, if the mobile field is left blank?

After implementing these techniques, if the output is similar to the expected result, the function is considered to be bug-free, but if the output is not similar to the expected result, so it is sent to the development team to fix the defects.

However, error guessing is the key technique among all testing techniques as it depends on the experience of a tester, but it does not give surety of highest quality benchmark. It does not provide full coverage to the software. This technique can yield a better result if combined with other techniques of testing.

Example2

Suppose we have one bank account, and we have to deposit some money over there, but the amount will be accepted on a particular range of **which is 5000-7000**. So here, we will provide the different input's value until it covers the maximum test coverage based on the error guessing technique, and see whether it is accepted or give the error message:

value	description
6000	Accept
5555	Accept
4000	Error message
8000	Error message
blank	Error message
100\$	Error message
----	----
----	----
Maximum test coverage	

Note:

Condition: if amount >5000 and amount<7000 amount

And, if we enter 5000 → error message (not accepted based on the condition)

7000→ error message (not accepted based on the condition)

Advantages and disadvantage of Error guessing technique

Advantages

The benefits of error guessing technique are as follows:

- It is a good approach to find the challenging parts of the software.
- It is beneficial when we will use this technique with the grouping of other formal testing techniques.
- It is used to enhance the formal test design techniques.
- With the help of this technique, we can disclose those bugs which were probably identified over extensive testing; therefore, the test engineer can save lots of time and effort.

Disadvantage

Following are the drawbacks of error guessing technique:

- The error guessing technique is person-oriented rather than process-oriented because it depends on the person's thinking.
- If we use this technique, we may not achieve the minimum test coverage.

- With the help of this, we may not cover all the input or boundary values.
- With this, we cannot give the surety of the product quality.
- The Error guessing technique can be done by those people who have product knowledge; it cannot be done by those who are new to the product.

Equivalence Partitioning Technique

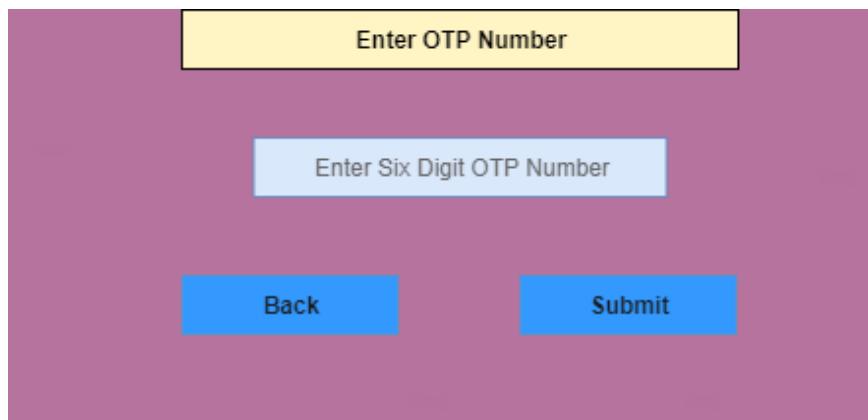
Equivalence partitioning is a technique of software testing in which input data is divided into partitions of valid and invalid values, and it is mandatory that all partitions must exhibit the same behavior. If a condition of one partition is true, then the condition of another equal partition must also be true, and if a condition of one partition is false, then the condition of another equal partition must also be false. The principle of equivalence partitioning is, test cases should be designed to cover each partition at least once. Each value of every equal partition must exhibit the same behavior as other.

The equivalence partitions are derived from requirements and specifications of the software. The advantage of this approach is, it helps to reduce the time of testing due to a smaller number of test cases from infinite to finite. It is applicable at all levels of the testing process.

Examples of Equivalence Partitioning technique

Assume that there is a function of a software application that accepts a particular number of digits, not greater and less than that particular number. For example, an OTP number which contains only six digits, less or more than six digits will not be accepted, and the application will redirect the user to the error page.

1. OTP Number = 6 digits



INVALID	INVALID	VALID	VALID
1 Test case	2 Test case	3 Test case	
DIGITS >=7	DIGITS<=5	DIGITS = 6	DIGITS = 6
93847262	9845	456234	451483

Let's see one more example.

A function of the software application accepts a 10 digit mobile number.

2. Mobile number = 10 digits

	Enter Mobile Number
Enter Ten Digit Mobile Number	
Back	Submit

INVALID 1 Test case	INVALID 2 Test case	VALID 3 Test case	VALID
DIGITS >=11	DIGITS<=9	DIGITS = 10	DIGITS =10
93847262219	984543985	9991456234	9893451483

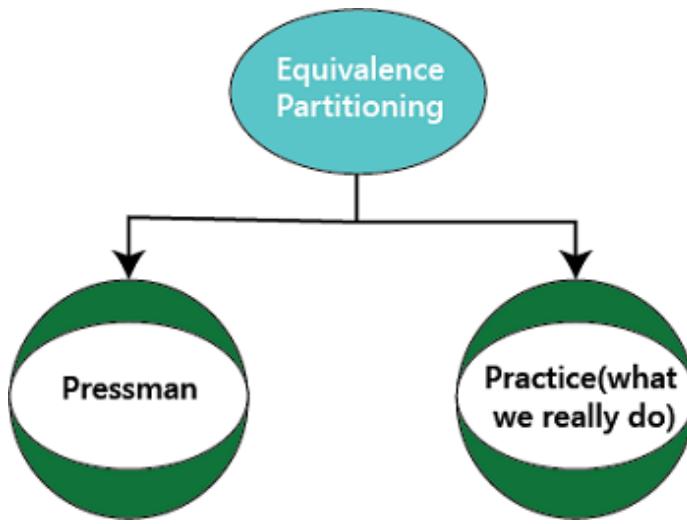
In both examples, we can see that there is a partition of two equally valid and invalid partitions, on applying valid value such as OTP of six digits in the first example and mobile number of 10 digits in the second example, both valid partitions behave same, i.e. redirected to the next page.

Another two partitions contain invalid values such as 5 or less than 5 and 7 or more than 7 digits in the first example and 9 or less than 9 and 11 or more than 11 digits in the second example, and on applying these invalid values, both invalid partitions behave same, i.e. redirected to the error page.

We can see in the example, there are only three test cases for each example and that is also the principle of equivalence partitioning which states that this method intended to reduce the number of test cases.

How we perform equivalence partitioning

We can perform equivalence partitioning in two ways which are as follows:



Let us see how pressman and general practice approaches are going to use in different conditions:

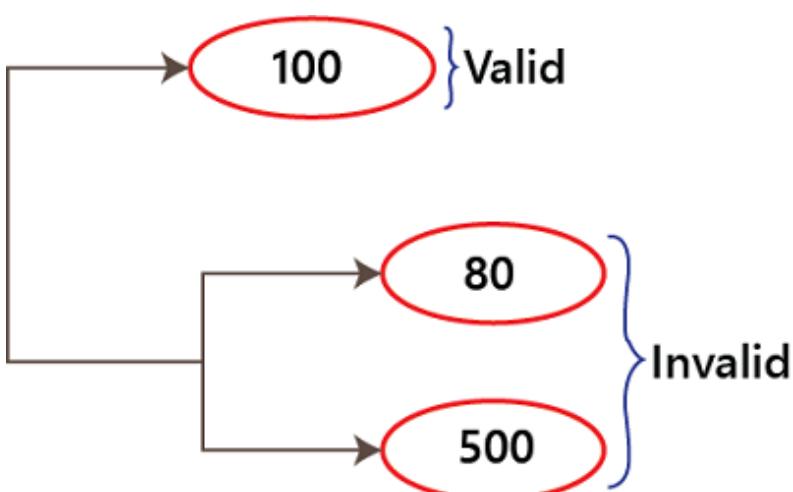
Condition1

If the requirement is **a range of values**, then derive the test case for **one valid** and **two invalid** inputs.

Here, the **Range of values** implies that whenever we want to identify the range values, we go for equivalence partitioning to achieve the minimum test coverage. And after that, we go for error guessing to achieve maximum test coverage.

According to pressman:

For example, the Amount of test field accepts a Range (100-400) of values:



According to General Practice method:

Whenever the requirement is Range + criteria, then divide the Range into the internals and check for all these values.

For example:

In the below image, the pressman technique is enough to test for an age text field for one valid and two invalids. But, if we have the condition for insurance of ten years and above are required and multiple policies for various age groups in the age text field, then we need to use the practice method.

Condition2

If the requirement is **a set of values**, then derive the test case for **one valid** and **two invalid** inputs.

Here, **Set of values** implies that whenever we have to test a set of values, we go for **one positive** and **two negative** inputs, then we move for error guessing, and we also need to verify that all the sets of values are as per the requirement.

Example 1

Based on the Pressman Method

If the Amount Transfer is (100000-700000)

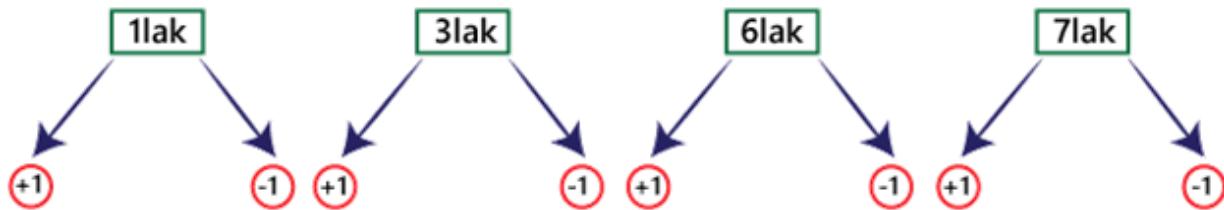
Then for, 1 lakh → Accept

And according to General Practice method

The Range + Percentage given to 1 lakh - 7 lakh

Like: 1lak - 3lak → 5.60%

3lak - 6lak → 3.66%



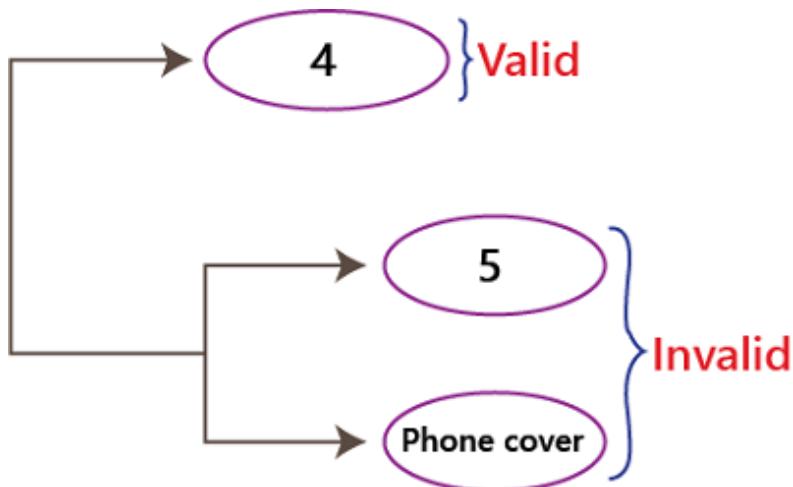
If we have things like loans, we should go for the general practice approach and separate the stuff into the intervals to achieve the minimum test coverage.

Example 2

if we are doing online shopping, mobile phone product, and the different **Product ID -1,4,7,9**

Here, **1 → phone covers 4 → earphones 7 → charger 9 → Screen guard**

And if we give the product id as **4**, it will be accepted, and it is one valid value, and if we provide the product id as **5 and phone cover**, it will not be accepted as per the requirement, and these are the two invalid values.



Condition 3

If the requirement id **Boolean (true/false)**, then derive the test case for both true/false values.

The Boolean value can be true and false for the radio button, checkboxes.

For example

Serial no	Description	Input	Expected	Note
1	Select valid	NA	True	---
2	Select invalid	NA	False	Values can be change based according to the requirement.
3	Do not select	NA	Do not select anything, error message should be displayed	We cannot go for next question
4	Select both	NA	We can select any radio button	Only one radio button can be selected at a time.

Note:

In **Practice** method, we will follow the below process:

Here, we are testing the application by deriving the below inputs values:

80	500	1000	2000	3000	4000	5000	6000	7000
----	-----	------	------	------	------	------	------	------

Let us see one program for our better understanding.

```

if( amount < 500 or > 7000)
{
    Error Message
}

if( amount is between 500 & 3000)
{
    deduct 2%
}

if (amount > 3000)
{
    deduct 3%
}

```

When the pressman technique is used, the first two conditions are tested, but if we use the practice method, all three conditions are covered.

We don't need to use the practice approach for all applications. Sometime we will use the pressman method also.

But, if the application has much precision, then we go for the practice method.

If we want to use the practice method, it should follow the below aspects:

- It should be product-specific
- It should be case-specific
- The number of divisions depends on the precision(2% and 3 % deduction)

Advantages and disadvantages of Equivalence Partitioning technique

Following are pros and cons of equivalence partitioning technique:

Advantages	disadvantages
It is process-oriented	All necessary inputs may not cover.
We can achieve the Minimum test coverage	This technique will not consider the condition for boundary value analysis.
It helps to decrease the general test execution time and also reduce the set of test data.	The test engineer might assume that the output for all data set is right, which leads to the problem during the testing process.

Boundary Value Analysis

Boundary value analysis is one of the widely used case design technique for black box testing. It is used to test boundary values because the input values near the boundary have higher chances of error.

Whenever we do the testing by boundary value analysis, the tester focuses on, while entering boundary value whether the software is producing correct output or not.

Boundary values are those that contain the upper and lower limit of a variable. Assume that, age is a variable of any function, and its minimum value is 18 and the maximum value is 30, both 18 and 30 will be considered as boundary values.

The basic assumption of boundary value analysis is, the test cases that are created using boundary values are most likely to cause an error.

There is 18 and 30 are the boundary values that's why tester pays more attention to these values, but this doesn't mean that the middle values like 19, 20, 21, 27, 29 are ignored. Test cases are developed for each and every value of the range.

The form consists of four input fields:

- Name:** Enter Your Name
- Age:** Between 18 to 30
- Adhar:** Number of 12 Digits
- Address:** Enter Your Address

Testing of boundary values is done by making valid and invalid partitions. Invalid partitions are tested because testing of output in adverse condition is also essential.

Let's understand via practical:

Imagine, there is a function that accepts a number between 18 to 30, where 18 is the minimum and 30 is the maximum value of valid partition, the other values of this partition are 19, 20, 21, 22, 23, 24, 25, 26, 27, 28 and 29. The invalid partition consists of the numbers which are less than 18 such as 12, 14, 15, 16 and 17, and more than 30 such as 31, 32, 34, 36 and 40. Tester develops test cases for both valid and invalid partitions to capture the behavior of the system on different input conditions.



Invalid test cases	Valid test cases	Invalid test cases
11, 13, 14, 15, 16, 17	18, 19, 24, 27, 28, 30	31, 32, 36, 37, 38, 39

The software system will be passed in the test if it accepts a valid number and gives the desired output, if it is not, then it is unsuccessful. In another scenario, the software system should not accept invalid numbers, and if the entered number is invalid, then it should display error message.

If the software which is under test, follows all the testing guidelines and specifications then it is sent to the releasing team otherwise to the development team to fix the defects.

Test Plan

A test plan is a detailed document which describes software testing areas and activities. It outlines the test strategy, objectives, test schedule, required resources (human resources, software, and hardware), test estimation and test deliverables.

The test plan is a base of every software's testing. It is the most crucial activity which ensures availability of all the lists of planned activities in an appropriate sequence.

The test plan is a template for conducting software testing activities as a defined process that is fully monitored and controlled by the testing manager. The test plan is prepared by the Test Lead (60%), Test Manager(20%), and by the test engineer(20%).

Types of Test Plan

There are three types of the test plan

- Master Test Plan
- Phase Test Plan
- Testing Type Specific Test Plans

Master Test Plan

Master Test Plan is a type of test plan that has multiple levels of testing. It includes a complete test strategy.

Phase Test Plan

A phase test plan is a type of test plan that addresses any one phase of the testing strategy. For example, a list of tools, a list of test cases, etc.

Specific Test Plans

Specific test plan designed for major types of testing like security testing, load testing, performance testing, etc. In other words, a specific test plan designed for non-functional testing.

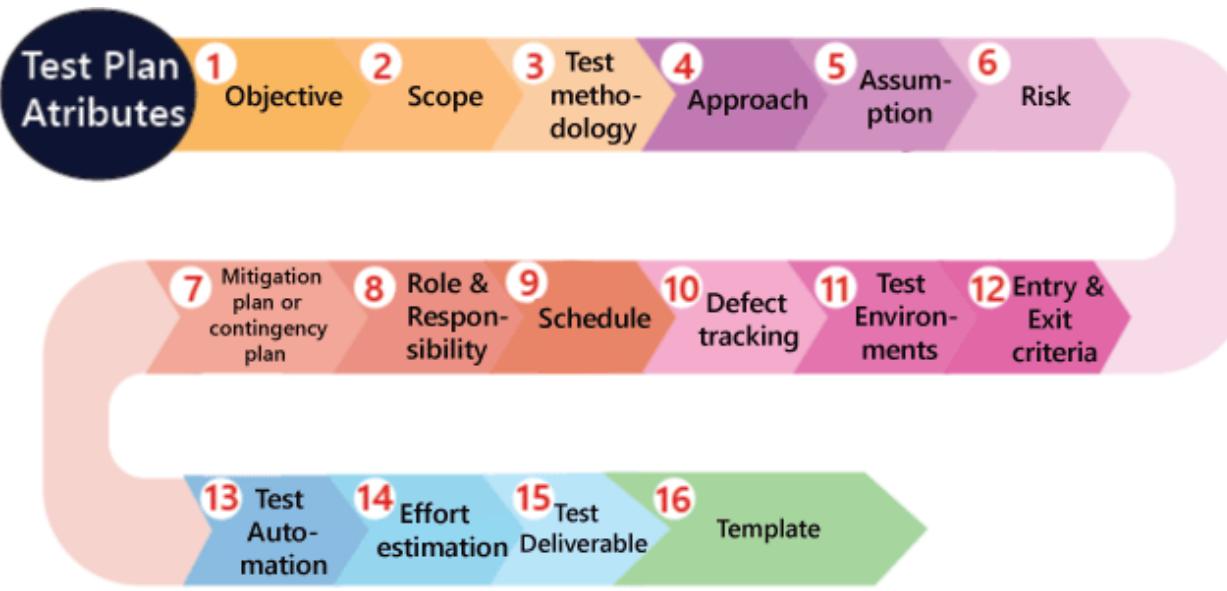
How to write a Test Plan

Making a test plan is the most crucial task of the test management process. According to IEEE 829, follow the following seven steps to prepare a test plan.

- First, analyze product structure and architecture.
- Now design the test strategy.
- Define all the test objectives.
- Define the testing area.
- Define all the useable resources.
- Schedule all activities in an appropriate manner.
- Determine all the Test Deliverables.

Test plan components or attributes

The test plan consists of various parts, which help us to derive the entire testing activity.



Objectives: It consists of information about modules, features, test data etc., which indicate the aim of the application means the application behavior, goal, etc.

Scope: It contains information that needs to be tested with respective of an application. The Scope can be further divided into two parts:

- In scope
- Out scope

In scope: These are the modules that need to be tested rigorously (in-detail).

Out scope: These are the modules, which need not be tested rigorously.

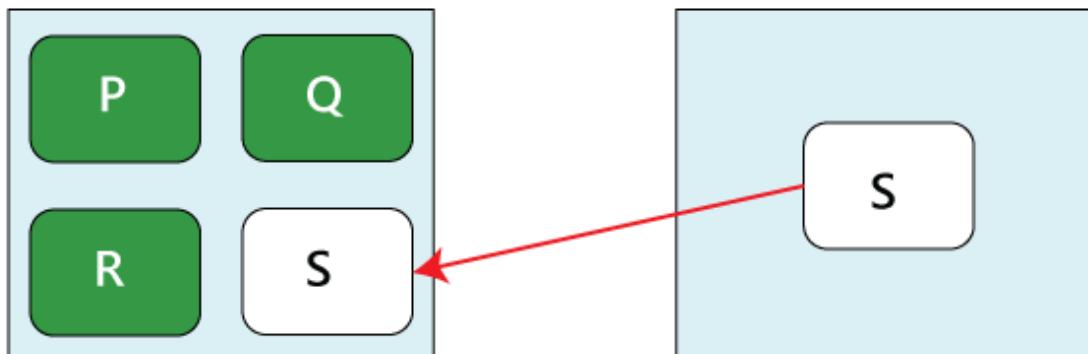
For example, Suppose we have a Gmail application to test, where **features to be tested** such as **Compose mail, Sent Items, Inbox, Drafts** and the **features which not be tested** such as **Help**, and so on which means that in the planning stage, we will decide that which functionality has to be checked or not based on the time limit given in the product.

Now **how we decide which features not to be tested?**

We have the following aspects where we can decide which feature not to be tested:

- As we see above that **Help** features is not going to be tested, as it is written and developed by the technical writer and reviewed by another professional writer.
- Let us assume that we have one application that have **P, Q, R, and S** features, which need to be developed based on the requirements. But here, the S feature has already been designed and used by some other company. So the development team will purchase S from that company and integrate with additional features such as P, Q, and R.

Now, we will not perform functional testing on the S feature because it has already been used in real-time. But we will do the integration testing, and system testing between P, Q, R, and S features because the new features might not work correctly with S feature as we can see in the below image:



- Suppose in the first release of the product, the elements that have been developed, such as **P, Q, R, S, T, U, V, W....X, Y, Z**. Now the client will provide the requirements for the new features which improve the product

in the second release and the new features are **A1, B2, C3, D4, and E5**.

After that, we will write the scope during the test plan as

Scope

Features to be tested

A1, B2, C3, D4, E5 (new features)

P, Q, R, S, T

Features not to be tested

W....X, Y, Z

Therefore, we will check the new features first and then continue with the old features because that might be affected after adding the new features, which means it will also affect the impact areas, so we will do one round of regressing testing for P, Q, R..., T features.

Test methodology:

It contains information about performing a different kind of testing like Functional testing, Integration testing, and System testing, etc. on the application. In this, we will decide what type of testing; we will perform on the various features based on the application requirement. And here, we should also define that what kind of testing we will use in the testing methodologies so that everyone, like the management, the development team, and the testing team can understand easily because the testing terms are not standard.

For example, for standalone application such as **Adobe Photoshop**, we will perform the following types of testing:

Smoke testing → Functional testing → Integration testing → System testing → Adhoc testing → Compatibility testing → Regression testing → Globalization testing → Accessibility testing → Usability testing → Reliability testing → Recovery testing → Installation or Uninstallation testing

And suppose we have to test the application, so we will perform following types of testing:

Smoke testing	Functional testing	Integration testing
System testing	Adhoc testing	Compatibility testing
Regression testing	Globalization testing	Accessibility testing
Usability testing	Performance testing	

Approach

This attribute is used to describe the flow of the application while performing testing and for the future reference.

We can understand the flow of the application with the help of below aspects:

- **By writing the high-level scenarios**
- **By writing the flow graph**

By writing the high-level scenarios

For example, suppose we are testing the **Gmail** application:

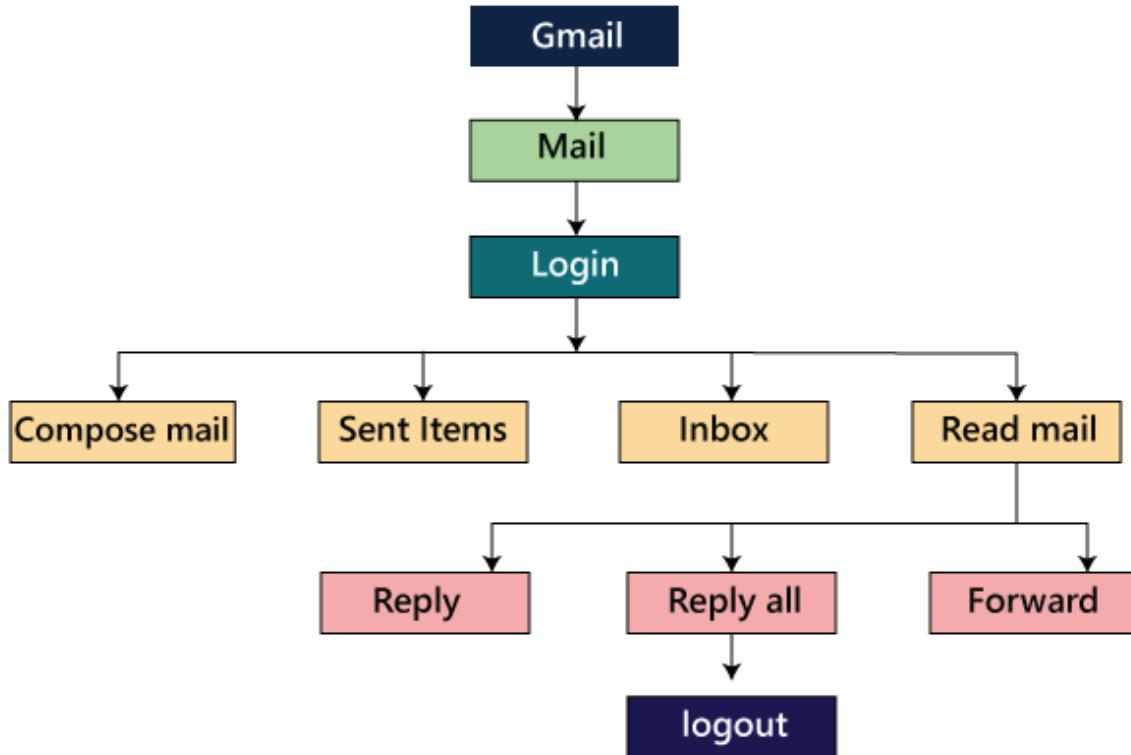
- Login to Gmail- sends an email and check whether it is in the Sent Items page
- Login to
-

-

We are writing this to describe the approaches which have to be taken for testing the product and only for the critical features where we will write the high-level scenarios. Here, we will not be focusing on covering all the scenarios because it can be decided by the particular test engineer that which features have to be tested or not.

By writing the flow graph

The flow graph is written because writing the high-level scenarios are bit time taking process, as we can see in the below image:



We are creating flow graphs to make the following benefits such as:

- Coverage is easy
- Merging is easy

The approach can be classified into two parts which are as following:

- Top to bottom approach
- Bottom to top approach

Assumption

It contains information about a problem or issue which maybe occurred during the testing process and when we are writing the test plans, the assured assumptions would be made like resources and technologies, etc.

Risk

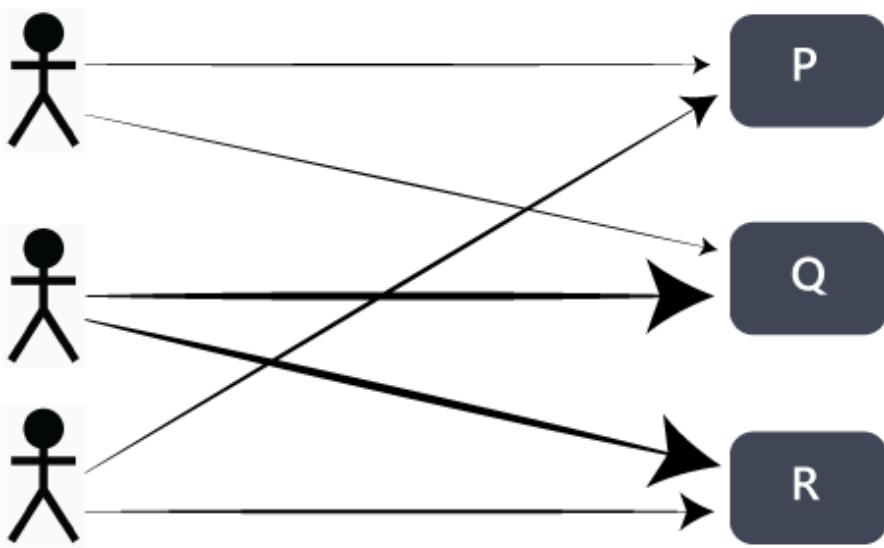
These are the challenges which we need to face to test the application in the current release and if the assumptions will fail then the risks are involved.

For example, the effect for an application, release date becomes postponed.

Mitigation Plan or Contingency Plan

It is a back-up plan which is prepared to overcome the risks or issues.

Let us see one example for assumption, risk, and the contingency plan together because they are co-related to each other.



In any product, the **assumption** we will make is that all 3 test engineers will be there until the completion of the product and each of them is assigned different modules such as P, Q, and R. In this particular scenario, the **risk** could be that if the test engineer left the project in the middle of it.

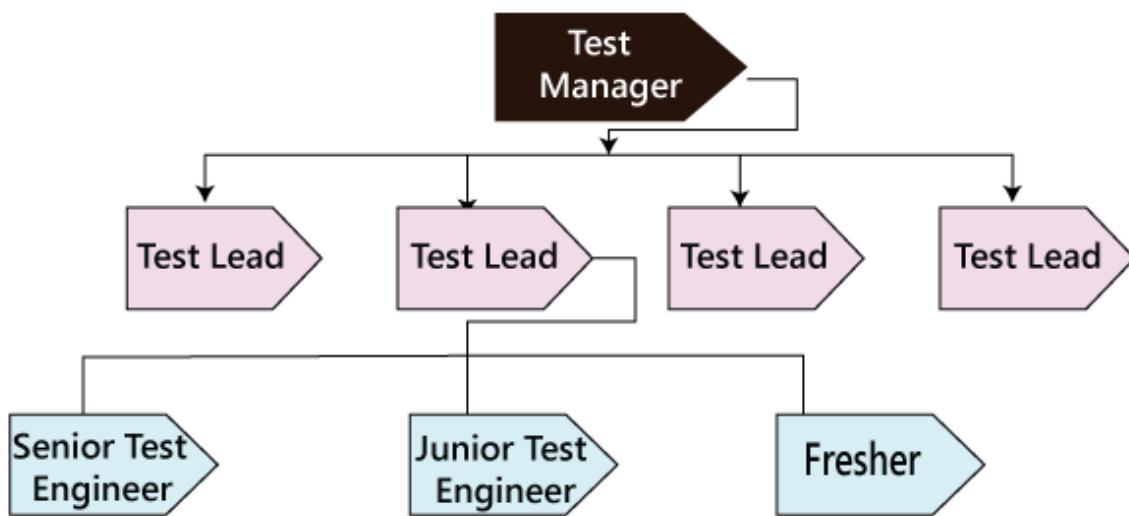
Therefore, the **contingency plan** will be assigned a primary and subordinate owner to each feature. So if one test engineer will leave, the subordinate owner takes over that specific feature and also helps the new test engineer, so he/she can understand their assigned modules.

The assumptions, risk, and mitigation or contingency plan are always precise on the product itself. The various types of risks are as follows:

- Customer perspective
- Resource approach
- Technical opinion

Role & Responsibility

It defines the complete task which needs to be performed by the entire testing team. When a large project comes, then the **Test Manager** is a person who writes the test plan. If there are 3-4 small projects, then the test manager will assign each project to each Test Lead. And then, the test lead writes the test plan for the project, which he/she is assigned.



Let see one example where we will understand the roles and responsibility of the Test manager, test lead, and the test engineers.

Role: Test Manager

Name: Ryan

Responsibility:

- Prepare(write and review) the test plan

- Conduct the meeting with the development team
- Conduct the meeting with the testing team
- Conduct the meeting with the customer
- Conduct one monthly stand up meeting
- Sign off release note
- Handling Escalations and issues

Role: Test Lead

Name: Harvey

Responsibility:

- Prepare(write and review) the test plan
- Conduct daily stand up meeting
- Review and approve the test case
- Prepare the RTM and Reports
- Assign modules
- Handling schedule

Role: Test Engineer 1, Test Engineer 2 and Test Engineer 3

Name: Louis, Jessica, Donna

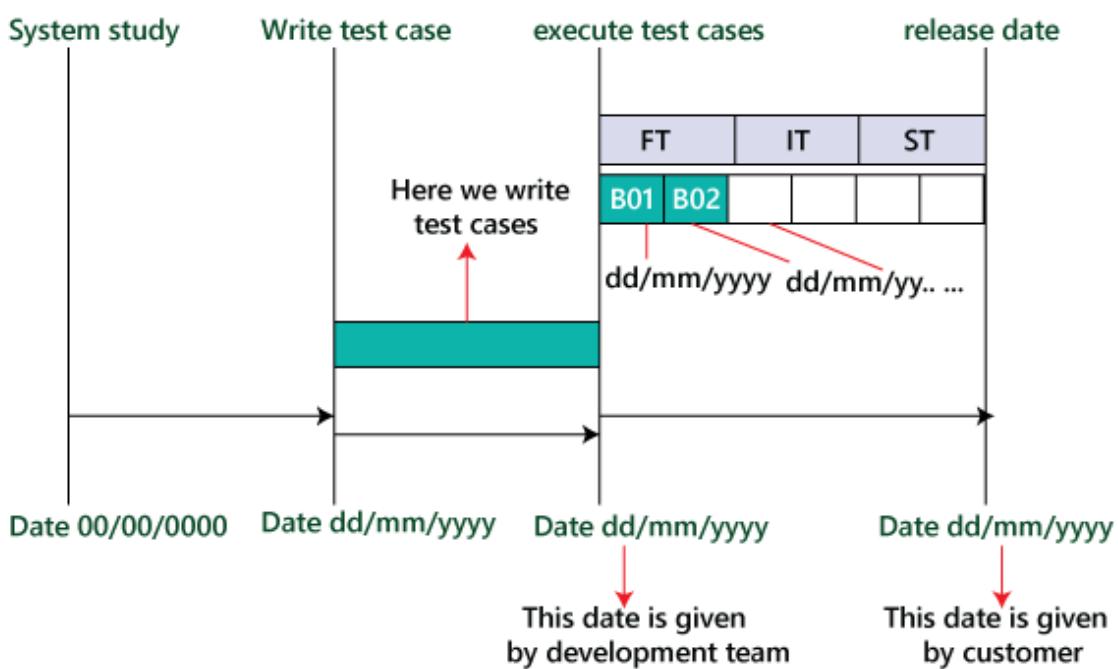
Assign modules: M1, M2, and M3

Responsibility:

- Write, Review, and Execute the test documents which consists of test case and test scenarios
- Read, review, understand and analysis the requirement
- Write the flow of the application
- Execute the test case
- RTM for respective modules
- Defect tracking
- Prepare the test execution report and communicate it to the Test Lead.

Schedule

It is used to explain the timing to work, which needs to be done or this attribute covers when exactly each testing activity should start and end? And the exact date is also mentioned for every testing activity for the particular date.



Therefore as we can see in the below image that for the particular activity, there will be a starting date and ending date; for each testing to a specific build, there will be the specified date.

For example

- Writing test cases
- Execution process

Defect tracking

It is generally done with the help of tools because we cannot track the status of each bug manually. And we also comment about how we communicate the bugs which are identified during the testing process and send it back to the development team and how the development team will reply. Here we also mention the priority of the bugs such as high, medium, and low.

Following are various aspects of the defect tracking:

- **Techniques to track the bug**
- **Bug tracking tools** We can comment on the name of the tool, which we will use for tracking the bugs. Some of the most commonly used bug tracking tools are Jira, Bugzilla, Mantis, and Trac, etc.<
- **Severity** The severity could be as following: **Blocker or showstopper** (Explain it with an example in the test plan) **For example**, there will be a defect in the module; we cannot go further to test other modules because if the bug is blocked, we can proceed to check other modules. **Critical** (Explain it with an example in the test plan) In this situation, the defects will affect the business. **Major** (Explain it with an example in the test plan) **Minor** (Explain it with an example in the test plan) These defects are those, which affect the look and feel of the application.
- **Priority High-P1 Medium-P2 Low-P3 P4**

Therefore, based on the priority of bugs like high, medium, and low, we will categorize it as P1, P2, P3, and P4.

Test Environments

These are the environments where we will test the application, and here we have two types of environments, which are of **software** and **hardware** configuration.

The **software configuration** means the details about different **Operating Systems** such as **Windows, Linux, UNIX, and Mac** and various **Browsers** like **Google Chrome, Firefox, Opera, Internet Explorer**, and so on.

And the **hardware configuration** means the information about different sizes of **RAM, ROM, and the Processors**.

For example

- The **Software** includes the following:

Server

Operating system	Linux
Webserver	Apache Tomcat
Application server	Websphere
Database server	Oracle or MS-SQL Server

Note: The above servers are the servers that are used by the testing team to test the application.

Client

Operating System	Windows XP, Vista 7
Browsers	Mozilla Firefox, Google Chrome, Internet Explorer, Internet Explorer 7, and Internet Explorer 8

Note: The above details provide the various operating systems and browsers in which the testing team will test the application.

- The **Hardware** includes the following:

Server: Sun StarCat 1500

This particular server can be used by the testing team to test their application.

Client:

It has the following configuration, which is as follows:

Processor	Intel 2GHz
RAM	2GB
....

Note: It will provide the configuration of the systems of the test engineers that is the testing team.

- **Process to install the software**

The development team will provide the configuration of how to install the software. If the development team will not yet provide the process, then we will write it as Task-Based Development (TBD) in the test plan.

Entry and Exit criteria

It is a necessary condition, which needs to be satisfied before starting and stopping the testing process.

Entry Criteria

The entry criteria contain the following conditions:

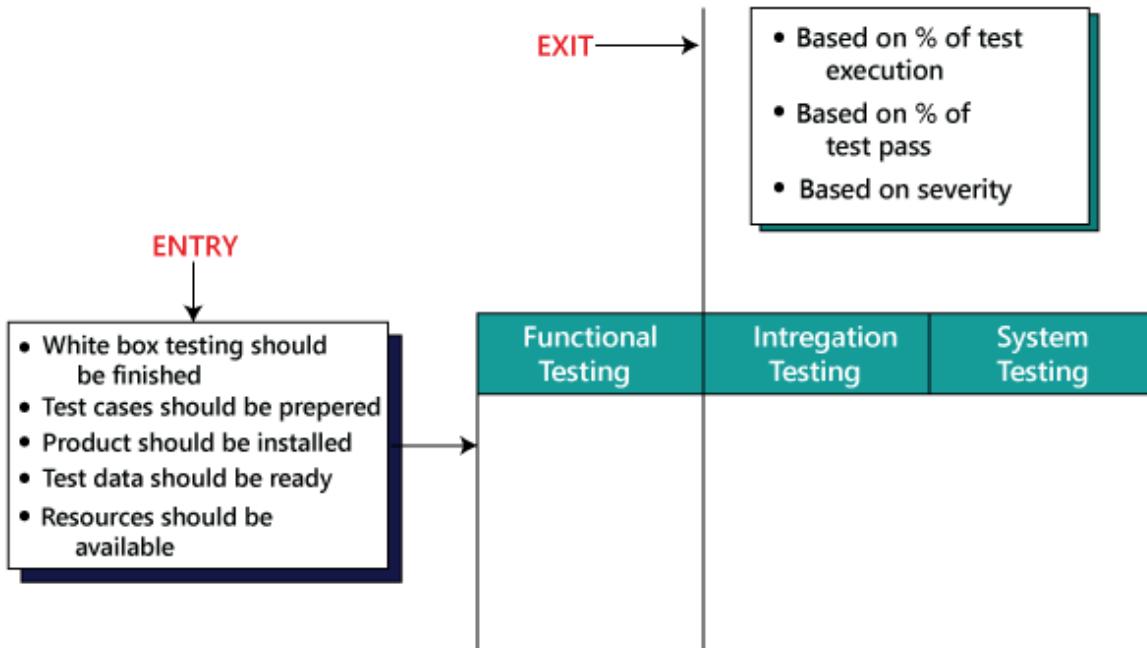
- White box testing should be finished.
- Understand and analyze the requirement and prepare the test documents or when the test documents are ready.
- Test data should be ready.
- Build or the application must be prepared
- Modules or features need to be assigned to the different test engineers.
- The necessary resource must be ready.

Exit Criteria

The exit criteria contain the following conditions:

- When all the test cases are executed.
- Most of the test cases must be **passed**.
- Depends on severity of the bugs which means that there must not be any blocker or major bug, whereas some minor bugs exist.

Before we start performing functional testing, all the above **Entry Criteria** should be followed. After we performed functional testing and before we will do integration testing, then the **Exit criteria of** the functional testing should be followed because the % of exit criteria are decided by the meeting with both development and test manager because their collaboration can achieve the percentage. But if the exit criteria of functional testing are not followed, then we cannot proceed further to integration testing.



Here **based on the severity** of the bug's means that the testing team would have decided that to proceed further for the next phases.

Test Automation

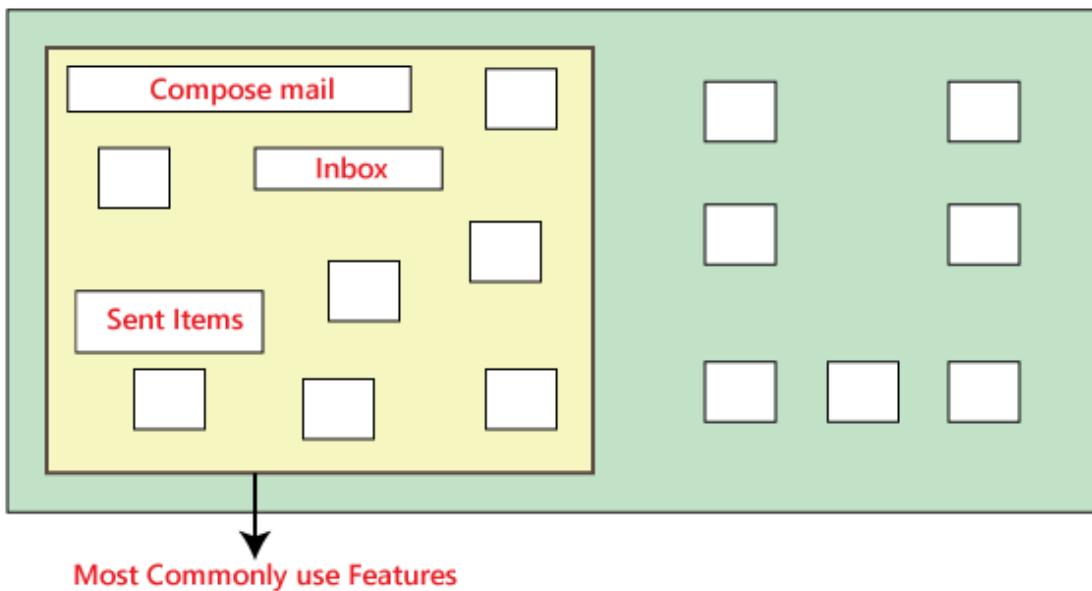
In this, we will decide the following:

- Which feature has to be automated and not to be automated?

- Which test automation tool we are going to use on which automation framework?

We automate the test case only after the first release.

Here the question arises that on what basis **we will decide which features have to be tested?**



In the above image, as we can see that the most commonly used features need to test again and again. Suppose we have to check the Gmail application where the essential features are **Compose mail, Sent Items, and Inbox**. So we will test these features because while performing manual testing, it takes more time, and it also becomes a monotonous job.

Now, **how we decide which features are not going to be tested?**

Suppose **the Help** feature of the Gmail application is not tested again and again because these features are not regularly used, so we do not need to check it frequently.

But if **some features are unstable and have lots of bugs**, which means that we will not test those features because it has to be tested again and again while doing manual testing.

If **there is a feature that has to be tested frequently**, but we are expecting the requirement change for that feature, so we do not check it because changing the manual test cases is more comfortable as compared to change in the automation test script.

Effort estimation

In this, we will plan the effort need to be applied by every team member.

Test Deliverable

These are the documents which are the output from the testing team, which we handed over to the customer along with the product. It includes the following:

- **Test plan**
- **Test Cases**
- **Test Scripts**
- **RTM(Requirement Traceability Matrix)**
- **Defect Report**
- **Test Execution Report**
- **Graphs and metrics**
- **Release Notes**

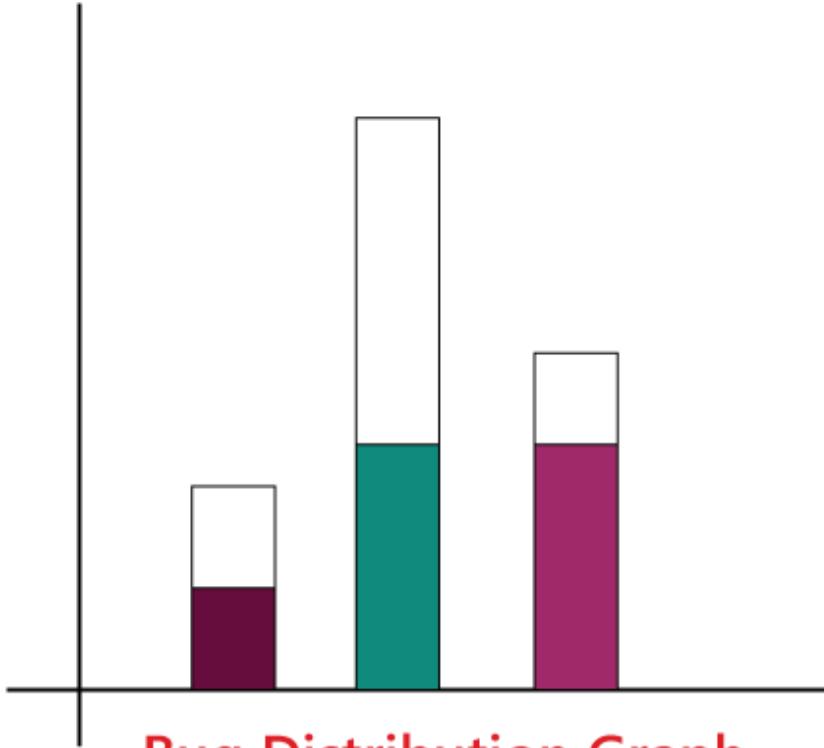
Graphs and Metrics

Graph

In this, we will discuss about the types of **graphs** we will send, and we will also provide a sample of each graph.

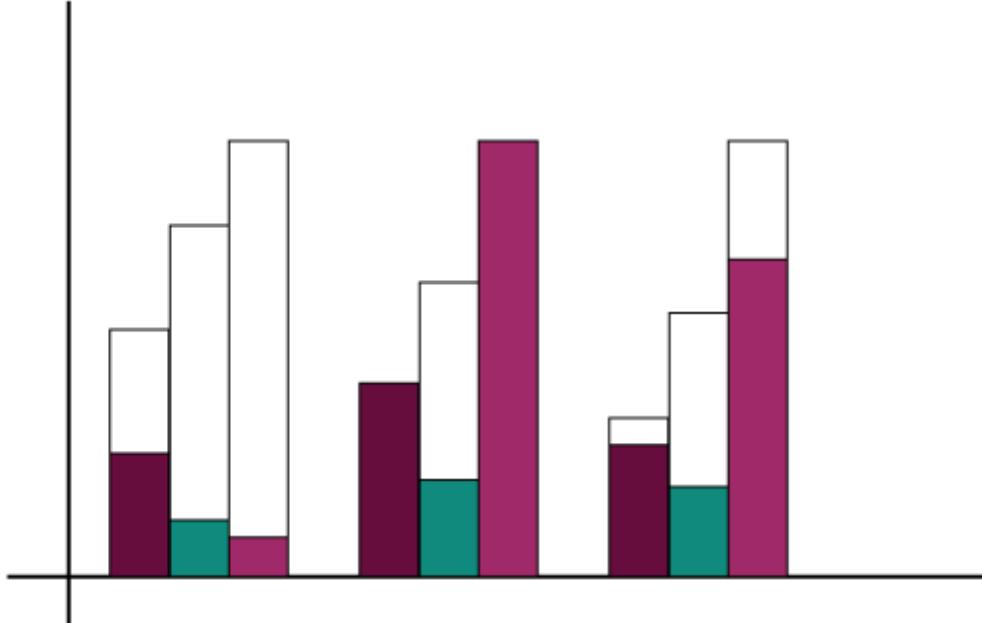
As we can see, we have five different graphs that show the various aspects of the testing process.

Graph1: In this, we will show how many defects have been identified and how many defects have been fixed in every module.

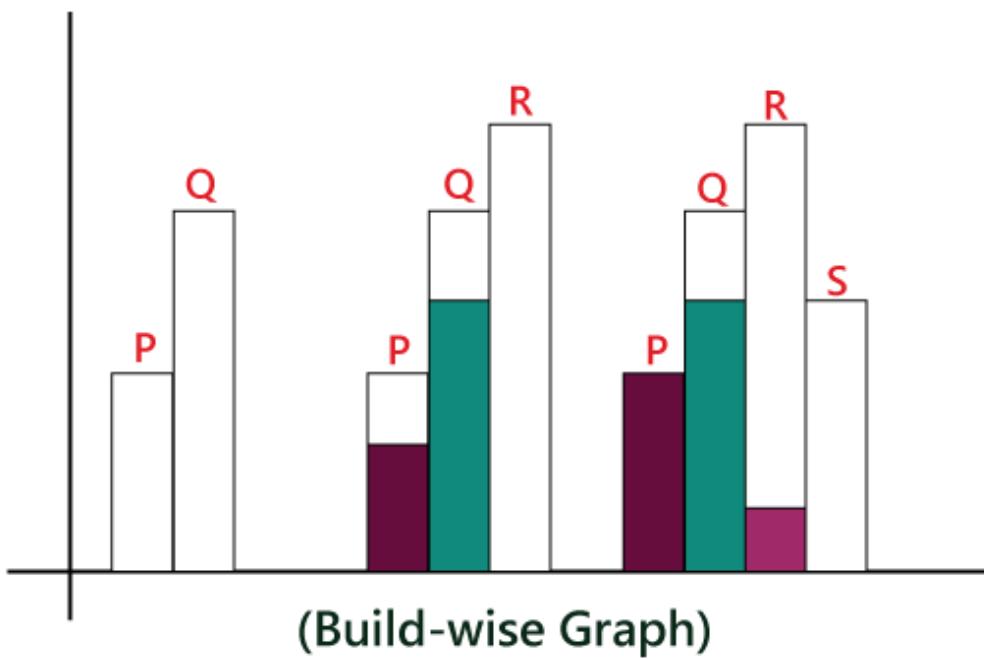


Bug Distribution Graph

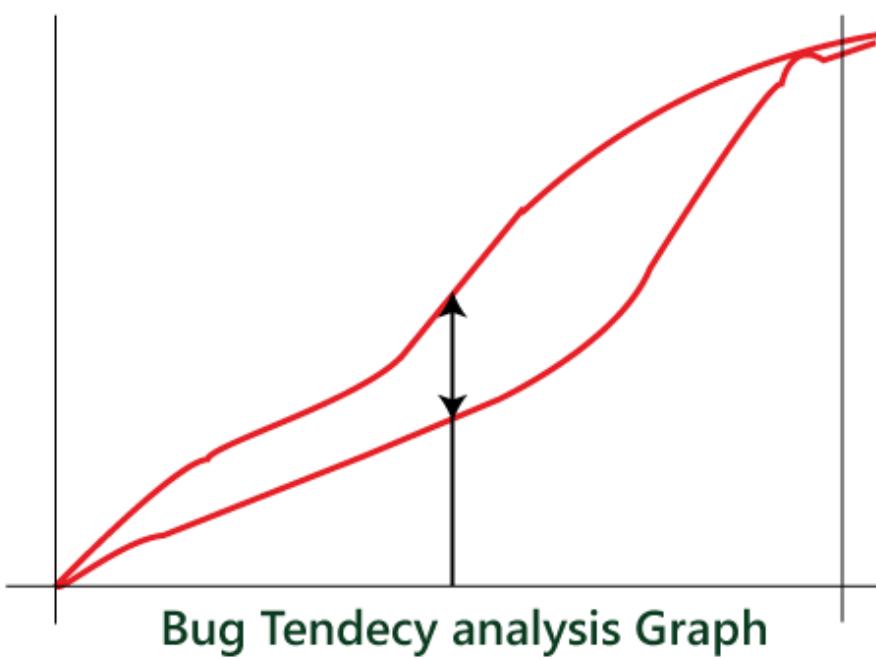
Graph 2: Figure one shows how many critical, major, and minor defects have been identified for every module and how many have been fixed for their respective modules.



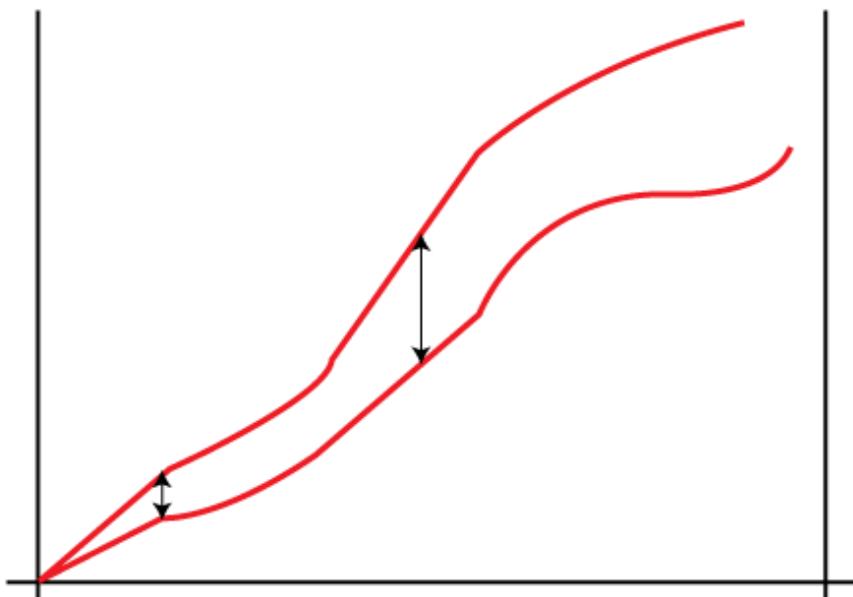
Graph3: In this particular graph, we represent the **build wise graph**, which means that in every builds how many defects have been identified and fixed for every module. Based on the module, we have determined the bugs. We will add **R** to show the number of defects in P and Q, and we also add **S** to show the defects in P, Q, and R.



Graph4: The test lead will design the **Bug Trend analysis** graph which is created every month and send it to the Management as well. And it is just like prediction which is done at the end of the product. And here, we can also **rate the bug fixes** as we can observe that **arc** has an **upward tendency** in the below image.



Graph5: The **Test Manager** has designed this type of graph. This graph is intended to understand the gap in the assessment of bugs and the actual bugs which have been occurred, and this graph also helps to improve the evaluation of bugs in the future.



Module Name	Critical		Major		Minor	
	Found	Fixed	Found	Fixed	Found	Fixed
Purchase	50	46	60	20	80	10
Sales
Asset Survey

As above, we create the bug distribution graph, which is in the figure 1, and with the help of above mention data, we will design the metrics as well.

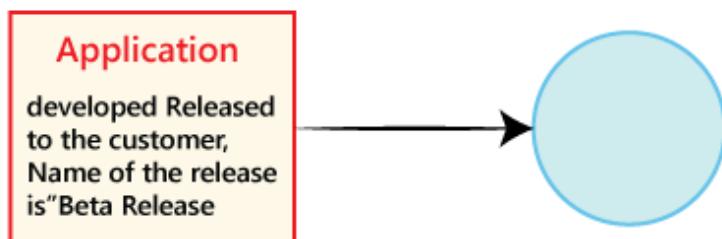
For example

Test Engineer Names	Critical		Major		Minor	
	Found	Fixed	Found	Fixed	Found	Fixed
John	50	46	60	20	80	10
James
Sophia

In the above figure, we retain the records of all the test engineers in a particular project and how many defects have been identified and fixed. We can also use this data for future analysis. When a new requirement comes, we can decide whom to provide the challenging feature for testing based on the number of defects they have found earlier according to the above metrics. And we will be in a better situation to know who can handle the problematic features very well and find maximum numbers of defects.

Release Note: It is a document that is prepared during the release of the product and signed by the Test Manager.

In the below image, we can see that the final product is developed and deployed to the customer, and the latest release name is **Beta**.



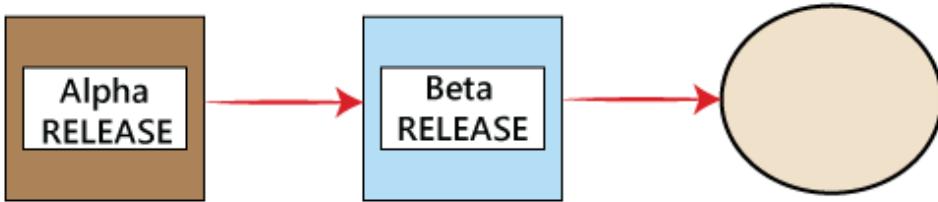
The **Release note** consists of the following:

- User manual.
- List of pending and open defects.
- List of added, modified, and deleted features.
- List of the platform (Operating System, Hardware, Browsers) on which the product is tested.
- Platform in which the product is not tested.
- List of bugs fixed in the current release, and the list of fixed bugs in the previous release.
- Installation process

- Versions of the software

For Example

Suppose that **Beta** is the second release of the application after the first release **Alpha** is released. Some of the defect identified in the first released and that has been fixed in the later released. And here, we will also point out the list of newly added, modified, and deleted features from alpha release to the beta release.



Template

This part contains all the templates for the documents that will be used in the product, and all the test engineers will use only these templates in the project to maintain the consistency of the product. Here, we have different types of the template which are used during the entire testing process such as:

- Test case template
- Test case review template
- RTM Template
- Bug Report Template
- Test execution Report

Let us see one sample of test plan document

Page-1

<u>Testplan</u>						
Version	Author	Reviewed By	Approved By	Comments	Approval date	
1	Name of manager	Version 1.0 is developed	dd/mm/yyyy	
1.1	Version 1.1 is developed. PQR feature is added	dd/mm/yyyy	
.....	
.....	

TABLE OF CONTENTS

- Objective pg 1
- Scope pg 2
- Approach pg 3
- ...
- ...
- ...
- ...
- ...



Entire Test Plan document

Page-20

REFERENCES

- 1) Customer requirement specification(CRS)
- 2) Software requirement specification(SRS)
- 3) Functional specification(FS)
- 4) Design Documents
- ...
- ...
- ...

In-Page 1, we primarily fill only the **Versions**, **Author**, **Comments**, and **Reviewed By** fields, and after the manager approves it, we will mention the details in the **Approved By** and **Approval Date** fields.

Mostly the test plan is approved by the Test Manager, and the test engineers only reviews it. And when the new features come, we will modify the test plan and do the necessary modification in **Version** field, and then it will be sent again for further review, update, and approval of the manager. The test plan must be updated whenever any changes have occurred. On page 20, the **References** specify the details about all the documents which we are going to use to write the test plan document.

Note:

Who writes the test plan?

- Test Lead→60%
- Test Manager→20%
- Test Engineer→20%

Therefore, as we can see from above that in 60% of the product, the test plan is written by the Test Lead.

Who reviews the Test Plan?

- Test Lead
- Test Manager
- Test engineer
- Customer
- Development team

The Test Engineer review the Test plan for their module perspective and the test manager review the Test plan based on the customer opinion.

Who approve the test Plan?

- Customer
- Test Manager

Who writes the test case?

- Test Lead
- Test Engineer

Who review the test case?

- Test Engineer
- Test Lead
- Customer
- Development Team

Who approves the Test cases?

- Test Manager
- Test Lead
- Customer

Test Plan Guidelines

- Collapse your test plan.
- Avoid overlapping and redundancy.
- If you think that you do not need a section that is already mentioned above, then delete that section and proceed ahead.
- Be specific. For example, when you specify a software system as the part of the test environment, then mention the software version instead of only name.
- Avoid lengthy paragraphs.
- Use lists and tables wherever possible.
- Update plan when needed.
- Do not use an outdated and unused document.

Importance of Test Plan

- The test plan gives direction to our thinking. This is like a rule book, which must be followed.
- The test plan helps in determining the necessary efforts to validate the quality of the software application under the test.
- The test plan helps those people to understand the test details that are related to the outside like developers, business managers, customers, etc.
- Important aspects like test schedule, test strategy, test scope etc are documented in the test plan so that the management team can review them and reuse them for other similar projects.

Test case review process

When the test engineer writes a test case, he/she may skip some scenarios, inputs and writes wrong navigation steps, which may affect the entire test execution process.

To avoid this, we will do one round of **review and approval process** before starting test execution.

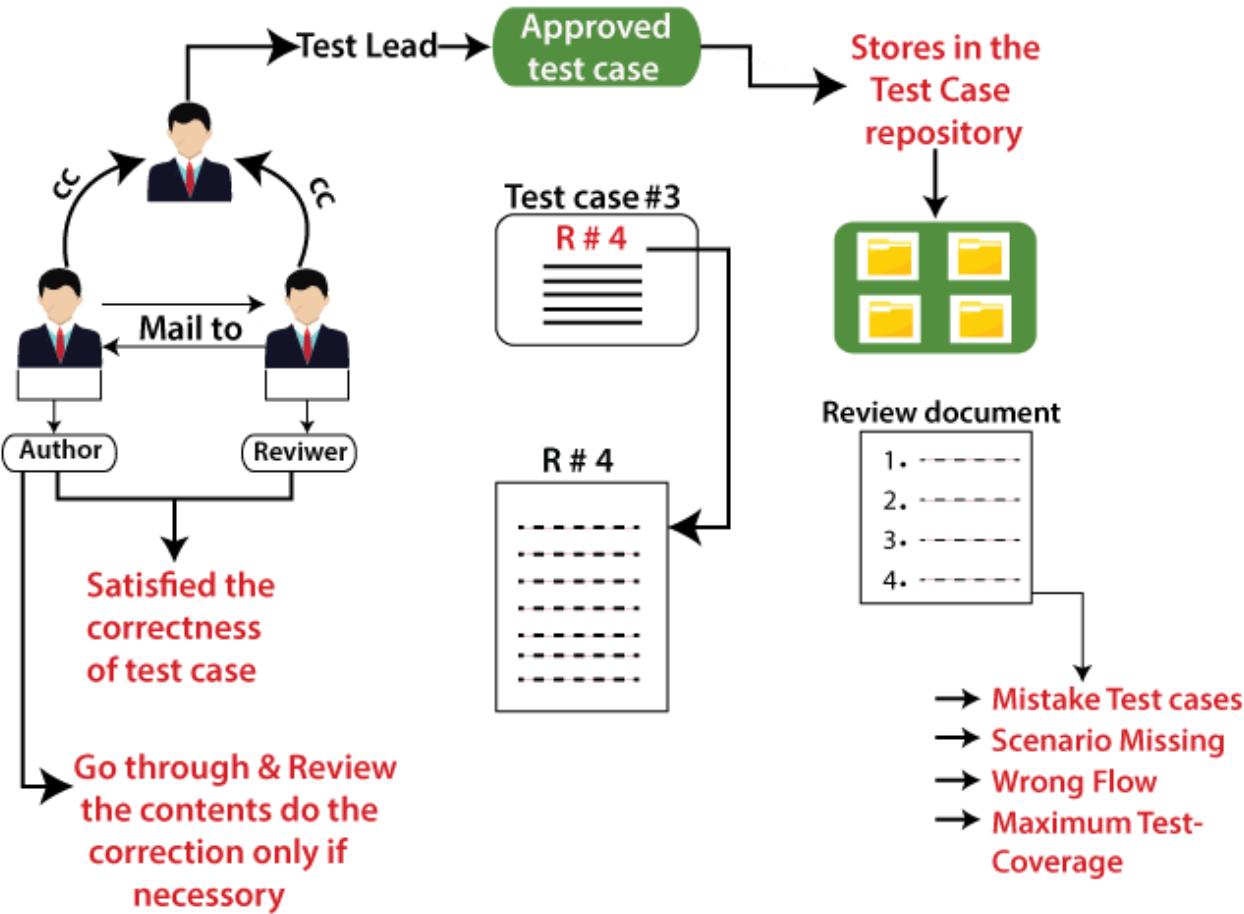
If we don't go for the review process, we **miss out some scenarios, accuracy won't be there, and the test engineer won't be serious.**

All the cases need to be sent for the review process only after the completion of writing the test case. So, the other person does not get disturbed.

Once the author finishes writing the test case, it needs to be sent for the other test engineer known as a **reviewer** for the review process.

The reviewer opens the **test case** with the corresponding requirement and checks the **correctness of the test case, proper flow, and maximum test coverage.**

During this review process, if the reviewer found any mistake, he/she writes it in a separate document, which is known as **Review document** and sends it back to the author.



The author goes through all the review comments and starts doing the changes if it is necessary, then send it back once again for the review process.

This correction process will continue until both authors, and the reviewer will satisfy.

Once the review is successful, the reviewer sends it back to the **test lead** for the final approval process.

During this approval process, the **Team leads** are always kept in the loop so that the author and reviewer will be serious in their jobs.

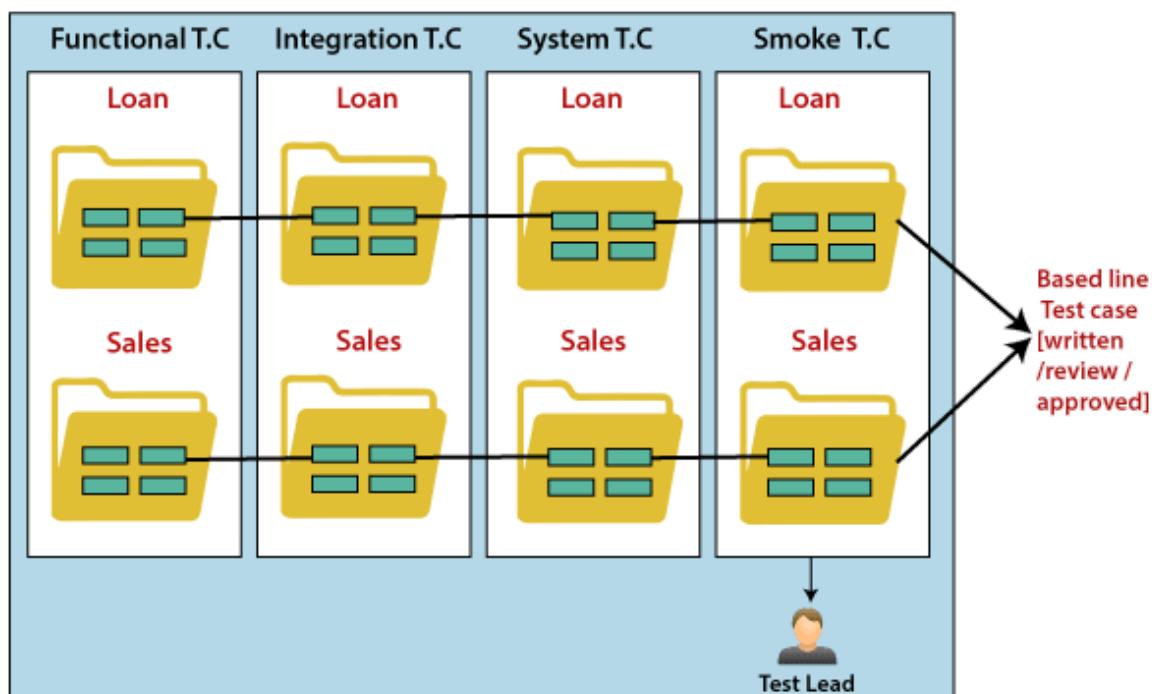
When the test cases are written, review, and approved, it will be stored in one centralized location, which is known as **Test Case Repository**.

Note:

Test Case Repository

- A test case repository is a centralized location where all the baseline test cases (written, review, and approved) are stored.
- When the client gives the requirements, the developer starts developing the modules, and the test engineer will write the test cases according to the requirements.
- A test case repository is used to store the approved test cases.
- Any test engineer wants to test the application, then he/she needs to access the test case only from the test case repository.
- If we do not require any test case, we can drop them from the test case repository.
- For every release, we maintain a different test case repository.
- Once the test cases are baselined or stored in the test case repository, they cannot be edited or changed without the permission of the test lead.
- The testing team always has a complete back-up of the test case repository if any crash happened which is affecting the software.

Test Case Repository



Review Process

While reviewing, the reviewer checks the following aspect in the **test cases**:

Template

The reviewer checks whether the template is as per requirements for the product or not.

Header

In the header, we check the following aspects:

- All the attributes are captured or not.
- All the attributes are relevant or not.
- All the attributes are filled or not.

Body

In the body of the test case, we will check the following aspects:

- The test case should be prepared so that it should take minimum time for the execution process.

- All the possible scenarios are covered or not.
 - Look for the flow including maximum test Coverage
 - The test case design technique is applied or not.
 - The test case should be simple to understand
 - Proper navigation is written or not.

Once the test case is reviewed, the review comments will be sent to the test case review template.

Test Case Name	Step No.	Reviewer		Author Comments	Comments
		Comments	Severity		
Cheetah-20-ICICI-AT	N.A	Pre-condition is missing	Critical	Not Fixed "No Pre-condition"	-----
	6	Used More Negative value	Minor	Fixed	-----
	11	Positive Value missing	Critical	Fixed	-----
	19	Need More Positive Value	Major	-----	-----
	20	Sentence not cleared	Minor	Not Fixed	-----

The reviewer will use the above template and send the comments. If the author fixes the test case, he/she would report it as fixed.

Text Execution Report [Excel]

It is the final document, which is prepared by a test lead when the entire testing process is completed.

The test execution report defined the stability of the application and contained the information like the number of cases written, executed, pass, fail, and their percentage.

The test execution report is a final summary report based on which the quality of the application is defined, and it also helps in deciding that the application can be handed over to the customer or not.

Every module has a separate spreadsheet of their respective module.

Test Execution Report Template

Let see one example of a test execution report where we have different modules such as **Sales**, **Amount transfer**, **Tax**, **Loan**.

Module name	Total No. of T.C written	Total No. of T.C Execution	Total No. of T.C pass	Total No. of T.C Fail	Pass%	Fail%
Sales	170	170	168	2	98%	2%
Amount transfer	90	90	88	2	97%	3%
Tax	127	127	127	0	100%	0%
Loans	110	110	109	1	99%	1%
Total	497	497	492	5	97%	2%

Sheet1 Test execution report	Sheet2 Sales	Sheet3 Amount transfer	Sheet4 Tax	Sheet5 Loans
-------------------------------------	---------------------	-------------------------------	-------------------	---------------------

The test lead made this report, and the test engineer sends the individual features that he/she has tested and executed.

The test lead sends this report to the following:

- **Development Team**
- **Management**
- **Test manager**
- **Customer**

Where a development team needs the list of failed test cases.

As we can see in the below table that we have a list of test case names, related status, and comments.

The below table is showing the amount transfer test case data.

Amount Transfer

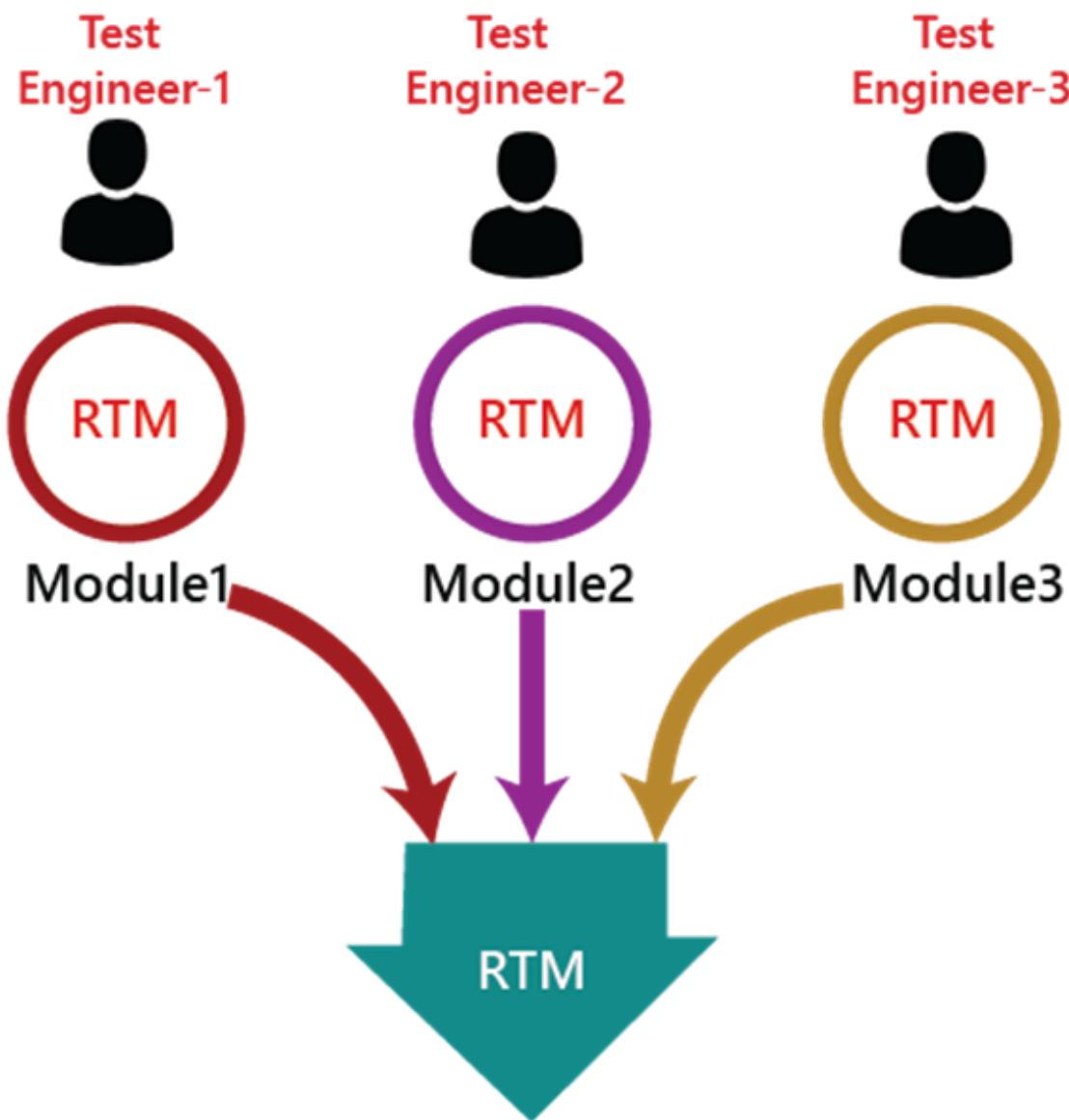
Step no.	Test case name	Status	Comments
1	Pass
2	Pass
3	Fail	Bug#9
4	pass
5	pass
6
.....
.....
.....
90	fail	Bug 29

Traceability Matrix

Traceability matrix is a table type document that is used in the development of software application to trace requirements. It can be used for both forward (from Requirements to Design or Coding) and backward (from Coding to Requirements) tracing. It is also known as **Requirement Traceability Matrix (RTM)** or **Cross Reference Matrix (CRM)**.

It is prepared before the test execution process to make sure that every requirement is covered in the form of a Test case so that we don't miss out any testing. In the RTM document, we map all the requirements and corresponding test cases to ensure that we have written all the test cases for each condition.

The **test engineer** will prepare RTM for their respective assign modules, and then it will be sent to the Test Lead. The Test Lead will go repository to check whether the Test Case is there or not and finally Test Lead consolidate and prepare one necessary RTM document.



This document is designed to make sure that each requirement has a test case, and the test case is written based on business needs, which are given by the client. It will be performed with the help of the test cases if any requirement is missing, which means that the test case is not written for a particular need, and that specific requirement is not tested because it may have some bugs. The traceability is written to make sure that the entire requirement is covered.

We can observe in the below image that the requirement number 2 and 4 test case names are not mentioned that's why we highlighted them, so that we can easily understand that we have to write the test case for them.

TRACEABILITY MATRIX

Requirement Number	Test Case Name
1	• • •
2	
3	• • •
4	
5	• • •
6	• • •
7	• • •
8	• • •

Generally, this is like a worksheet document, which contains a table, but there are also many user-defined templates for the traceability matrix. Each requirement in the traceability matrix is connected with its respective test case so that tests can be carried out sequentially according to specific requirements.

Note:

We go for RTM after approval and before execution so that we don't miss out on any Test Case for any requirement.

We don't write RTM while writing the testing because it can be incomplete, and after writing the test case, we don't go here because the test case can be rejected.

RTM document ensures that at least there is one test case written in each requirement, whereas it does not talk about all possible test cases written for the particular requirement.

RTM Template

Below is the sample template of requirement traceability matrix (RTM):

Requirement no	Module name	High level requirement	Low level requirement	Test case name

Example of RTM template

Let us one sample of RTM template for better understanding:

	A	B	C	D	E
1	RTM Template				
2	Requirement number	Module number	High level requirement	Low level requirement	Test case name
3		2 Loan	2.1 Personal loan	2.1.1--> personal loan for private employee 2.1.2--> personal loan for government employee 2.1.3--> personal loan for jobless people	beta-2.0-personal loan
4			2.2 Car loan	2.2.1--> car loan for private employee	
5			2.3 Home loan		
6					
7					
8					
9					
10					
11					

Goals of Traceability Matrix

- It helps in tracing the documents that are developed during various phases of SDLC.
- It ensures that the software completely meets the customer's requirements.
- It helps in detecting the root cause of any bug.

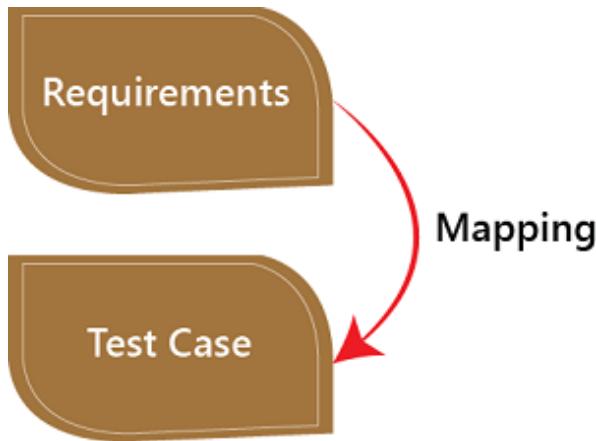
Types of Traceability Test Matrix

The traceability matrix can be classified into three different types which are as follows:

- Forward traceability
- Backward or reverse traceability
- Bi-directional traceability

Forward traceability

The forward traceability test matrix is used to ensure that every business's needs or requirements are executed correctly in the application and also tested rigorously. The main objective of this is to verify whether the product developments are going in the right direction. In this, the requirements are mapped into the forward direction to the test cases.



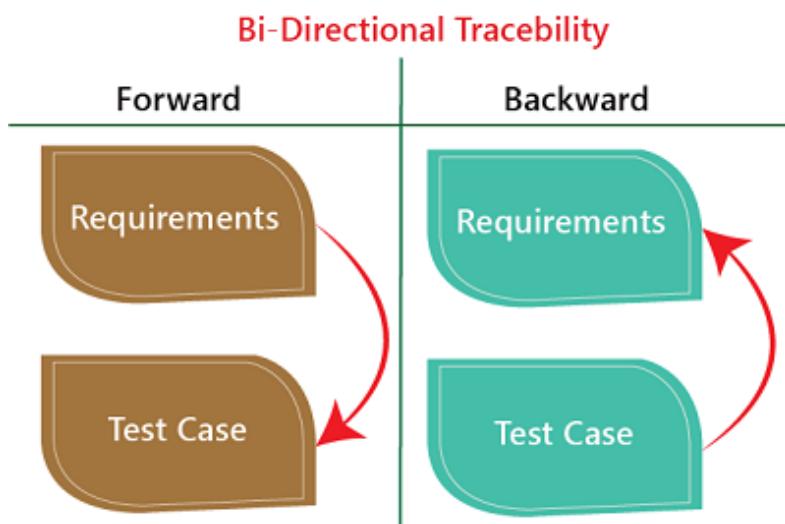
Backward or reverse traceability

The reverse or backward traceability is used to check that we are not increasing the space of the product by enhancing the design elements, code, test other things which are not mentioned in the business needs. And the main objective of this that the existing project remains in the correct direction. In this, the requirements are mapped into the backward direction to the test cases.



Bi-directional traceability

It is a combination of forwarding and backward traceability matrix, which is used to make sure that all the business needs are executed in the test cases. It also evaluates the modification in the requirement which is occurring due to the bugs in the application.



Advantage of RTM

Following are the benefits of requirement traceability matrix:

- With the help of the RTM document, we can display the complete test execution and bugs status based on requirements.
- It is used to show the missing requirements or conflicts in documents.
- In this, we can ensure the complete test coverage, which means all the modules are tested.
- It will also consider the efforts of the testing teamwork towards reworking or reconsidering on the test cases.

Bug in Software Testing

In this chapter, we will learn about defect/bug in software testing and why it occurs, basic terminology of a defect, and bug tracking tool.

What is a bug in software testing?

The Bug is the informal name of defects, which means that software or application is not working as per the requirement.

In [software testing](#), a software bug can also be issue, error, fault, or failure. The bug occurred when developers made any mistake or error while developing the product.



Defect/Bug

While testing the application or executing the test cases, the test engineer may not get the expected result as per the requirement. And the bug had various names in different companies such as error, issues, problem, fault, and mistake, etc.

Basic terminology of defect

Let see the different terminology of defect:

- **Defect**
- **Bug**
- **Error**
- **Issue**
- **Mistakev**
- **Failurev**

Terms	Description	Raised by
Defect	When the application is not working as per the requirement.	Test Engineer
Bug	Informal name of defect	Test Engineer
Error	Problem in code leads to the errors.	Developer, Automation Test Engineer

Issue	When the application is not meeting the business requirement.	Customer
Mistake	Problem in the document is known as a mistake.	--
Failure	Lots of defect leads to failure of the software.	--

Why defect/bug occur?

In software testing, the bug can occur for the following reasons:

- Wrong coding
- Missing coding
- Extra coding

Wrong coding

Wrong coding means improper implementation.

For example: Suppose if we take the Gmail application where we click on the "**Inbox**" link, and it navigates to the "**Draft**" page, this is happening because of the wrong coding which is done by the developer, that's why it is a bug.

Missing coding

Here, missing coding means that the developer may not have developed the code only for that particular feature.

For example: if we take the above example and open the inbox link, we see that it is not there only, which means the feature is not developed only.

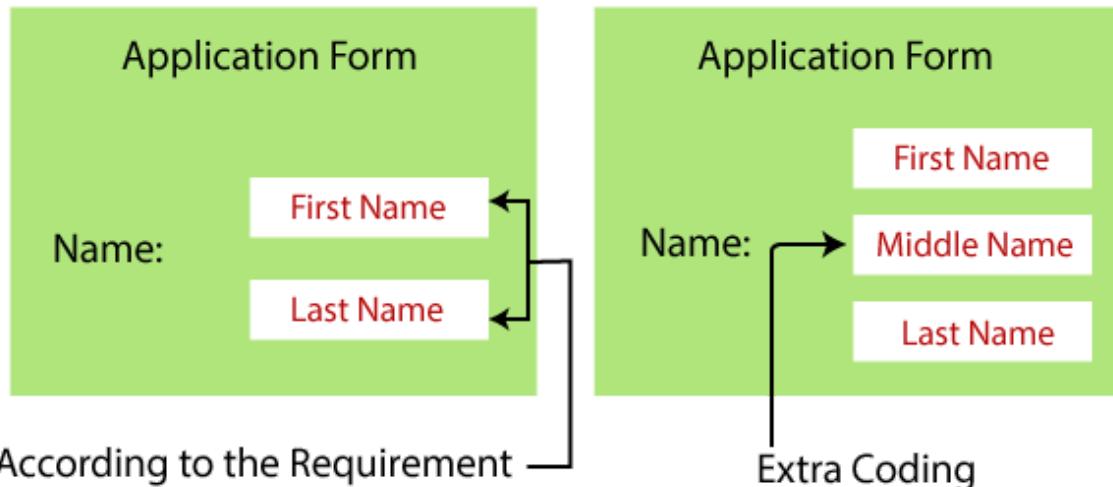
Extra coding

Here, extra coding means that the developers develop the extra features, which are not required according to the client's requirements.

For example:

Suppose we have one application form wherein the **Name field**, the **First name**, and **Last name** textbox are needed to develop according to the client's requirement.

But, the developers also develop the "**Middle name**" textbox, which is not needed according to the client's requirements as we can see in the below image:



If we develop an extra feature that is not needed in the requirement, it leads to unnecessary extra effort. And it might also happen that adding up the extra feature affects the other elements too.

Bug tracking tool

We have various types of bug tracking tools available in software testing that helps us to track the bug, which is related to the software or the application.

Some of the most commonly used bug tracking tools are as follows:

- **Jira**
- **Bugzilla**
- **Redmine**
- **Mantis**
- **Backlog**

Jira

Jira is one of the most important bug tracking tools. Jira is an open-source tool that is used for bug tracking, project management, and issue tracking in **manual testing**.

Jira includes different features like reporting, recording, and workflow. In Jira, we can track all kinds of bugs and issues, which are related to the software and generated by the test engineer.

To get the complete details about Jira tool, refer to the below link:

<https://www.javatpoint.com/jira-tutorial>

Bugzilla

Bugzilla is another important bug tracking tool, which is most widely used by many organizations to track the bugs.

Bugzilla is an open-source tool, which is used to help the customer, and the client to maintain the track of the bugs.

It is also used as a test management tool because, in this, we can easily link other test case management tools such as ALM, quality Centre, etc.

Bugzilla supports various operating systems such as Windows, Linux, and Mac.

Bugzilla has some features which help us to report the bug easily:

- A bug can be listed in multiple formats
- Email notification controlled by user preferences.
- Advanced searching capabilities
- Excellent security
- Time tracking

Redmine

It is an open-source tool which is used to track the issues and web-based project management tool. Redmine tool is written in **Ruby** programming language and also compatible with multiple databases like MySQL, Microsoft SQL, and SQLite.

While using the Redmine tool, users can also manage the various project and related subprojects.

Some of the common characteristics of Redmine tools are as follows:

- Flexible role-based access control
- Time tracking functionality
- A flexible issue tracking system
- Feeds and email notification

- Multiple languages support (Albanian, Arabic, Dutch, English, Danish and so on)

MantisBT

MantisBT stands for **Mantis Bug Tracker**. It is a web-based bug tracking system, and it is also an open-source tool.

MantisBT is used to follow the software defects. It is executed in the **PHP** programming language.

Some of the common features of MantisBT are as follows:

- Full-text search
- Audit trails of changes made to issues
- Revision control system integration
- Revision control of text fields and notes
- Notifications
- Plug-ins
- Graphing of relationships between issues

Backlog

The backlog is widely used to manage the IT projects and track the bugs. It is mainly built for the development team for reporting the bugs with the complete details of the issues, comments. Updates and change of status. It is a project management software.

Features of backlog tool are as follows:

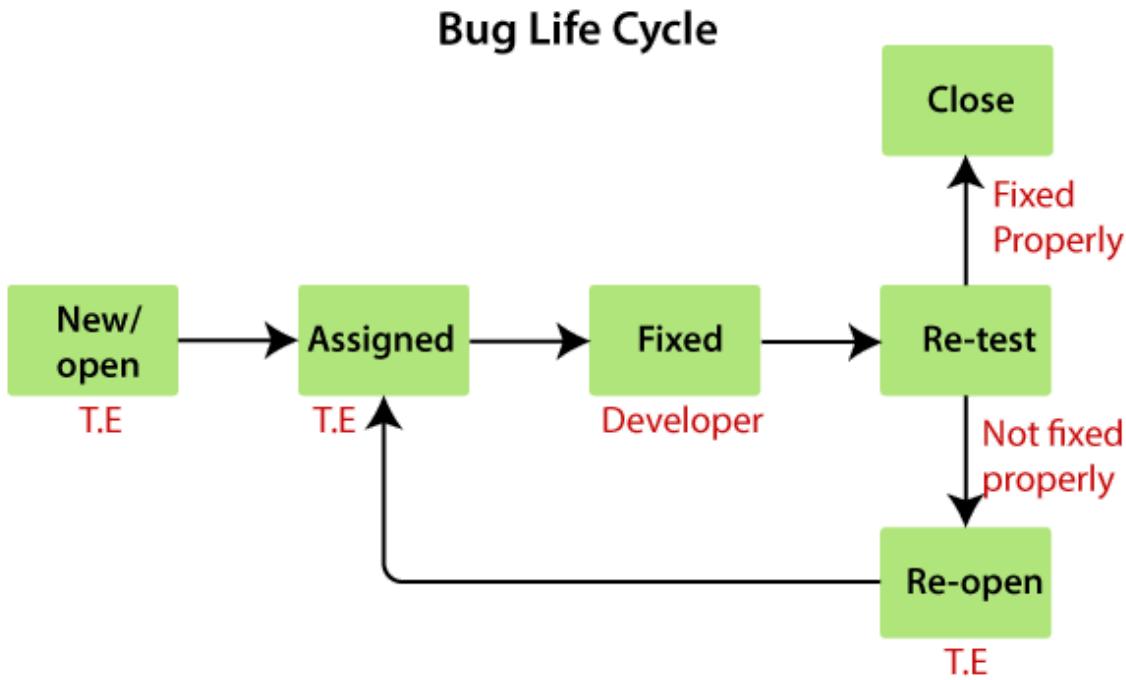
- Gantt and burn down charts
- It supports Git and SVN repositories
- IP access control
- Support Native iOS and Android apps

Bug Life cycle

In this section, we will learn about the bug life cycle and the different status of bugs and bug report template.

Here, we will talk about the complete life cycle of a bug from the stage it was **found, fixed, re-test, and close**.

We have some different status of bugs like **new/open, assigned, fix, re-open, and closed**.



As soon as the test engineer finds the bug, status is given as New, which indicates that a bug is just found.

This new bug needs to be reported to the concerned Developer by changing the status as **Assigned** so that the responsible person should take care of the bug.

Then the Developer first go through the bug, which means that the Developers read all the navigation steps to decide whether it is a valid bug or not.

Based on this, if the bug is valid, the Developer starts reproducing the bug on the application, once the bug is successfully reproduced, the Developer will analyze the code and does the necessary changes, and change the status as **Fixed**.

Once the code changes are done, and the bug is fixed, the test engineer re-test the bug, which means that the test engineer performs the same action once again, which is mentioned in the bug report, and changes the status accordingly:

Close, if the bug fixes properly, and functionally working according to the requirement.

OR

Re-open, if the bug still exists or not working properly as per the requirement, then the bug sends it back to the Developer once again.

This process is going on continuously until all the bugs are fixed and closed.

Note1: The test engineer cannot tell the bug orally to the Developer because of the following reasons:

- Developers might ignore the bug
- Developer misunderstood the bug
- Forget the bug
- The bug may not be found in the exact location

Whom to assign the bug

The bug can be assigned to the following:

- Developers
- Developers lead
- Test lead

Developers: If we know who has developed that particular module.

Developer lead: If we don't know the Developer who has developed the particular module.

Test lead: When we don't have any interaction with the development team.

When the bug is **fixed and closed** or if it is having any impact on the other module, then we go for a new bug report.

OR

When the status of the bug is **Re-open (not fixed)** and affecting another module, then we have to prepare the new bug report.

Note2:

- Whenever we find a bug and Developers fix it, we have to check one round of impact area.
- If the old bug is fixed correctly, change the status to Close.
- And if we find a bug in the impact area, then reported as a new bug.
- If the old bug is not fixed properly, then change the status to Re-open.
- Or, if we find a bug impact area, then change the status to New or reported as a new bug.

Another status of the bug

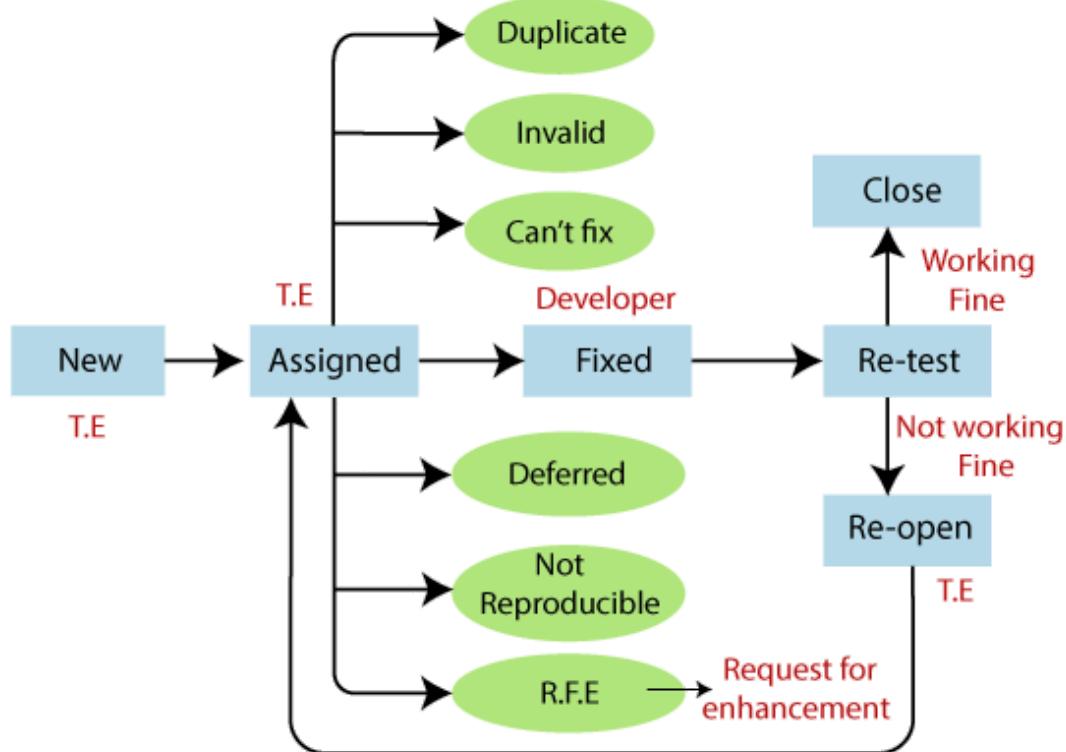
Once we prepared a bug report and send it to the Developers, the Developer will accept the bug and starts doing the necessary code changes that become the **positive flow** of the bug life cycle.

There may be several conditions where Developers may not do the necessary code changes and depend on the situation, which becomes a **negative flow or status** of the bug life cycle.

Following are the different status of the bug life cycle:

- **Invalid/rejected**
- **Duplicate**
- **Postpone/deferred**
- **Can't fix**
- **Not reproducible**
- **RFE (Request for Enhancement)**

Final Diagram of Bug life cycle

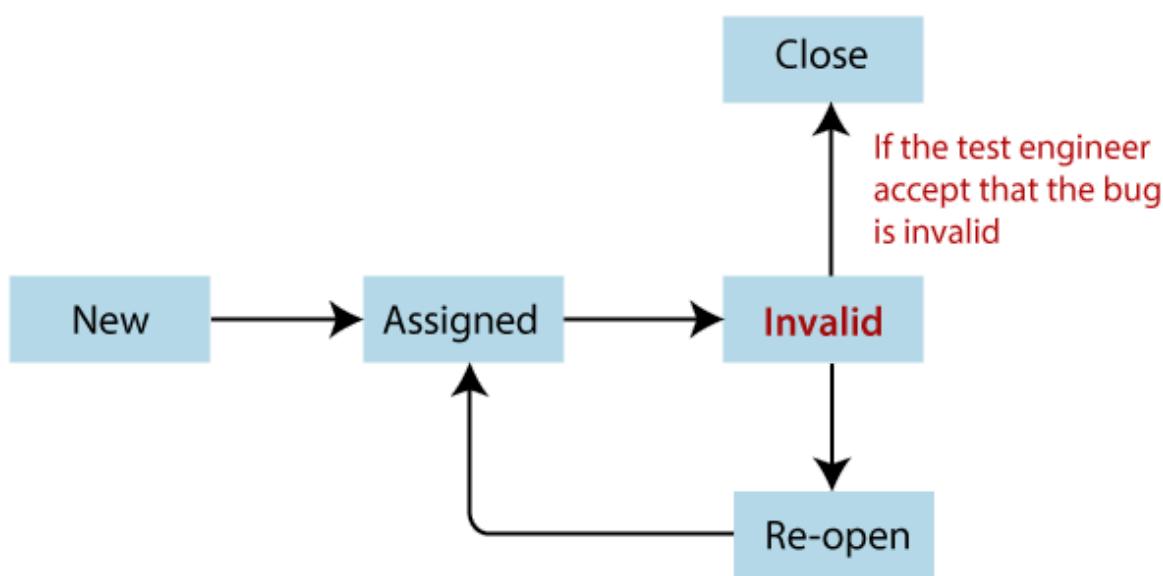


Invalid / rejected

When the Test Engineer wrote an incorrect Bug Report because of misunderstanding the requirements, then the Developer will not accept the bug, and gave the status as **Invalid** and sent it back. (Sometime Developer can also misunderstand the requirements).

“ Any bug which is not accepted by the developer is known as an invalid bug. ”

Invalid



Reasons for an invalid status of the bug

The invalid status of the bug is happened because of the following reasons:

- **Test Engineer misunderstood the requirements**
- **Developer misunderstood the requirements**

Let's see one example where the test engineer and developer misunderstood the requirements as we can see in the below image:

Bug Report # 1
1. Login Application 2. Compose Mail, Send 3. Click on "Trash" Observation: Mail Not found in "Trash" Status:- Assigned (invalid)

Bug Report # 2
1. Login Application 2. Compose Mail, Send 3. Click on "Send Item" Observation: Mail Not found in "Send Item" Status:- Assigned (invalid)

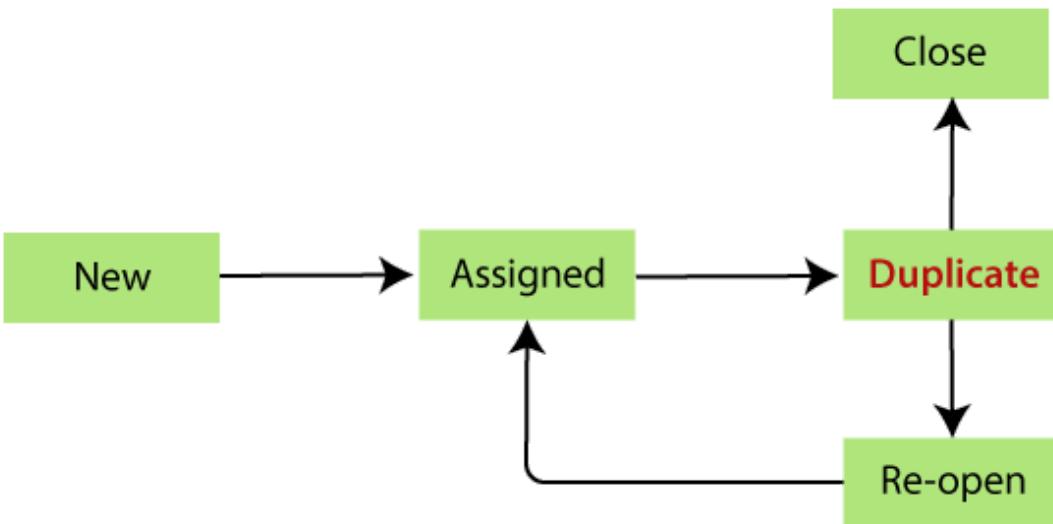
TE (Test Engineer)
misunderstood the
requirement

Developer
misunderstood the
requirement

Duplicate

When the same bug has been reported multiple times by the different test engineers are known as a **duplicate** bug.

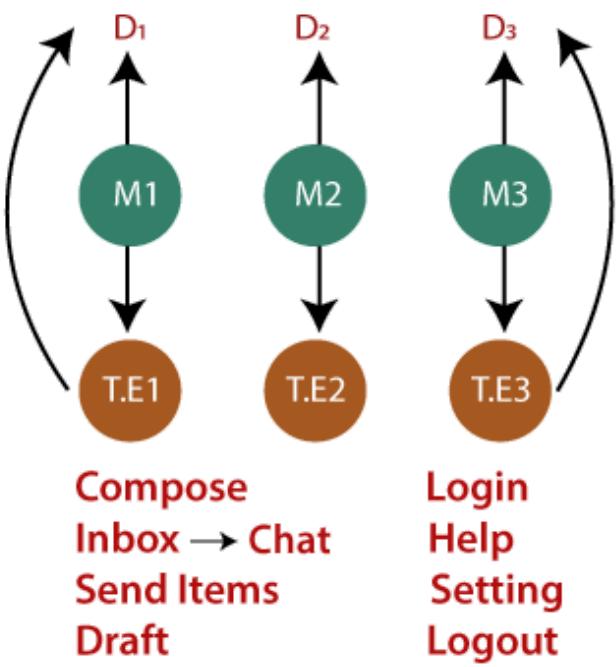
Duplicate



Reasons for the duplicate status of the bug

Following are the reasons for the duplicate status:

- **Common features: For example:** Suppose we have test engineer P and Q which are testing the software, the test engineer P and Q will test their features like **login the application**. Here, the test engineer P enters the valid username and password, and click on the login button. Once P click on the login button, it opens a blank page, which means that it is a bug. After that, P prepares a bug report for the particular bug and sends it to the developer. Then the test engineer Q also login the application and got the same bugs. Q also prepare a bug report and send it to the developer. Once the developer got both test engineers bug report, he/she sends back the bug report to the Q and say it is duplicate.
- **Dependent Modules** As we can see in the below image, that the test engineer wants to **compose a mail**, so first, the test engineer needs to **login**, then only he/she can able to compose a mail. If the bug is found in the **login module**, the test engineer cannot do further process because the composing module is dependent on the login module.



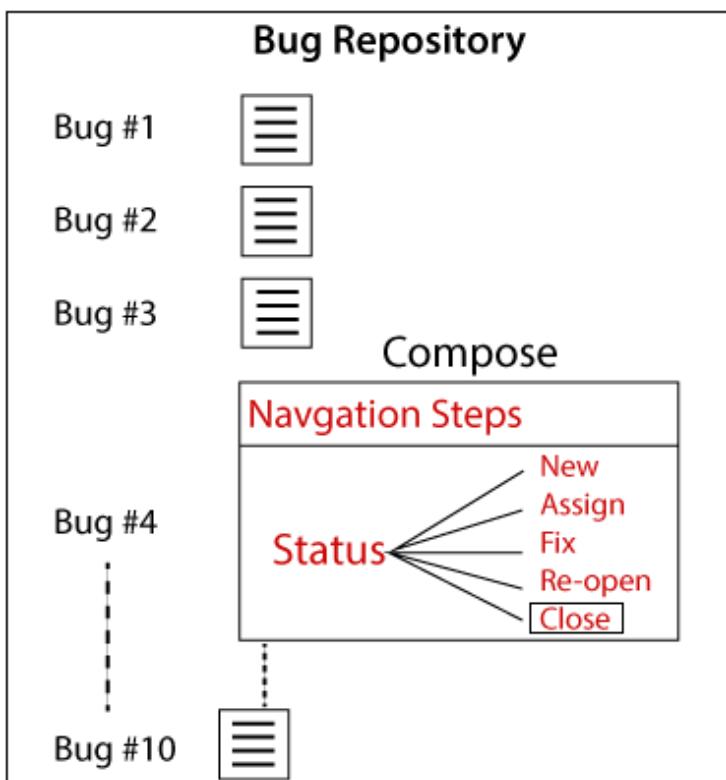
- **To avoid the duplicate bug** If the Developer got the duplicate bug, then he/she will go to the bug repository and search for the bug and also check whether the bug exist or not. If the same **bug exist**, then no need to log the same bug in the report again. **Or** If the bug **does not exist**, then log a bug and store in the bug repository, and send to Developers and Test Engineers adding them in [CC].

Note1:

- Generally, we don't search for each bug in the repository to check the duplicity.
- To save time, we only search that bug that has the common feature and dependent features.

Note2: Whenever we are comparing two bug reports to find out if it is duplicate or not, we should always look at two things, which are as follows:

- The navigation steps should be the same.
- Apart from the close status, any other status should exist, we should not log a bug, and else it will become a duplicate bug as we can see in the below image:



Not Reproducible

The Developer accepts the bug, but not able to Reproduce due to some reasons.

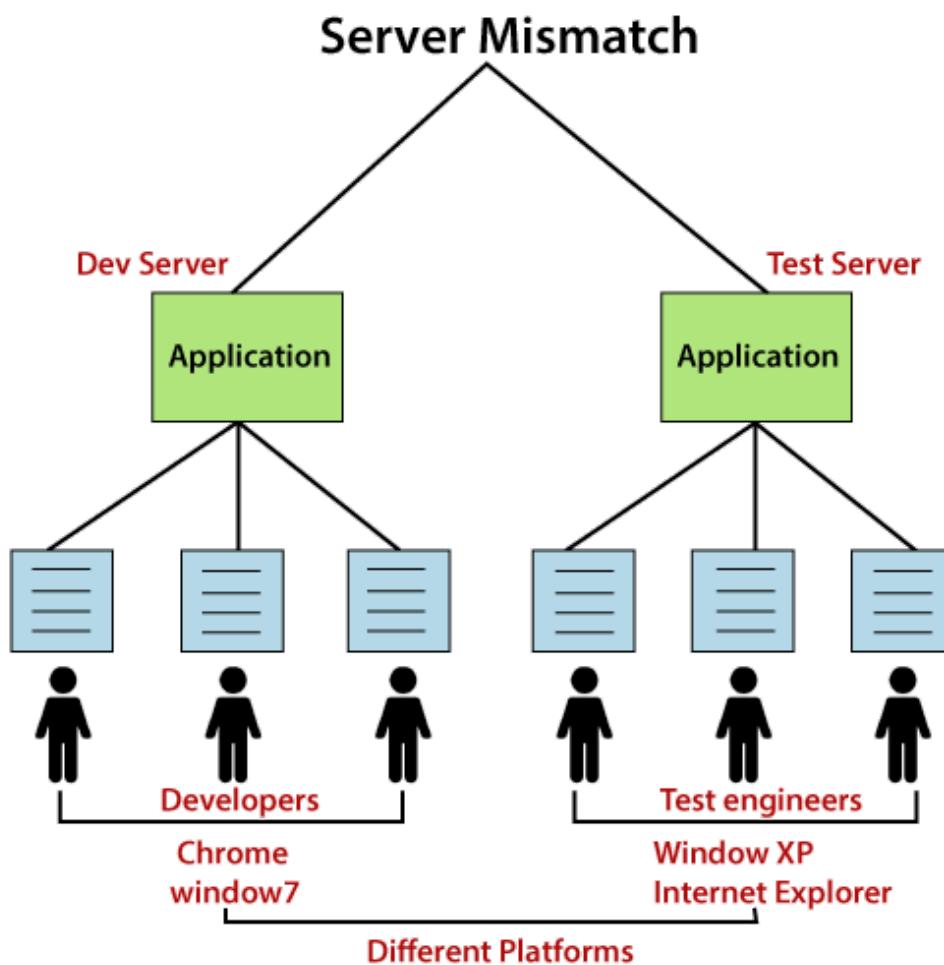
“ These are the bug where the developer is not able to find it, after going through the navigation step given by the test engineer in the bug report. ”

Reasons for the not reproducible status of the bug

Reasons for the not reproducible status of the bug are as follows:

- **Incomplete bug report** The Test engineer did not mention the complete navigation steps in the report.
- **Environment mismatch** Environment mismatch can be described in two ways:
 - **Server mismatch**
 - **Platform mismatch**

Server mismatch: Test Engineer is using a different server (**Test Server**), and the Developer is using the different server (**Development Server**) for reproducing the bug as we can see in the below image:



Platform mismatch: Test engineer using the different Platform (**window 7 and Google Chrome browser**), and the Developer using the different Platform (**window XP and internet explorer**) as well.

- **Data mismatch** Different Values used by test engineer while testing & Developer uses different values. **For example:** The requirement is given for admin and user.

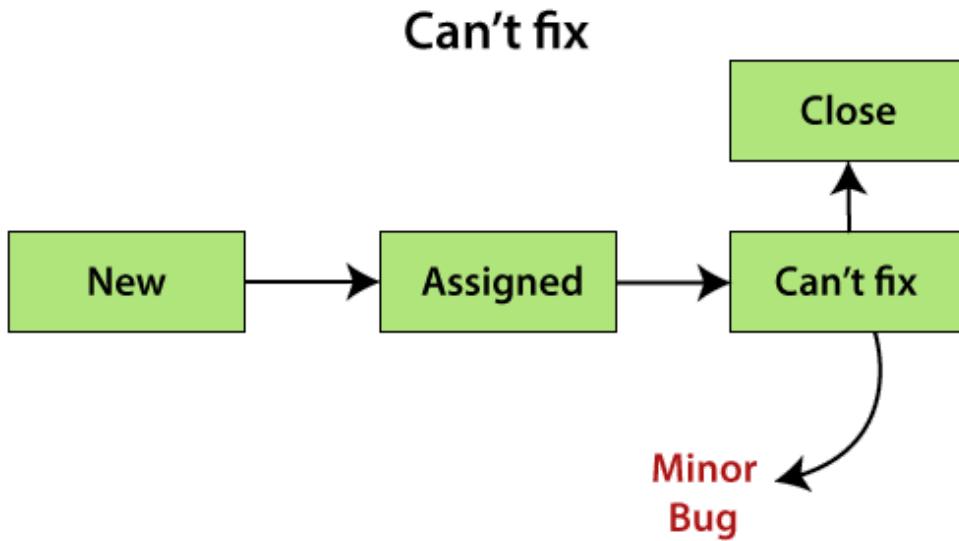
Test engineer(user) using the below requirements:	the Developer (admin) using the below requirements:
User name → abc Password → 123	User name → aaa Password → 111

I.e., both are using a different value for the same login module.

- **Build mismatch** The test engineer will find the bug in one Build, and the Developer is reproducing the same bug in another build. The bug may be automatically fixed while fixing another bug.
- **Inconsistent bug** The Bug is found at some time, and sometime it won't happen. **Solution for inconsistent bug:** As soon as we find the bug, first, **take the screenshot**, then developer will **re-confirm** the bug and fix it if exists.

Can't fix

When Developer accepting the bug and also able to reproduce, but can't do the necessary code changes due to some constraints.



Reasons for the can't fix status of the bug

Following are the constraints or reasons for the can't fix bug:

- **No technology support:** The programming language we used itself not having the capability to solve the problem.
- **The Bug is in the core of code (framework):** If the bug is **minor** (not important and does not affect the application), the development lead says it can be fixed in the next release. Or if the bug is **critical** (regularly used and important for the business) and development lead cannot reject the bug.
- **The cost of fixing a bug is more than keeping it.**

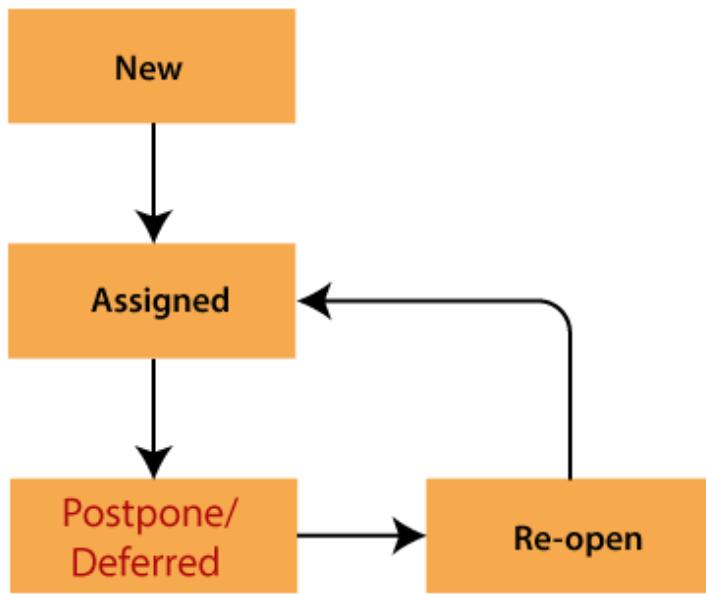
Note:

- If any bug is minor, but the Developer can't fix it, which means that the Developer can fix, but the bug is affecting the existing technology because it was present in the core of the code.
- Each can't fix bugs are the minor bug.

Deferred / postponed

The deferred/postpone is a status in which the bugs are postponed to the future release due to time constraints.

Postpone/Defferred

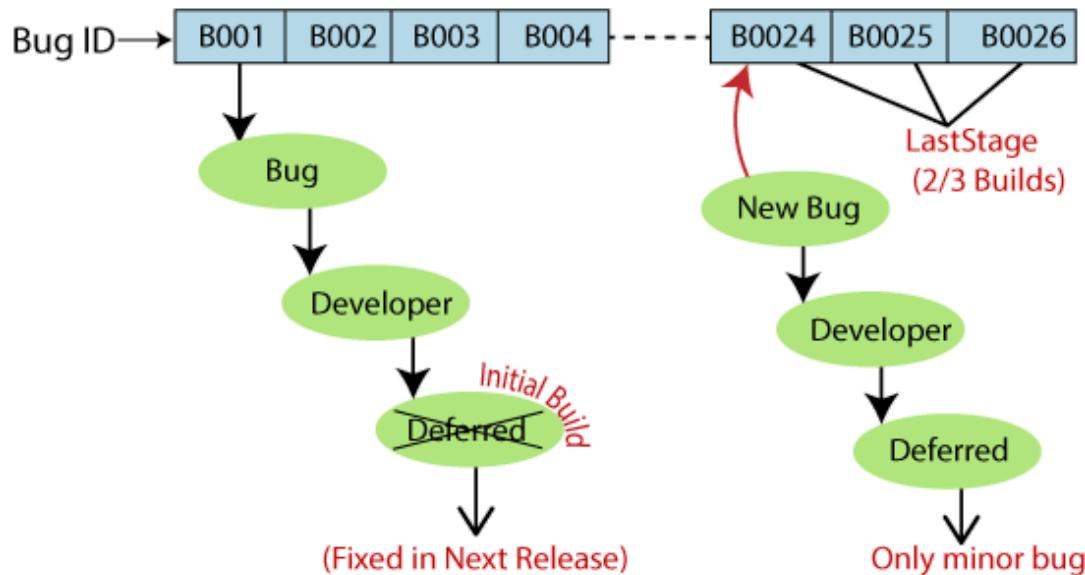


The deferred status of the bug was not fixed in the initial build because of the time constraints.

As we can see in the below image:

The **Bug ID-B001** bug is found at the initial build, but it will not be fixed in the same build, it will postpone, and **fixed in the next release**.

And **Bug ID- B0024, B0025, and B0026** are those bugs, which are found in the last stage of the build, **and they will be fixed because these bugs are the minor bugs**.

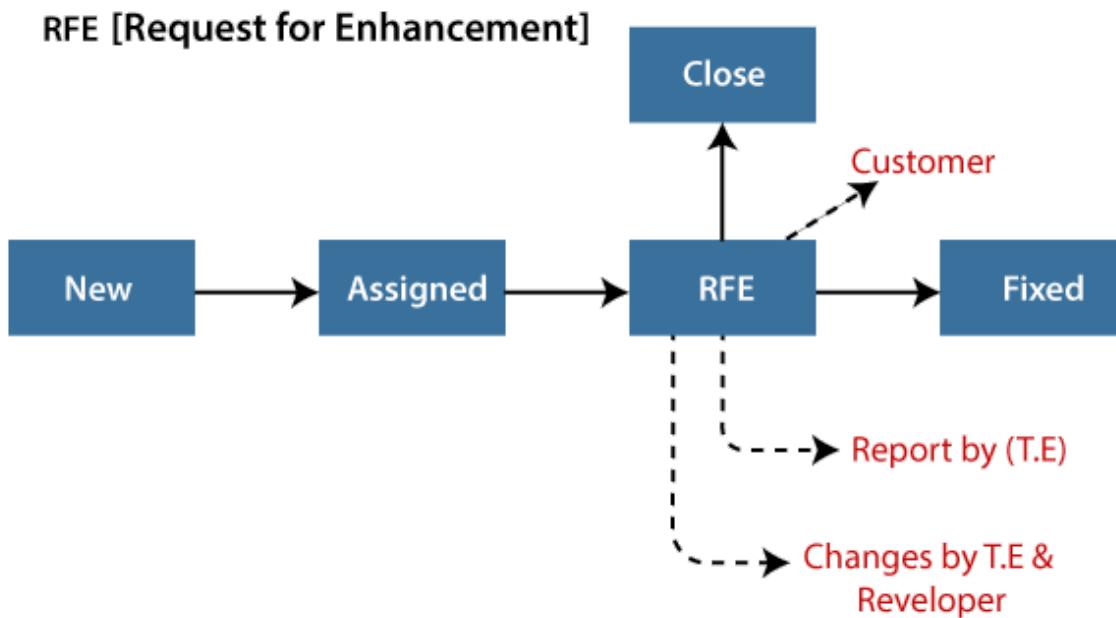


Note:

- All minor bugs can't be deferred, but all deferred bugs are minor bugs.
- Whenever there is no future release, then the postpone bug will be fixed at the maintenance stage only.

RFE (Request for Enhancement)

These are the suggestions given by the test engineer towards the enhancement of the application in the form of a bug report. The RFE stands for **Request for Enhancement**.

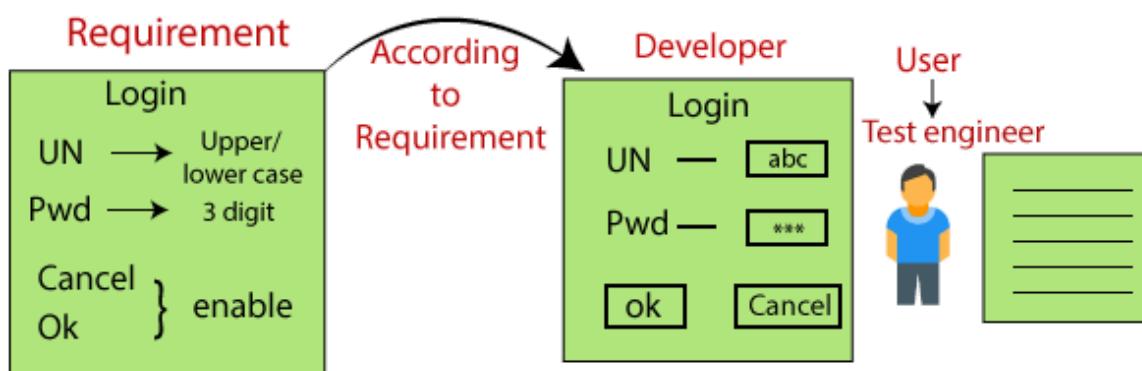


As we can see in the below example image that the test engineer thinks that the look and feel of the application or software are not good because the test engineer is testing the application as an end-user, and he/she will change the status as **RFE**.

And if the customer says **Yes**, then the status should be **Fix**.

Or

If the customer says **no**, then the status should be **Close**.



Bug Report Template (excel)

The bug report template is as follows:

Bug Report Template	
Bug ID	
Module	
Requirements	
Test Case Name	
Release	
Version	
Status	
Reporter	
Date	
Assign To	
CC	
Severity	
priority	
Server	
Platform	
Build No.	
Test Data	
Attachment	
Brief Description	
Navigation Steps	
Observation	
Expected Result	
Actual Result	
Additional Comments	

Let see one example of the bug report:

Bug ID	Boo12
Module	Login
Requirement	1
Test case name	Gmail_login_compose _mail
Reporter	Test engineer name
Release	delta
Version	3.0
Status	assigned
Date	23-02-2020
Assign to	YY developer
CC	Test lead, developer
Severity	critical
Priority	P2
Platform	Window XP, internet explorer7.0
Build no.	B02
Test data	Username=xyz, password= 123
Brief description	

Navigation steps	<ul style="list-style-type: none"> ◦ Login Gmail application ◦ Compose mail and confirmation message should be displayed
observation Mail	not found in inbox
Expected result Mail	should also be in the inbox.
Actual result	Mail not found in inbox
Additional comments	-----

Here, we are describing some important attributes of the bug report.

Bug ID: it is a unique number given to the bug.

Test case name: When we find a bug, we send a bug report, not the test case to the concerned developer. It is used as a reference for the test engineer.

Severity: It is the impact of a bug on the application. It can be a blocker, critical, major, and minor.

Priority: In this, we have to decide which bug has to be fixed first. It could be P1/P2/P3/P4, urgent, high, medium, and low.

Status: The different status of the bug which could be assigned, invalid, duplicate, deferred, and so on.

Reporter: In this, we will mention the name of the person who found the bug. It could be the test engineer, and sometime it may be a developer, business analyst, customer, etc.

Date: It provides the date when the bug is found.

Release/Build Version: It provides the release number in which the bug occurs, and also the build version of the application.

Platform: Mention the platform details, where we exactly find the bug.

Description: In this, we will explain the navigation steps, expected and actual results of the particular bug.

Attachments: Attach the screenshots of the bug, which we captured because it helps the developers to see the bug.

The Drawback of a manual bug report

Following are the disadvantages of manual bug report:

- **Time consuming** While searching every bug in the bug report, it will be time taken process.
- **Possibility of human error** A bug may be repeated, wrong data mentioned in the bug report, and miss something to add on the bug report.
- **No security** Anyone can change it or delete it.
- **Tedious process**
- **No centralized repository**

Severity and Priority in testing

In this section, we will learn about the severity and the Priority of a bug in software testing

Severity

The impact of the bug on the application is known as severity. It can be a **blocker, critical, major, and minor** for the bug.

Blocker: if the severity of a bug is a blocker, which means we cannot proceed to the next module, and unnecessarily test engineer sits ideal.

There are two types of **blocker** bug, which are as follows:

A major feature is not working: Login to HDFC, amount transfer is not working

The major flow is not working: Login and signup itself not working in HDFC application.

Critical: if it is critical, that means the main functionality is not working, and the test engineer cannot continue testing.

Major: if it is major, which means that the supporting components and modules are not working fine, but test engineer can continue the testing.

Minor: if the severity of a bug is major, which means that all the U.I problems are not working fine, but testing can be processed without interruption.

Priority

Priority is important for fixing the bug or which bug to be fixed first or how soon the bug should be fixed.

It can be **urgent, high, medium, and low**.

High: it is a major impact on the customer application, and it has to be fixed first.

Medium: In this, the problem should be fixed before the release of the current version in development.

Low: The flow should be fixed if there is time, but it can be deferred with the next release.

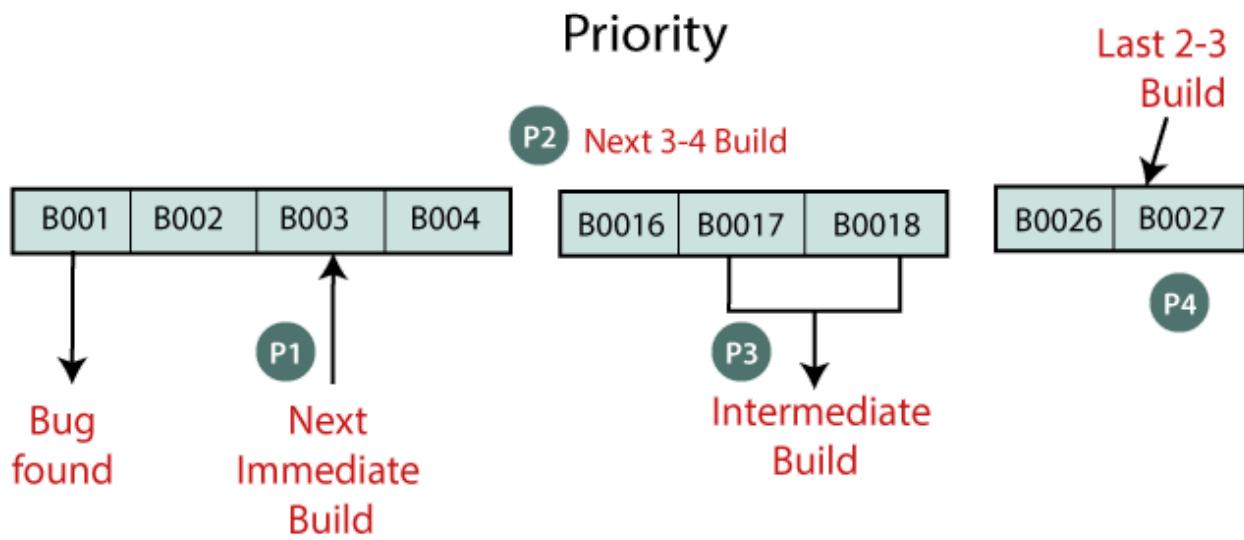
Note: The test engineer decides the severity and Priority, and the developer can also change the severity with a proper reason and comments on the bug reports. A developer cannot change the Priority, because if the developer changes the Priority, he/she may fix the easy bug's first.

Example of severity and Priority

Example 1

Suppose we have to send the priority means which bug needs to fix first according to the requirement of the client.

- When the bug is just found, it will be fixed in the next immediate build, and give the Priority as P1 or urgent.
- If the Priority of the bug is P2 or high, it will be fixed in the next 3-4 builds.
- When the Priority of the bug is P3/medium, then it will be fixed in the intermediate build of the application.
- And at last, if the Priority is P4/low, it will be fixed in the last 2-3 build of the software as we can see in the below image:



Example 2

If we take the case of a login module, then the severity and Priority could depend on the application like as we can see in the below image:

Severity	Requirement	Priority
Critical	Login	[P1]
Critical	Compose	[P1]
Critical	Inbox	[P1]
Major	Send Item	[P2]
Major	Trash	[P3]
Minor	Help	[P3]
Minor	Logout	[P4]

What is the test environment?

The test environment is a collection of hardware and software, which helps us to execute the test cases.

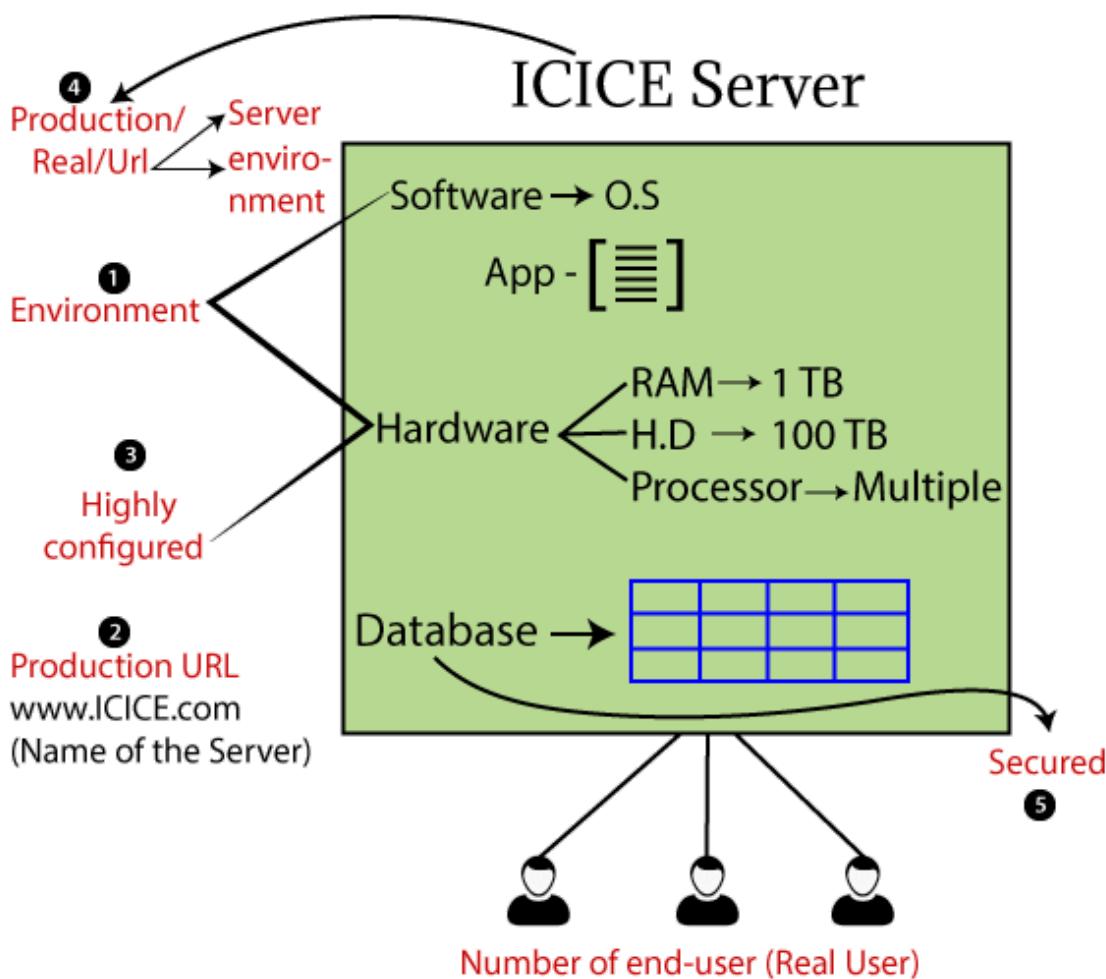
Once we get the requirement from the customer, the developer starts developing the code.

When the coding phase is completed, the application should be installed in the test server. And that application can be accessible with the help of the URL, and the URL could be like this:

HTTP://COMPUTER NAME/PROJECT/COMPANY NAME

Suppose we have the ICICE Server, which has been configured with software, hardware, database, and the application.

- The test environment has a software configuration (operating systems), hardware configuration (RAM, Hard Disk, and Processor), and the test console, which help us to execute the test cases.
- The environment setting is an important part of the testing process because if it is not set correctly, the testing team cannot start the testing process, and the application might be collapse.
- And the Production URL could be **the name of a server** such as , and **the environment**.
- The database is secured; that's why we can use it to create, modify, and delete the data.
- This server can be used by n-number of the end-users (real user).



Software server

Here, we have four types of servers, which are as follows:

- **Production server**
- **Development server**
- **QA Main server**
- **Staging server**

Production Server

The server, which consists of software, hardware, and the application development environment, is known as the **production server**. It is a core server in which the users access any web application or website.

The production server configuration is similar to the staging server, and the application should be debugged and tested on the staging server before dumped into the production server.

Features of the production server

- The end-user or real user use this server.
- Security is given to the production server since it contains real user data.
- Highly configured system since it is used by n-number of users.
- The production URL accesses the production server.
- Once the application is developed/tested/stable, then it is deployed into the production environment.

Development server

Generally, it is accessed by developers for writing new lines of code as well as bug fixes. The dev server gives a run-time environment where the program developed and debugged.

QA Main server

The test engineers access it for conducting all the types of testing and also obtain by developers for reproducing the bugs. In this server, we will perform the unit and integration testing.

Test (Staging) server

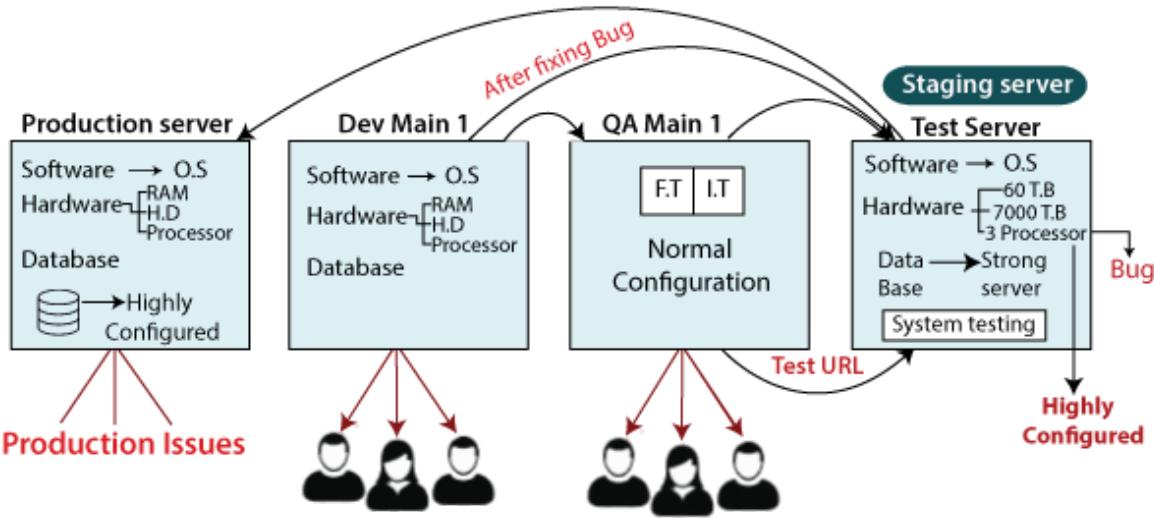
We take another server to check the compatibility of the server because the production server is highly configured, so we need a similar server where we can perform the system testing.

Checking the application compatibility **bugs** is known as a staging server.

This server helps us to identify the software performance, behavior, bugs, and other issues before the application is deployed into the production server.

Process of test environment

- Once the requirement is given by the customer and developers finish the coding, we start the initial round of testing on the test server like **functional/unit, integration** testing with having normal configuration.
- Here, all the servers software should be the same, but the hardware and the database are similar to the **production server** and the **staging server**.
- For the **system testing**, we need a highly configured server, which is similar to the production server (similarity to the software and hardware and database) that is called the **Staging server**.
- In the staging server, we are using the **test URL** and perform the **system testing**.
- If we find any bug, we will be handed over to the developers, and they will fix the bug, and it will directly store in the **staging test server**.



- In a company, we will use only two environments, which are as follows:
 - Developer environment
 - Testing environment
- In the testing process, we will move to the database from **QA Main 1** to the **Staging server** to save time, instead of creating a new one using the already existing dev database.
- And then move the application from the test server (staging server) to the Production server.

Note: Production Issues: These issues can happen in the production site in real-time where end users are using the application.

Why test environment similar to the production server?

Since, if we move the application from low configuration to the production, the user may find some issues. To avoid this, we do one round off end to end (system) testing on an environment, which is similar to the production environment.

Following are the aspects which shows why we need the test environment similar to the production server:

The software should be similar to the production

- The database server should be similar
- The operating system should be similar
- The web server should be similar
- The application server should be similar

The hardware should be similar to the production

- The hardware configuration should be similar to the production server

For example: if the production server is the ICICE, then the test server should also be the ICICE server.

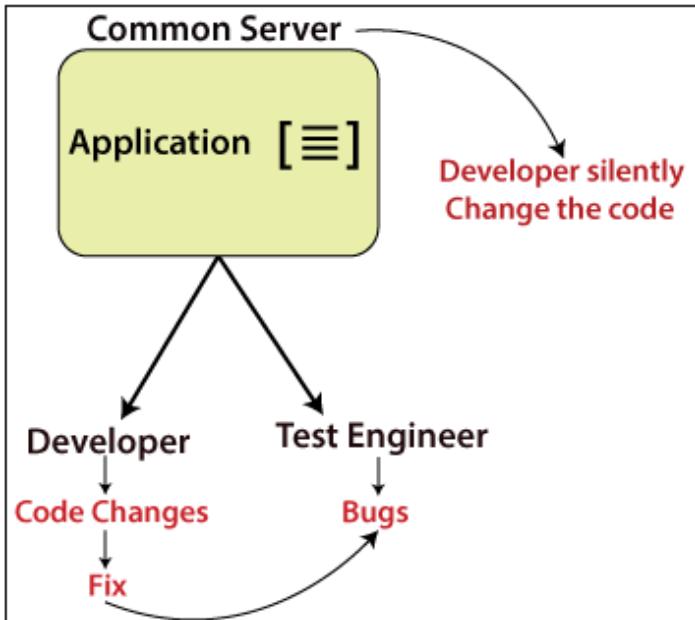
- The configuration and hardware must be similar but different capacities, i.e., the number of CPUs.

The data should be similar to the production

- We should create a data, which is similar to the production.
- In the real-time environment, we may make n-numbers of entries into the database, but while testing, we cannot enter n-numbers of entries manually. So, we write a test script that makes thousands of entries, which can be used for testing.

Note:

- Can we work on the one common server for the development and testing team? No, because there are no continuous tasks, and the developer silently change the code if we have the common server.



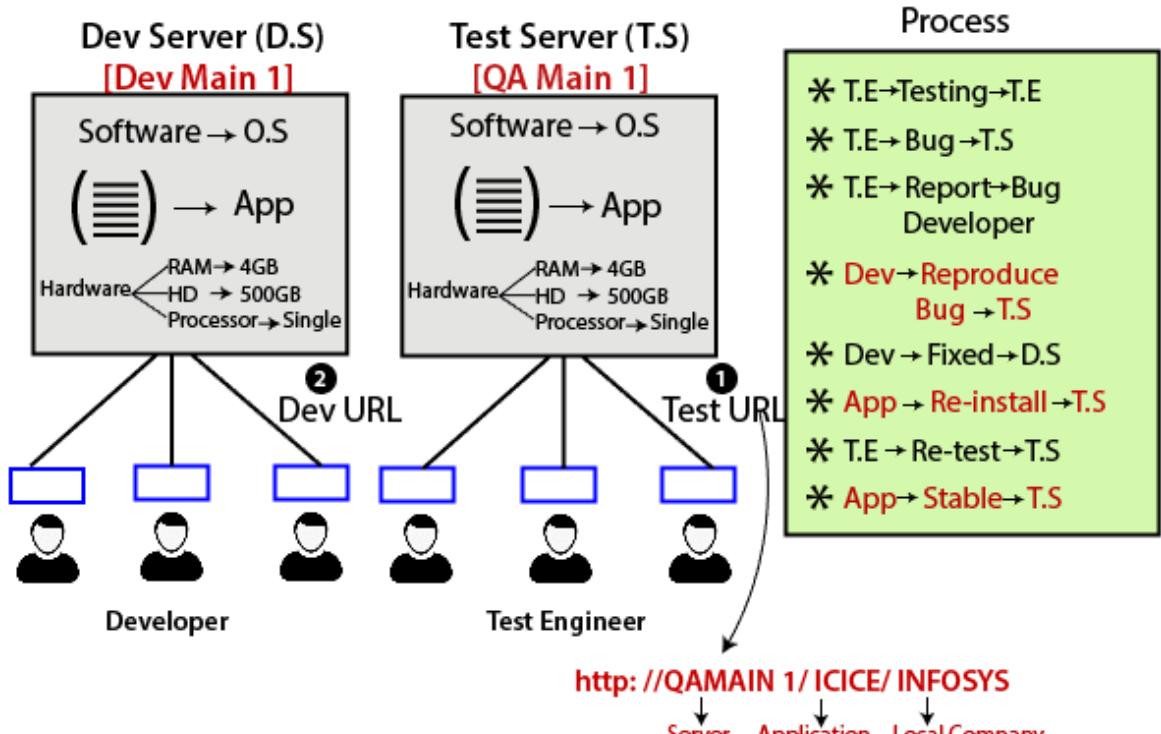
- When the developer reproduce the bug, and in which server, test server or development server?

- Where the bug is found, it can only be reproduced there (test server, founded bug).
- If we do this on the developer server, it is possible that some other developers can change the code, so the good approach is to reproduce the bug only on the test server.
- The Application is always moved from the test server to the production server 569because the developer can change the code at any time.
- But, in the test server, the people can't access the code, as we always transfer the application from the test server to the production server.

- Does the URL only gives access to the application which has to be tested?

- When the customer needs to change or give the new requirement for the application, it will give it to a particular company.
- After getting the requirement, the development team creates one separate server and use the developer URL to run the application.
- Once the application is ready, it will hand over to the test engineer, and the test engineer will test the application using the test URL, which can be used only in a particular company.

- While we are using a URL, we can only give access to the application from the server or the frontend of

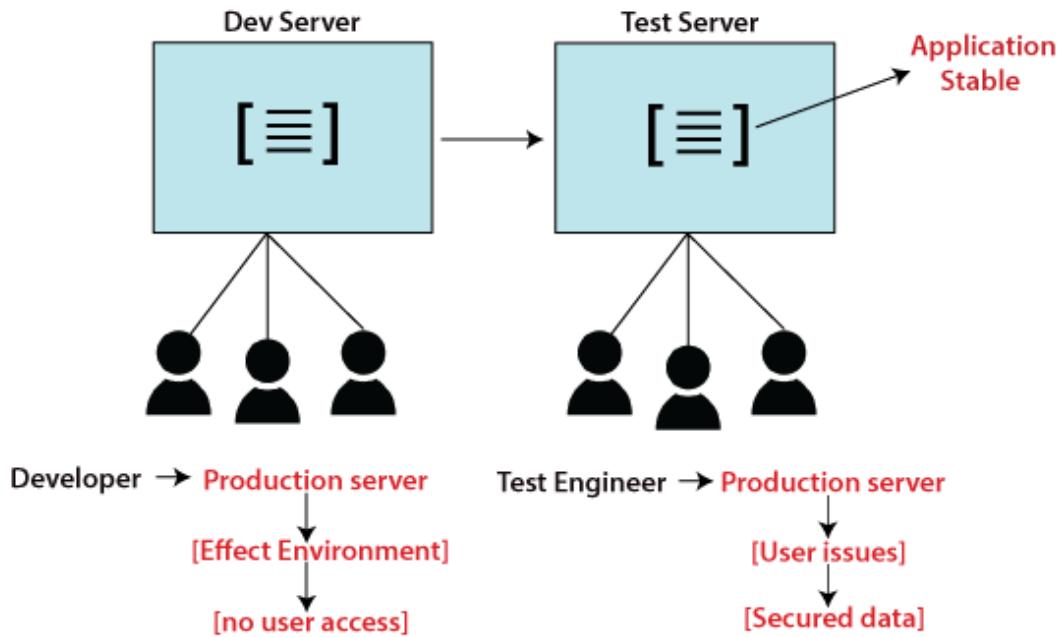


the application.

- The customer wants an application, then he/she goes to Software Company and gives the requirements; the company provides a requirement for the developer to develop the code in the development server.
- After completing the code, the application is installed in the test server so that the test engineer will test the application until the application is stable then it will be deployed into the production server.

- **Can we change the requirements into two servers?**

- The test engineer and developer can't change anything when the application is in the production server because they may have changed the requirement on the dev server and test server.



Defect Management Process

In this section, we are going to understand the working of the **Defect management process**.

Also, see the defect in software testing, **defect management process's objective**, **defect management process**, **advantages and disadvantages of the defect management process**.

But, firstly, we will understand the process of defect management, and we will understand the **defect in software testing**.

Defect in Software Testing

- The bug announced by the **programmer** and inside the code is called a **defect**.
- In other words, we can say that when the application is not working as per the requirement is known as **defects**.
- It is specified as the irregularity from the **actual and expected result** of the application or software.
- The **Defect** is the difference between the actual outcomes and expected outputs.
- The **Test engineer** can identify the defect, and it was fixed by the developer in the development phase of the **software development life cycle**.
- When a test engineer tests a piece of code, he/she comes across differences in expected output to the existing output, which is known as a **defect**. And the substitute of defect can be further known as **issues, bugs, and incidents** in software testing.

What is Defect Management Process?

The **defect management process** is the core of software testing. Once the defects have been identified, the most significant activity for any organization is to manage the flaws, not only for the testing team but also for everyone involved in the software development or project management process.

As we know, **defect prevention** is an effective and efficient way to decrease the number of defects. The defect prevention is a very cost-effective process to fix those defects discovered in the earlier stages of software processes.

The **Defect Management Process** is process where most of the organizations manage the **Defect Discovery**, **Defect Removal**, and then the **Process Improvement**.

As the name recommends, the **Defect Management Process (DMP)** manages defects by purely detecting and resolving or fixing the faults.

It is impossible to make a software 100% error or defect-free, but several defects can be declined by fixing or resolving them.

The defect management process primarily focuses on stopping defects, finding defects in the earlier stages, and moderating the effect of defects.

The Objective of Defect Management Process (DMP)

The main objective of the defect management process is as discussed below:

- The primary objective of DMP is to expose the defects at an early stage of the software development process.
- The execution of the defect management process will help us enhance the process and implementation of software.
- The defect management process reduces the impact or effects of defects on software.
- The Defect management process (DMP) helps us to avoid defects.
- The main goal of the Defect management process is to resolve or fixing the defects.

And for the **different organization or projects** the critical goals of the Defect management process is as follows:

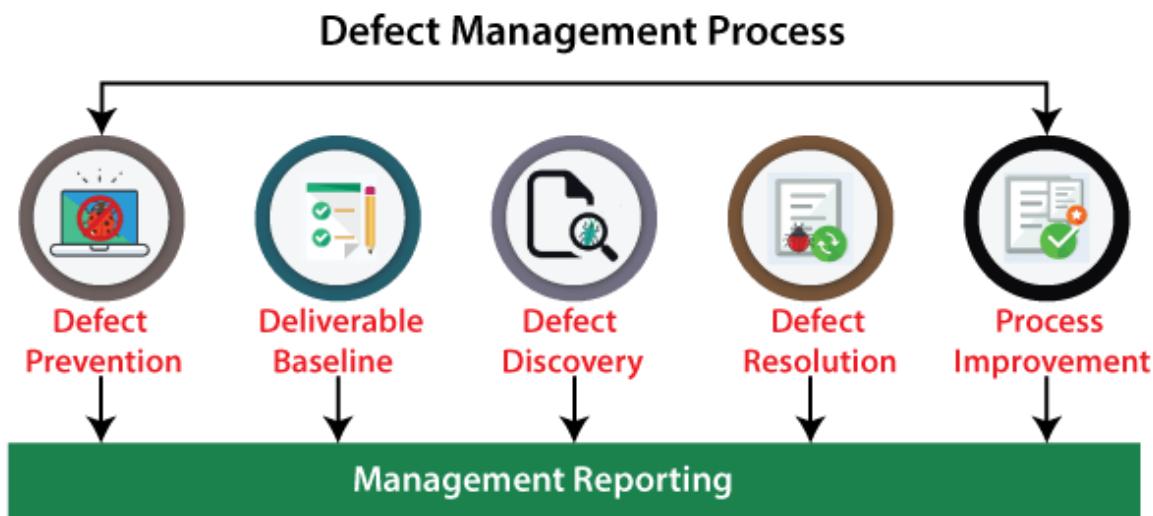
- The defect management process allows us to provide input for status and progress reports about the defect.
- To find the primary cause that how the defect happened and how to handle it.
- To provide input, for information related to the release of the defect.

Various Stages of Defect Management Process

The defect management process includes several stages, which are as follows:

1. **Defect Prevention**
2. **Deliverable Baseline**
3. **Defect Discovery**
4. **Defect Resolution**
5. **Process Improvement**
6. **Management Reporting**

Let's discuss them one by one:



1. Defect Prevention

The first stage of the **defect management process** is **defect prevention**. In this stage, the execution of procedures, methodology, and standard approaches decreases the risk of defects. Defect removal at the initial phase is the best approach in order to reduction its impact.

Because in the initial phase of fixing or resolving defects is less expensive, and the impact can also be diminished.

But for the future phases, identifying faults and then fixing it is an expensive process, and the effects of defect can also be amplified.

The defect prevention stage includes the following significant steps:

- **Estimate Predictable Impact**
- **Minimize expected impact**
- **Identify Critical Risk**



Defect Prevention

01 Estimate Predictable impact

02 Minimize expected impact

03 Identify Critical Risk

Step1: Estimate Predictable Impact

In this step, if the risk is encountered, then we can calculate the estimated financial impact for every critical occasion.

Step2: Minimize expected impact

When all the critical risk has been discovered, we can take the topmost risks that may be dangerous to the system if encountered and try to diminish or eliminate it.

Those risks that cannot be removed will decrease the possibility of existence and its financial impact.

Step3: Identify Critical Risk

In defect prevention, we can quickly identify the system's critical risks that will affect more if they happened throughout the testing or in the future stage.

2. Deliverable Baseline

The second stage of the defect management process is the **Deliverable baseline**. Here, the deliverable defines the **system, documents, or product**.

We can say that the **deliverable is a baseline** as soon as a deliverable reaches its pre-defined milestone.

Note: The Pre-defined milestone describes what the software is supposed to accomplish.

In this stage, the deliverable is carried from one step to another; the system's existing defects also move forward to the next step or a milestone.

In other words, we can say that once a deliverable is baselined, any additional changes are controlled.

3. Defect Discovery

The next stage of the defect management process is **defect discovery**. At the early stage of the defect management process, defect discovery is very significant. And later, it might cause greater damage.

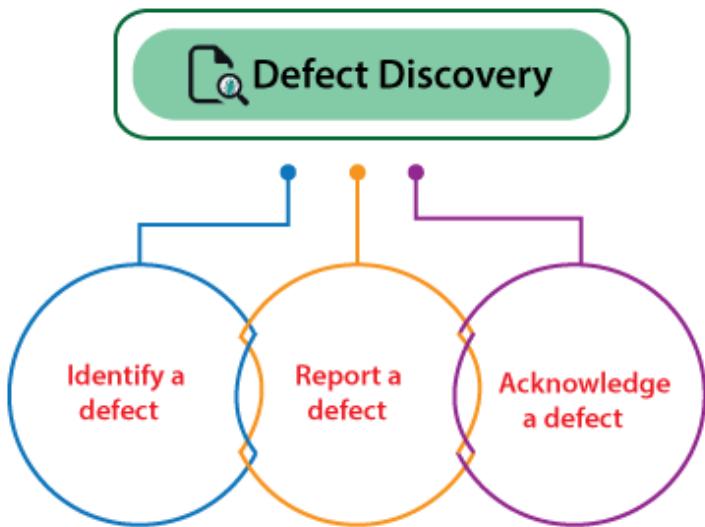
If developers have approved or documented the defect as a valid one, then only a defect is considered as **discovered**.

As we understood that, it is practically impossible to eliminate each defect from the system and make a system defect-free. But we can detect the defects early before they become expensive to the project.

The following phases have been including in the defect discovery stage; let's understand them in details:

- **Identify a defect**

- **Report a defect**
- **Acknowledge Defect**



Phase1: Identify a Defect

In the first phase of defect discovery where we need to find the defects before becoming a critical problem.

Phase2: Report a Defect

The moment testing team identifies a defect, they need to assign known issues to the development team for further evaluation and fixing process.

Phase3: Acknowledge Defect

Once the test engineers' hand over the defect to the assigned developers, now it is the responsibility of development teams to acknowledge the fault and remain further to fix it if the defect is a valid one.

4. Defect Resolution

Once the **defect discovery** stage has been completed successfully, we move to the next step of the defect management process, **Defect Resolution**.

The **Defect Resolution** is a step-by-step procedure of fixing the defects, or we can say that this process is beneficial in order to specified and track the defects.

This process begins with handing over the defects to the development team. The developers need to proceed with the resolution of the defect and fixed them based on the **priority**.

Once the defect has been selected, the developer sends a defect report of resolution to the test manager's testing team.

The defect resolution process also involves the notification back to the test engineer to confirm that the resolution is verified.

We need to follow the below steps in order to accomplish the defect resolution stage.

- **Prioritize the risk**
- **Fix the defect**
- **Report the Resolution**



Step1: Prioritize the risk

In the first step of defect resolution, the development team evaluates the defects and arranges the fault's fixing. If a defect is more impactful on the system, then developers need to fix those defects on a high priority.

Step2: Fix the defect

In the second step, the developer will fix the defects as per the priority, which implies that the higher priority defects are resolved first. Then the developer will fix the lower priority defects.

Step3: Report the Resolution

In the last step of defect resolution, the developer needs to send the fixed defects report. As it is the responsibility of development teams to make sure that the testing team is well aware of when the defects are going to be fixed and how the fault has been fixed.

This step will be beneficial for the testing team's perspective to understand the root of the defect.

5. Process Improvement

In the above stage (defect resolution), the defects have been arranged and fixed.

Now, in the **process improvement** phase, we will look into the lower priority defects because these defects are also essential as well as impact the system.

All the acknowledged defects are equal to a critical defect from the process improvement phase perspective and need to be fixed.

The people involved in this particular stage need to recall and check from where the defect was initiated.

Depending on that, we can make modifications in the **validation process, base-lining document, review process** that may find the flaws early in the process, and make the process less costly.

These minor defects allow us to learn how we can enhance the process and avoid the existence of any kind of defects that may affect the system or the product failure in the future.

6. Management Reporting

Management reporting is the last stage of the **defect management process**. It is a significant and essential part of the defect management process. The management reporting is required to make sure that the generated reports have an objective and increase the defect management process.

In simple words, we can say that the evaluation and reporting of defect information support organization and risk management, process improvement, and project management.

The information collected on specific defects by the project teams is the root of the management reporting. Therefore, every organization needs to consider the information gathered throughout the defect management process and the grouping of individual defects.

Defect Workflow and States

Various organizations that achieve the **software testing** with the help of a tool that keeps track of the defects during the **bug/defect lifecycle** and also contains defect reports.

Generally, one owner of the defects reports at each state of **defect lifecycle**, responsible for finishing a task that would move defect report to the successive state.

Sometimes, **defect report may not have an owner** in the last phases of the defect lifecycle if we may face the following situation:

- If the defect is invalid, then the Defect report is **cancelled**.
- The defect report is considered **deferred** if the defect won't be fixed as part of the project.
- If the fault cannot be detected anymore, hence defect report is regarded as not **reproducible**.
- The defect report is considered **closed** if the defect has been fixed and tested.

Defect States

If defects are identified throughout the testing, the testing team must manage them in the following three states:

- **Initial state**
- **Returned state**
- **Confirmation state**



1. Initial State

- It is the first state of defect, which is also known as open state.
- One or several test engineers are responsible for collecting all required data to fix the defects in this state.

2. Returned state

- The second state of defect is **returned state**. In this, the person receiving the test report rejects and asks the report creator to provide further information.
- In a returned state, the test engineers can provide more information or accept the rejection of the report.
- If various reports are rejected, the test manager should look out for faults in the initial information collection process itself.
- The returned state is also referred as the **clarification state or rejected state**.

3. Confirmation state

- The last state of defect is **the confirmation state**, where the test engineer performed a **confirmation testing** to make sure that the defect has been fixed.
- It is achieved by repeating the steps, which found the defect at the time of testing.
- If the defect is resolved, then the report is closed.
- And if the defect was not resolved, then the report is considered as **re-opened** and reported back to the owner who formerly preserved the defect report for fixing.
- A confirmation state is also known as **a verified or resolved state**.

Advantages of Defect Management Process

Following are the most significant benefits of the Defect management process:

Confirm Resolution

- The defect management process will also help us to make sure the resolution of defects being tracked.

Accessibility of Automation Tools

- One of the most significant procedures of the defect management process is the **defect or bug tracking process**.
- For defect tracking, we have various automation tools available in the market, which can help us to track the defect in the early stages.
- These days, various different tools are available in order to track different types of defects. **For example,**
 - **Software Tools:** These types of tools are used to identify or track non-technical problems.
 - **User-facing Tools:** These types of tools will help us to discover the defects, which are related to production.

Offer Valuable Metrics

- The defect management process is also offering us valuable defect metrics together with automation tools.
- And these valuable defect metrics help us in reporting and continuous enhancements.

Disadvantages of Defect Management Process

The drawbacks of the defect management process are as follows:

- If the defect management process is not performed appropriately, then we may have a loss of customers, loss of revenue, and damaged brand reputations.
- If the defect management process is not handled properly, then there will be a huge amplified cost in a creeping that is a rise in the price of the product.
- If defects are not accomplished appropriately at an early stage, then afterward, the defect might cause greater damage, and costs to fix the defects will also get enhanced.

Overview

In this article, we have seen the **defects in software testing, the Defect Management Process, benefits, and drawbacks**.

In **software testing**, the Defect Management Process is important as we are aware of any software written code, defects need to be tested.

The process of defect management includes discovering defects in software and fixing them. The complete defect management process will help us to find the defect in the early stages and also make sure to deliver a high-quality product.

The execution of the defect management process ensures that there are no further defects in the application while moving it to production. And the outcome of that will save lots of money.

In agile methodology, the defect management process is specifically significant as the development sprints must also contain the involvement, participation, and action from test engineers.

In any organization, the senior management should also understand and support the defect management process from the perspective of the company's betterment.

What is regression testing?

Regression testing is a black box testing techniques. It is used to authenticate a code change in the software does not impact the existing functionality of the product. Regression testing is making sure that the product works fine with new functionality, bug fixes, or any change in the existing feature.

Regression testing is a type of [software testing](#). Test cases are re-executed to check the previous functionality of the application is working fine, and the new changes have not produced any bugs.

Regression testing can be performed on a new build when there is a significant change in the original functionality. It ensures that the code still works even when the changes are occurring. Regression means Re-test those parts of the application, which are unchanged.

Regression tests are also known as the Verification Method. Test cases are often automated. [Test cases](#) are required to execute many times and running the same test case again and again manually, is time-consuming and tedious too.

Example of Regression testing

Here we are going to take a case to define the regression testing efficiently:

Consider a product Y, in which one of the functionality is to trigger confirmation, acceptance, and dispatched emails. It also needs to be tested to ensure that the change in the code not affected them. Regressing testing does not depend on any programming language like [Java](#), [C++](#), [C#](#), etc. This method is used to test the product for modifications or any updates done. It ensures that any change in a product does not affect the existing module of the product. Verify that the bugs fixed and the newly added features not created any problem in the previous working version of the Software.

When can we perform Regression Testing?

We do regression testing whenever the production code is modified.

We can perform regression testing in the following scenario, these are:

1. When new functionality added to the application.

Example:

A website has a login functionality which allows users to log in only with Email. Now providing a new feature to do login using Facebook.

2. When there is a Change Requirement.

Example:

Remember password removed from the login page which is applicable previously.

3. When the defect fixed

Example:

Assume login button is not working in a login page and a tester reports a bug stating that the login button is broken. Once the bug fixed by developers, tester tests it to make sure Login Button is working as per the expected result. Simultaneously, tester tests other functionality which is related to the login button.

4. When there is a performance issue fix

Example:

Loading of a home page takes 5 seconds, reducing the load time to 2 seconds.

5. When there is an environment change

Example:

When we update the database from MySql to Oracle.

How to perform Regression Testing?

The need for regression testing comes when software maintenance includes enhancements, error corrections, optimization, and deletion of existing features. These modifications may affect system functionality. Regression Testing becomes necessary in this case.

Regression testing can be performed using the following techniques:



1. Re-test All:

Re-Test is one of the approaches to do regression testing. In this approach, all the test case suits should be re-executed. Here we can define re-test as when a test fails, and we determine the cause of the failure is a software fault. The fault is reported, we can expect a new version of the software in which defect fixed. In this case, we will need to execute the test again to confirm that the fault fixed. This is known as re-testing. Some will refer to this as confirmation testing.

The re-test is very expensive, as it requires enormous time and resources.

2. Regression test Selection:

- In this technique, a selected test-case suit will execute rather than an entire test-case suit.
- The selected test case suits divided in two cases
 1. Reusable Test cases.
 2. Obsolete Test cases.
- Reusable test cases can use in succeeding regression cycle.
- Obsolete test cases can't use in succeeding regression cycle.

3. Prioritization of test cases:

Prioritize the test case depending on business impact, critical and frequently functionality used. Selection of test cases will reduce the regression test suite.

What are the Regression Testing tools?

Regression Testing is a vital part of the QA process; while performing the regression we may face the below challenges:

- **Time Consuming** Regression Testing consumes a lot of time to complete. Regression testing involves executing tests again, so testers are not excited to re-run the test.
- **Complex** Regression Testing is complex as well when there is a need to update any product; lists of the test are also increasing.
- **Communicating business rule** Regression Testing ensures the existing product features are still in working order. Communication about regression testing with a non-technical leader can be a difficult task. The

executive wants to see the product move forward and making a considerable time investment in regression testing to ensure existing functionality working can be hard.

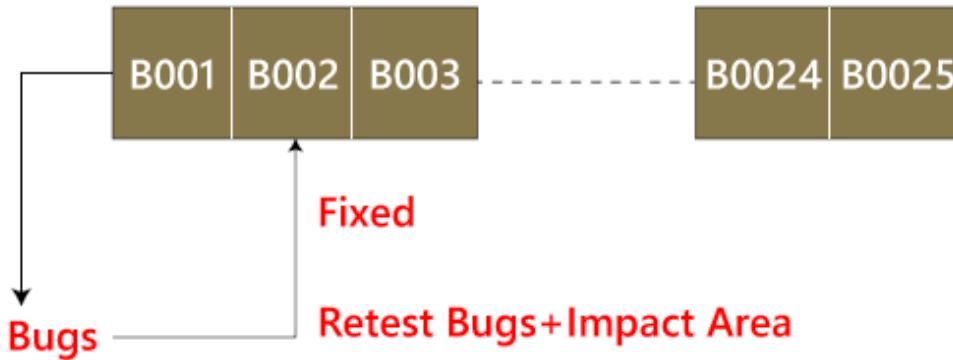
- **Identify Impact Area**
- **Test Cases Increases Release by Release**
- **Less Resources**
- **No Accuracy**
- **Repetitive Task**
- **Monotonous Job**

Regression testing process

The regression testing process can be performed across the **builds** and the **releases**.

Regression testing across the builds

Whenever the bug fixed, we retest the Bug, and if there is any dependent module, we go for a Regression Testing.



For example, How we perform the regression testing if we have different builds as **Build 1, Build 2, and Build 3**, which having different scenarios.

Build1

- Firstly the client will provide the business needs.
- Then the development team starts developing the features.
- After that, the testing team will start writing the test cases; for example, they write 900 test cases for the release#1 of the product.
- And then, they will start implementing the test cases.
- Once the product is released, the customer performs one round of acceptance testing.
- And in the end, the product is moved to the production server.

Build2

- Now, the customer asks for 3-4 extra (new) features to be added and also provides the requirements for the new features.
- The development team starts developing new features.
- After that, the testing team will start writing the test case for the new features, and they write about 150 new test cases. Therefore, the total number of the test case written is 1050 for both the releases.
- Now the testing team starts testing the new features using 150 new test cases.
- Once it is done, they will begin testing the old features with the help of 900 test cases to verify that adding the new feature has damaged the old features or not.

- Here, testing the old features is known as **Regression Testing**.
- Once all the features (New and Old) have been tested, the product is handed over to the customer, and then the customer will do the acceptance testing.
- Once the acceptance testing is done, the product is moved to the production server.

Build3

- After the second release, the customer wants to remove one of the features like Sales.
- Then he/she will delete all the test cases which are belonging to the sales module (about 120 test cases).
- And then, test the other feature for verifying that if all the other features are working fine after removing the sales module test cases, and this process is done under the regression testing.

Note:

- Testing the stable features to ensure that it is broken because of the changes. Here changes imply that the **modification, addition, bug fixing, or the deletion**.
- Re-execution of the same test cases in the different builds or releases is to ensure that changes (modification, addition, bug fixing, or the deletion) are not introducing bugs in stable features.

Regression testing across the release

The regression testing process starts whenever there is a new Release for same project because the new feature may affect the old elements in the previous releases.

To understand the regression testing process, we will follow the below steps:

Step1

There is no regression testing in **Release#1** because there is no modification happen in the Release#1 as the release is new itself.

Step2

The concept of Regression testing starts from **Release#2** when the customer gives some **new requirements**.

Step3

After getting the new requirements (modifying features) first, they (the developers and test engineers) will understand the needs before going to the **impact analysis**.

Step4

After understanding the new requirements, we will perform one round of **impact analysis** to avoid the major risk, but here the question arises who will do the Impact analysis?

Step5

The impact analysis is done by the **customer** based on their **business knowledge**, the **developer** based on their **coding knowledge**, and most importantly, it is done by the **test engineer** because they have the **product knowledge**.

Note: If a single person does, he/she may not cover all the impact areas, so we include all persons so that we may cover a maximum impact area, and Impact Analysis should be done at the early stages of the releases.

Step6

Once we are done with the **impact area**, then the developer will prepare the **impact area (document)**, and the **customer** will also prepare the **impact area document** so that we can achieve the **maximum coverage of impact analysis**.

Step7

After completing the impact analysis, the developer, the customer, and the test engineer will send the **Reports#** of the impact area documents to the **Test Lead**. And in the meantime, the test engineer and the developer are busy working on the new test case.

Step8

Once the Test lead gets the Reports#, he/she will **consolidate** the reports and stored in the **test case requirement repository** for the release#1.

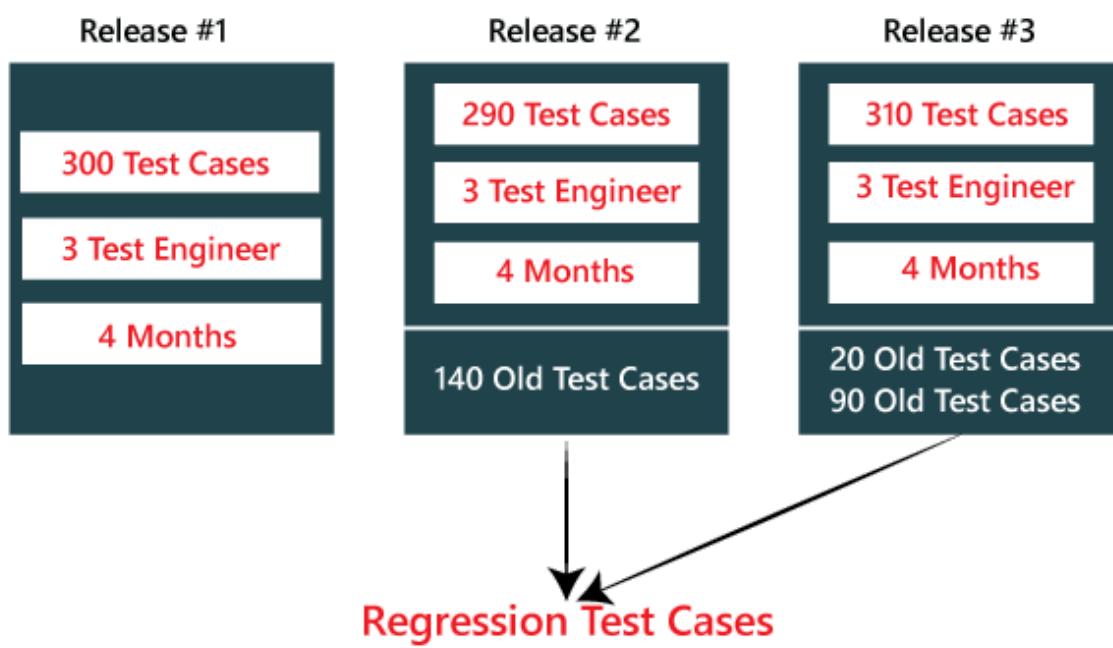
Note: Test case Repository: Here, we will save all the test case of releases.

Step9

After that, the Test Lead will take the help of RTM and pick the necessary **regression test case** from the **test case repository**, and those files will be placed in the **Regression Test Suite**.

Note:

- The test lead will store the regression test case in the regression test suite for no further confusion.
- **Regression test suite:** Here, we will save all the impact area test documents.
- **Regression Test Cases:** These are the test cases of the old releases text document which need to be re-executed as we can see in the below image:

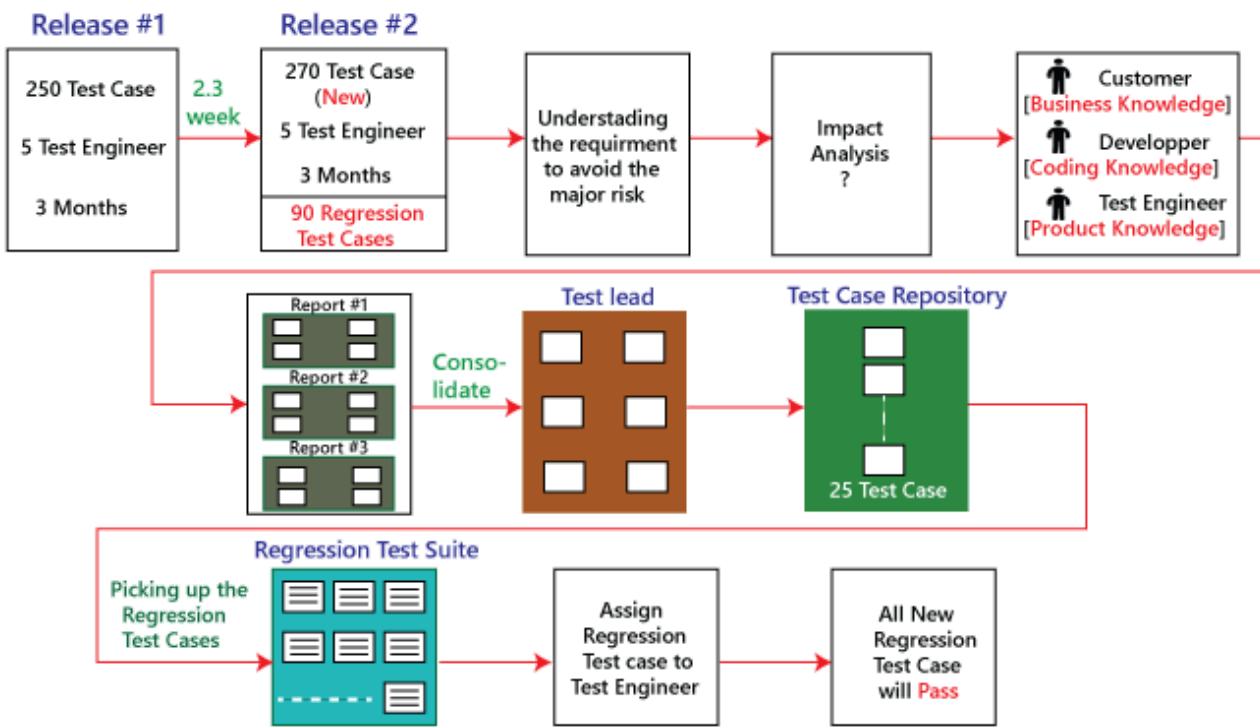


Step10

After that, when the test engineer has done working on the new test cases, the test lead will **assign the regression test case** to the test engineer.

Step11

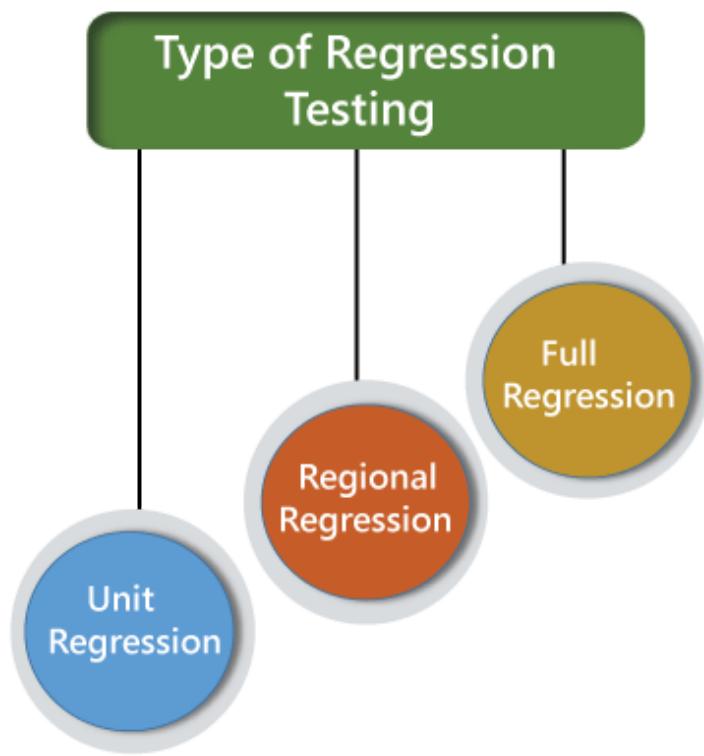
When all the regression test cases and the new features are **stable and pass**, then check the **impact area using the test case** until it is durable for old features plus the new features, and then it will be handed over to the customer.



Types of Regression Testing

The different types of Regression Testing are as follows:

- Unit Regression Testing [URT]
- Regional Regression Testing[RRT]
- Full or Complete Regression Testing [FRT]



Unit Regression Testing [URT]

In this, we are going to test only the changed unit, not the impact area, because it may affect the components of the same module.

Example1

In the below application, and in the first build, the developer develops the **Search** button that accepts **1-15 characters**. Then the test engineer tests the Search button with the help of the **test case design technique**.

(Search field)1-15 charcters

SEARCH **CANCEL**

Build 1-B001

(Search field)1-35 charcters

SEARCH **CANCEL**

Build 1-B002

Now, the client does some modification in the requirement and also requests that the **Search button** can accept the **1-35 characters**. The test engineer will test only the Search button to verify that it takes 1-35 characters and does not check any further feature of the first build.

Example2

Here, we have **Build B001**, and a defect is identified, and the report is delivered to the developer. The developer will fix the bug and sends along with some new features which are developed in the second **Build B002**. After that, the test engineer will test only after the defect is fixed.

- The test engineer will identify that clicking on the **Submit** button goes to the blank page.
- And it is a defect, and it is sent to the developer for fixing it.
- When the new build comes along with the bug fixes, the test engineer will test only the Submit button.
- And here, we are not going to check other features of the first build and move to test the new features and sent in the second build.
- We are sure that fixing the **Submit**button is not going to affect the other features, so we test only the fixed bug.

Create User

Name	<input type="text"/>
Address	<input type="text"/>
Telephone No.	<input type="text"/>
Email ID	<input type="text"/>
.....	
SUBMIT	CANCEL

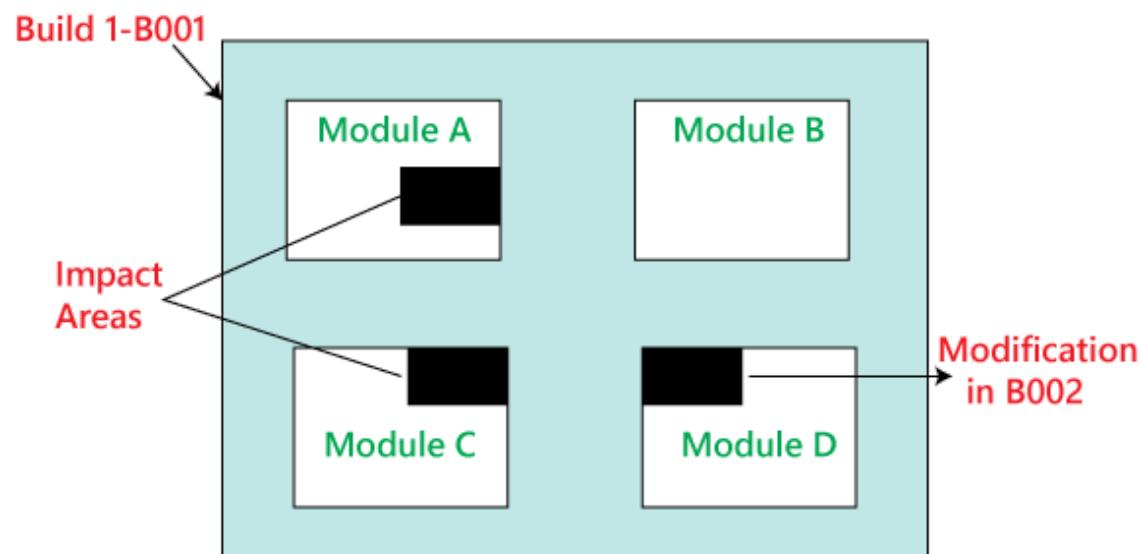
Therefore, we can say that by testing only the changed feature is called the **Unit Regression Testing**.

Regional Regression testing [RRT]

In this, we are going to test the modification along with the impact area or regions, are called the **Regional Regression testing**. Here, we are testing the impact area because if there are dependable modules, it will affect the other modules also.

For example:

In the below image as we can see that we have four different modules, such as **Module A**, **Module B**, **Module C**, and **Module D**, which are provided by the developers for the testing during the first build. Now, the test engineer will identify the bugs in **Module D**. The bug report is sent to the developers, and the development team fixes those defects and sends the second build.



In the second build, the previous defects are fixed. Now the test engineer understands that the bug fixing in Module D has impacted some features in **Module A and Module C**. Hence, the test engineer first tests the Module D where the bug has been fixed and then checks the impact areas in **Module A and Module C**. Therefore, this testing is known as **Regional regression testing**.

While performing the regional regression testing, we may face the below problem:

Problem:

In the first build, the client sends some modification in requirement and also wants to add new features in the product. The needs are sent to both the teams, i.e., development and testing.

After getting the requirements, the development team starts doing the modification and also develops the new features based on the needs.

Now, the test lead sends mail to the clients and asks them that all are the impact areas that will be affected after the necessary modification have been done. Therefore, the customer will get an idea, which all features are needed to be tested again. And he/she will also send a mail to the development team to know which all areas in the application will be affected as a result of the changes and additions of new features.

And similarly, the customer sends a mail to the testing team for a list of impact areas. Hence, the test lead will collect the impact list from the client, development team, and the testing team as well.

This **Impact list** is sent to all the test engineers who look at the list and check if their features are modified and if yes, then they do **regional regression testing**. The impact areas and modified areas are all tested by the respective engineers. Every test engineer tests only their features that could have been affected as a result of the modification.

The problem with this above approach is that the test lead may not get the whole idea of the impact areas because the development team and the client may not have so much time to revert his/her mails.

Solution

To resolve the above problem, we will follow the below process:

When a new build comes along with the latest features and bug fixes, the testing team will arrange the meeting where they will talk about if their features are affecting because of the above modification. Therefore, they will do one round of **Impact Analysis** and generate the **Impact List**. In this particular list, the test engineer tries to enclose the maximum probable impact areas, which also decreases the chance of getting the defects.

When a new build comes, the testing team will follow the below procedure:

- They will do smoke testing to check the basic functionality of an application.
- Then they will test new features.
- After that, they will check the changed features.
- Once they are done with checking the changed features, the test engineer will re-test the bugs.
- And then they will check the impact area by performing the regional regression testing.

Disadvantage of using Unit and Regional Regression testing

Following are some of the drawbacks of using unit and Regional regression testing:

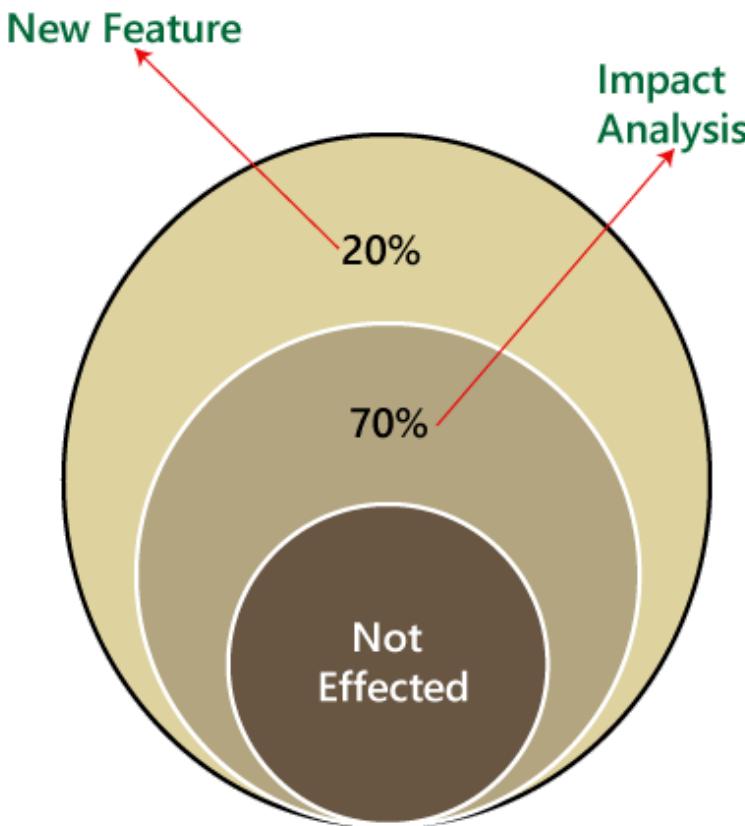
- We may miss some impact area.
- It is possible that we may identify the wrong impact area.

Note: We can say that the major work we do on the regional regression testing will lead us to get more number of defects. But, if we will perform the same dedication to work on the full regressing testing, we will get less number of defects. Therefore, we can determine here that enhancement in the testing effort will not help us to get more defects.

Full Regression testing [FRT]

During the second and the third release of the product, the client asks for adding 3-4 new features, and also some defects need to be fixed from the previous release. Then the testing team will do the Impact Analysis and identify that the above modification will lead us to test the entire product.

Therefore, we can say that testing the **modified features** and **all the remaining (old) features** is called the **Full Regression testing**.



When we perform Full Regression testing?

We will perform the FRT when we have the following conditions:

- When the modification is happening in the source file of the product. **For example**, JVM is the root file of the JAVA application, and if any change is going to happen in JVM, then the entire JAVA program will be tested.
- When we have to perform n-number of changes.

Note:

The regional regression testing is the ideal approach of regression testing, but the issue is, we may miss lots of defects while performing the Regional Regression testing.

And here we are going to solve this issue with the help of the following approach:

- When the application is given for the testing, the test engineer will test the first 10-14 cycle, and will do the **RRT**.
- Then for the 15th cycle, we do FRT. And again, for the next 10-15 cycle, we do **Regional regression testing**, and for the 31th cycle, we do the **full regression testing**, and we will continue like this.
- But for the last ten cycle of the release, we will perform only **complete regression testing**.

Therefore, if we follow the above approach, we can get more defects.

The drawback of doing regression testing manually repeatedly:

- Productivity will decrease.
- It is a difficult job to do.
- There is no consistency in test execution.
- And the test execution time is also increased.

Hence, we will go for the automation to get over with these issues; when we have n-number of the regression test cycle, we will go for the **automation regression testing process**.

Automated Regression testing process

Generally we go for the automation whenever there are multiple releases or multiple regression cycle or there is the repetitive task.

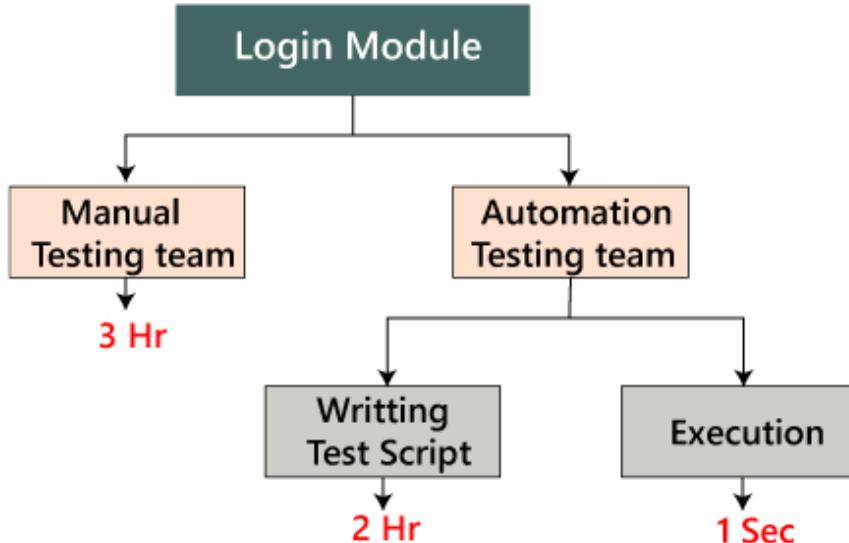
The automation regression testing process can be done in the following steps:

Note1:

The process of testing the application by using some tools is known as automation testing.

Suppose if we take one sample example of a **Login module**, then how we can perform the regression testing.

Here, the Login can be done in two ways, which are as follows:



Manually: In this, we will perform regression only one and twice.

Automation: In this, we will do the automation multiple times as we have to write the test scripts and do the execution.

Note2: In real-time, if we have faced some issues such as:

Issues	Handle by
New features	Manual test engineer
Regressing testing features	Automation test engineer
Remaining (110 feature + Release#1)	Manual test engineer

Step1

When the new release starts, we don't go for the automation because there is no concept of regression testing and regression test case as we understood this in the above process.

Step2

When the new release and the enhancement starts, we have two teams, i.e., manual team and the automation team.

Step3

The manual team will go through the requirements and also identify the impact area and hand over the **requirement test suite** to the automation team.

Step4

Now, the manual team starts working on the new features, and the automation team will start developing the test script and also start automating the **test case**, which means that the regression test cases will be converted into the test script.

Step5

Before they (automation team) start automating the test case, they will also analyze which all cases can be automated or not.

Step6

Based on the analysis, they will start the automation i.e., converting every regression test cases into the test script.

Step7

During this process, they will take help of the **Regression cases** because they don't have product knowledge as well as the **tool** and the **application**.

Step8

Once the test script is ready, they will start the execution of these scripts on the new application [old feature]. Since, the test script is written with the help of the regression feature or the old feature.

Step9

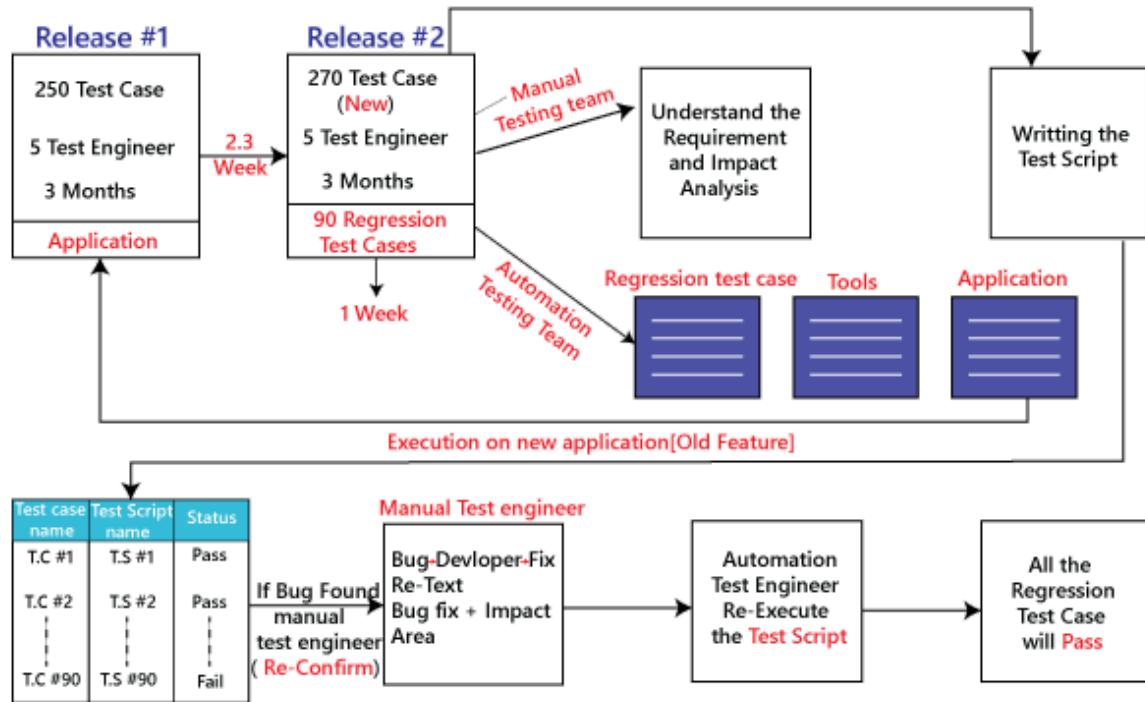
Once the execution is completed, we get a different status like **Pass/fail**.

Step10

If the status is failed, which means it needs to be re-confirmed manually, and if the Bug exists, then it will report to the concerned developer. When the developer fixes that bug, the Bug needs to be re-tested along with the Impact area by the manual test engineer, and also the script needs to be re-executed by the automation test engineer.

Step11

This process goes on until all the new features, and the regression feature will be passed.



Benefits of doing regression testing by the automation testing:

- **Accuracy** always exists because the task is done by the tools and tools never get bored or tired.
- The test script can be re-used across multiple releases.
- **Batch execution** is possible using the automation i.e.; all the written test scripts can be executed parallel or simultaneously.
- Even though the number of regression test case increase release per release, and we don't have to increase the automation resource since some regression case are already automated from the previous release.
- It is a **time-saving process** because the execution is always faster than the manual method.

How to select test cases for regression testing?

It was found from industry inspection. The several defects reported by the customer were due to last-minute bug fixes. These creating side effects and hence selecting the Test Case for regression testing is an art, not an easy task.

Regression test can be done by:

- A test case which has frequent defects
- Functionalities which are more visible to users.
- Test cases verify the core features of the product.
- All integration test cases
- All complex test cases
- Boundary value test cases
- A sample of successful test cases
- Failure of test cases

Regression testing tools

If Software undergoes frequent changes, regression testing costs also increase. In those cases, manual execution of test cases increases test execution time as well as costs. In that case, automation testing is the best choice. The duration of automation depends on the number of test cases that remain reusable for successive regression cycles.

Following are the essential tools used for regression testing:

Selenium

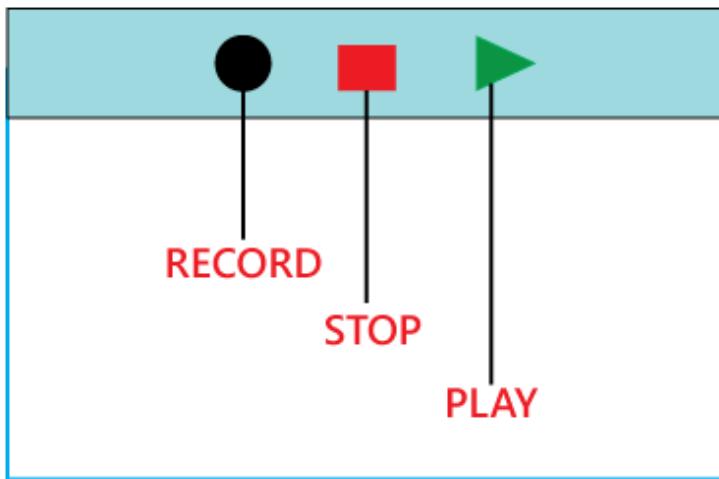
Selenium is an open-source tool. This tool used for automated testing of a web application. For browser-based regression testing, selenium used. Selenium used for UI level regression test for web-based application.

Ranorex Studio

All in one regression test automation for desktop, web, and mobile apps with built-in Selenium Web Driver. Ranorex Studio includes full IDE plus tools for codeless automation.

Quick Test Professional (QTP)

QTP is an automated testing tool used for Regression and Functional Testing. It is a Data-Driven, keyword-based tool. It used VBScript language for automation. If we open the QTP tool, we see the three buttons which are **Record, Play and Stop**. These buttons help to record every click and action performed on the computer system. It records the actions and play it back.



Rational Functional Tester (RTF)

Rational functional tester is a Java tool used to automate the test cases of software applications. RTF used for automating regression test cases, and it also integrates with the rational functional tester.

For more information about regression and automation testing tools refer to the below link:

<https://www.javatpoint.com/automation-testing-tool>

What are the Regression Testing and Configuration Management?

Configuration Management in the regression testing becomes imperative in Agile Environments, where a code is continuously modified. To ensure a valid regression test, we must follow the steps:

- Changes are not allowed in the code during the regression testing phase.
- A regression test case must be unaffected developer changes.
- The database used for regression testing must be isolated; changes are not allowed in the database.

What are the differences between Retesting and Regression Testing?

Re-testing Testing means testing the functionality or bug again to ensure the code fixed. If not set, defects need not be re-opened. If fixed, the defect closed.

Re-testing is a type of testing which performed to check the test-cases that were unsuccessful in the final execution are successfully pass after the defects repaired.

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

Regression testing is a type of testing executed to check whether a code has not changed the existing functionality of the application.

Differences between the Re-testing and Regression Testing are as follows:

Re-testing	Regression Testing
Re-testing is performed to ensure that the test cases that are failed in the final execution are passing after the defects fixed.	Regression Testing is done to confirm whether the code change has not affected the existing features.
Re-Testing works on defect fixes.	The purpose of regression testing is to ensure that the code changes adversely not affect the existing functionality.
Defect verification is the part of the Retesting.	Regression testing does not include defect verification
The priority of Retesting is higher than Regression Testing, so it is done before the Regression Testing.	Based on the project type and availability of resources, regression testing can be parallel to Retesting.
Re-Test is a planned Testing.	Regression testing is a generic Testing.
We cannot automate the test-cases for Retesting.	We can do automation for regression testing; manual testing could be expensive and time-consuming.
Re-testing is for failed test-cases.	Regression testing is for passed Test-cases.
Re-testing make sure that the original fault is corrected.	Regression testing checks for unexpected side effect.
Retesting executes defects with the same data and the same environment with different input with a new build.	Regression testing is when there is a modification or changes become mandatory in an existing project.
Re-testing cannot do before start testing.	Regression testing can obtain test cases from the functional specification, user tutorials and manuals, and defects reports in regards to the corrected problem.

What are the advantages of Regression Testing?

Advantages of Regression Testing are:

- Regression Testing increases the product's quality.
- It ensures that any bug fix or changes do not impact the existing functionality of the product.
- Automation tools can be used for regression testing.
- It makes sure the issues fixed do not occur again.

What are the disadvantages of Regression Testing?

There are several advantages of Regression Testing though there are disadvantages as well.

- Regression Testing should be done for small changes in the code because even a slight change in the code can create issues in the existing functionality.
- If in case automation is not used in the project for testing, it will be time consuming and tedious task to execute the test again and again.

Conclusion

Regression Testing is one of the essential aspects as it helps to deliver a quality product which saves organizations time and money. It helps to provide a quality product by making sure that any change in the code does not affect the existing functionality.

For automating the regression test cases, there are several automation tools available. A tool should have the ability to update the **test suite** as the regression test suit needs to be updated frequently.

Smoke Testing

Smoke Testing comes into the picture at the time of receiving build software from the development team. The purpose of smoke testing is to determine whether the build software is testable or not. It is done at the time of "building software." This process is also known as "Day 0".

It is a time-saving process. It reduces testing time because testing is done only when the key features of the application are not working or if the key bugs are not fixed. The focus of Smoke Testing is on the workflow of the core and primary functions of the application.

Testing the basic & critical feature of an application before doing one round of deep, rigorous testing (before checking all possible positive and negative values) is known as smoke testing.

In the smoke testing, we only focus on the positive flow of the application and enter only valid data, not the invalid data. In smoke testing, we verify every build is testable or not; hence it is also known as **Build Verification Testing**.

When we perform smoke testing, we can identify the blocker bug at the early stage so that the test engineer won't sit idle, or they can proceed further and test the **independent testable modules**.

Note:

- The test engineer knows that the module is independent testable because we have already done one round of smoke testing on them.
- The development team can take their time to fix the bug, and they are not under pressure because the testing team is not sitting idle, and the release does not get postponed, hence it is a time-saving process.

Process to conduct Smoke Testing

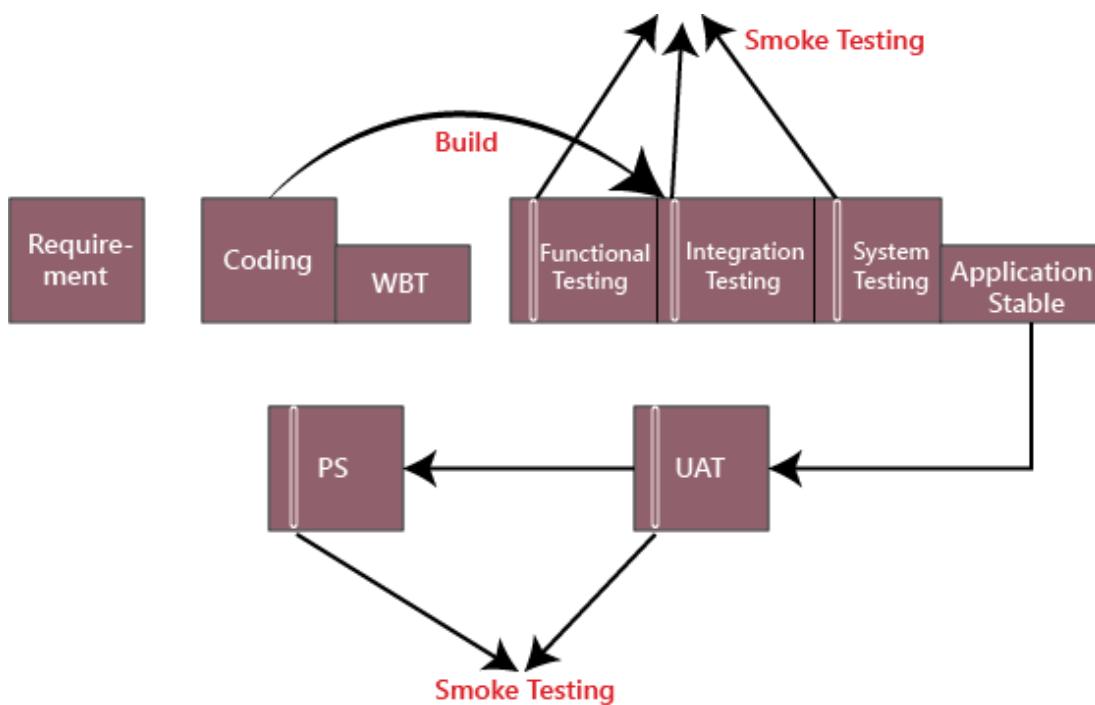
Smoke testing does not require to design test cases. There's need only to pick the required test cases from already designed test cases.

As mentioned above, Smoke Testing focuses on the workflow of core applications so; we choose test case suits that covers the major functionality of the application. The number of test cases should be minimized as much as possible and time of execution must not be more than half an hour.

When we perform smoke testing

Generally, whenever the new build is installed, we will perform one round of smoke testing because in the latest build, we may encounter the blocker bug. After all, there might be some change that might have broken a major feature (fixing the bug of or adding a new feature could have affected a major portion of the original software), or we do smoke testing where the installation is happening.

When the stable build is installed anywhere (Test Server, Production Server, and User Acceptance testing), we do smoke testing to find the blocker bug.

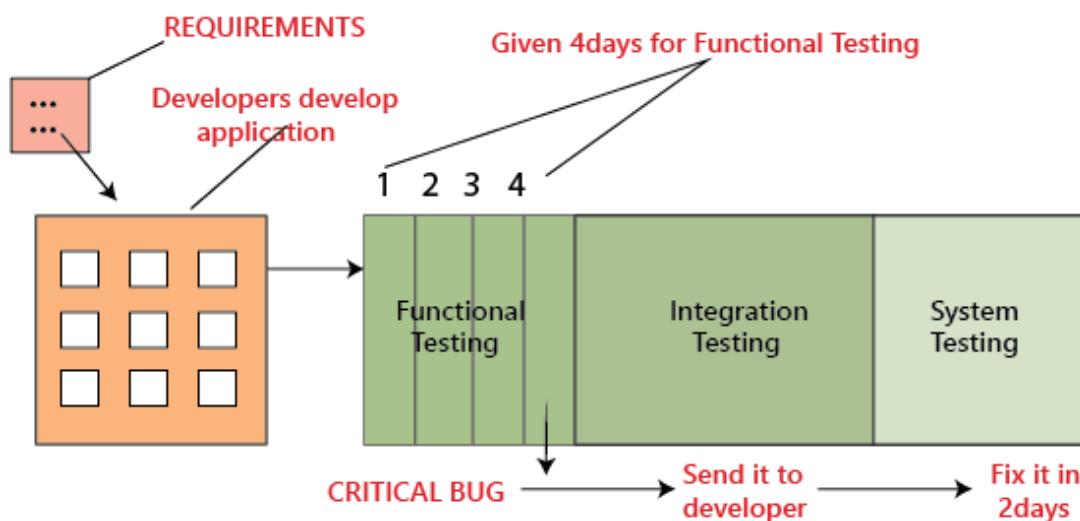


Let's use some different scenarios, which help us to understand better when to do smoke testing:

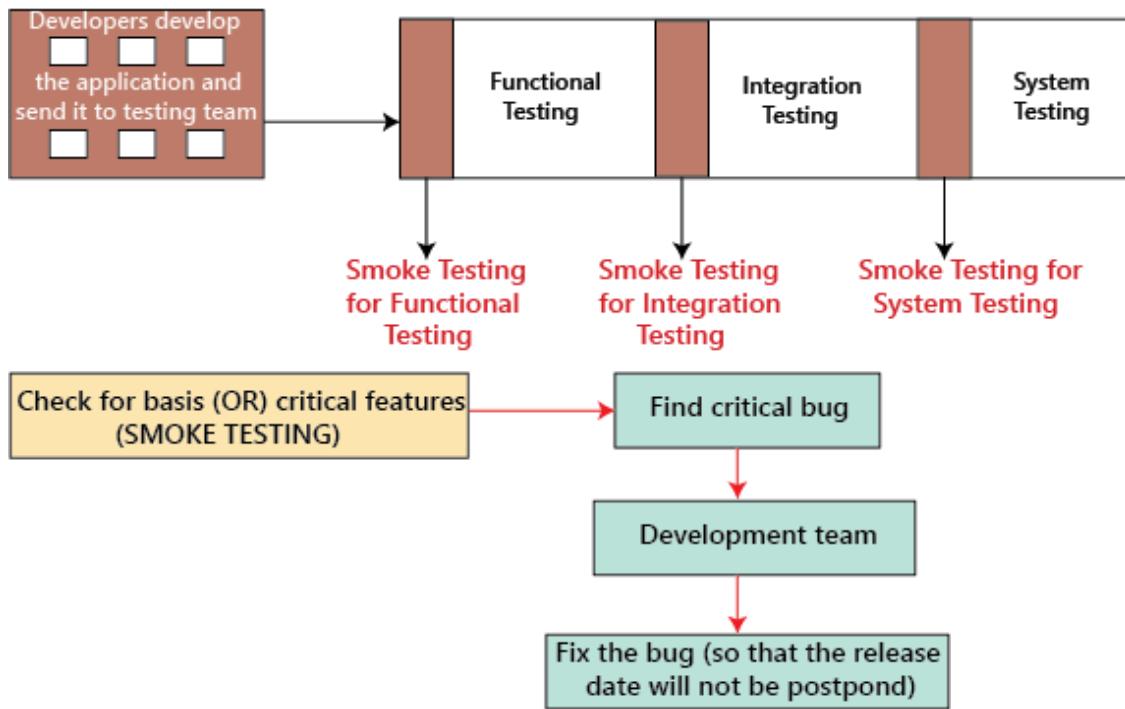
Scenarios 1

The developer develops the application and handed over to the testing team, and the testing team will start the functional testing

Suppose we assume that four days we are given to the **functional testing**. On the first day, we check one module, and on the second day, we will go for another module. And on the fourth day, we find a **critical bug** when it is given it to the developer; he/she says it will take another two days to fix it. Then we have to postpone the release date for these extra two days.



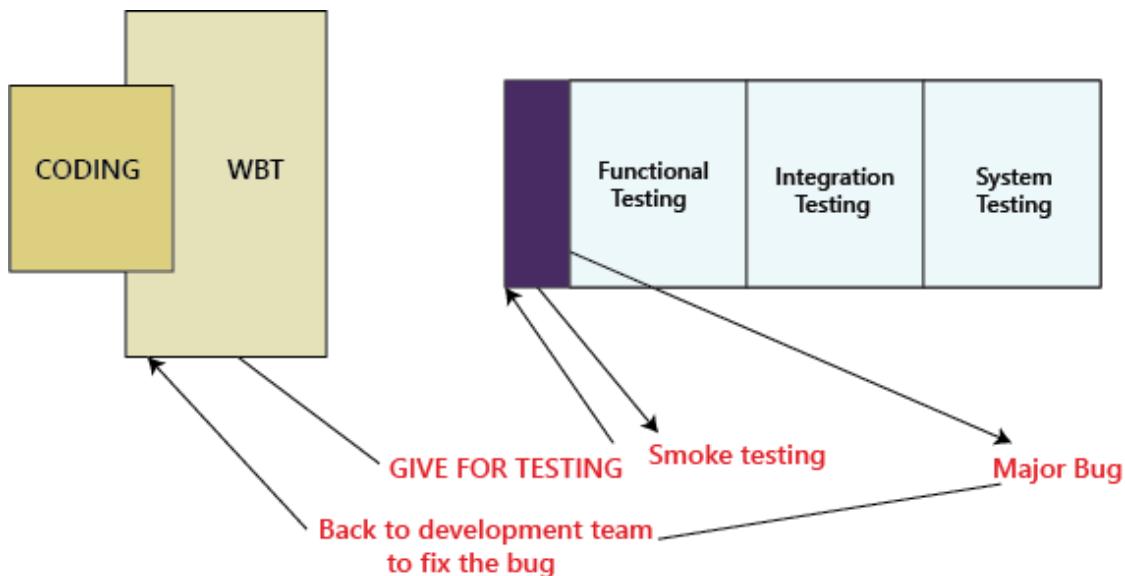
To overcome this problem, we perform **smoke testing**, let us see how it works, in the above situation, instead of the testing module by module thoroughly and come up with critical bug at the end, it is better to do **smoke testing** before we go for functional, integration and system testing that is, in each module we have to test for essential or critical features, and then proceed for further testing as we can see in the below images:



Scenario 2

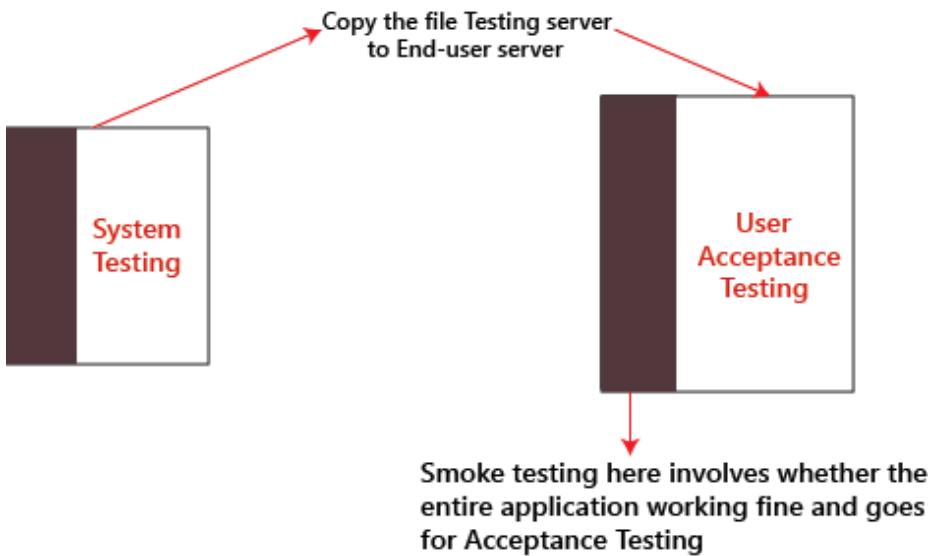
While performing the functional testing, if the test engineer identifies the major bug in the early stages, sometimes it is not suitable for the developer to find a major bug in the initial stages. So the test engineer will perform the smoke testing before doing the functional, integration, system, and other types of testing.

While doing smoke testing, the test engineer finds the major bug; he/she will give to the development team for fixing the bug. After the bug is fixed, the test engineer will continue for further testing as we can see in the below image:



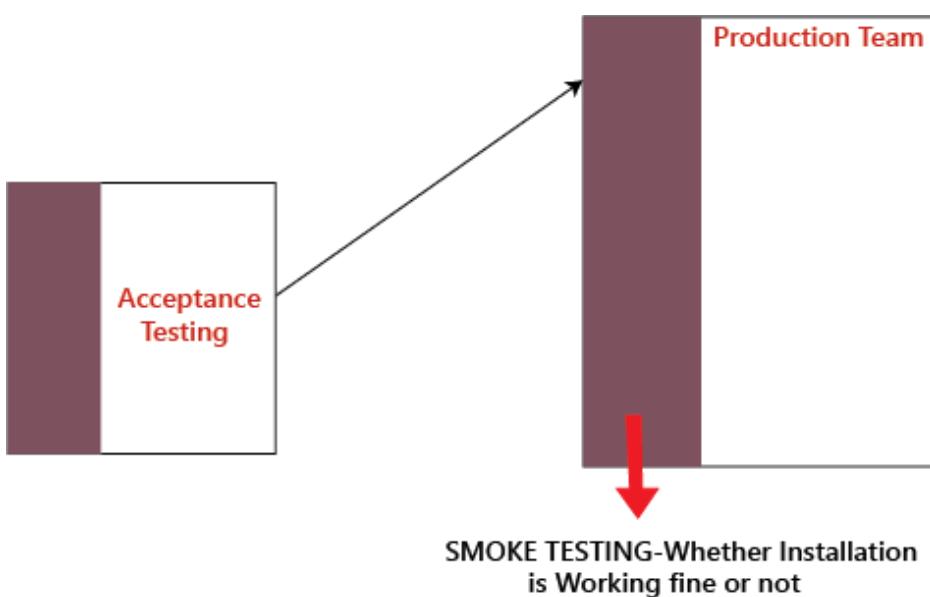
Scenario 3

In this scenario, if we are already perform the smoke testing and found the blocker bug and also resolved that bug. After performing the system testing, we will send the application from the testing server to the end-user server for one round of user acceptance testing. And when the customer performs the acceptance testing and does not find any issues and satisfied with the application because we already perform the smoke testing.



Scenarios 4

Once the acceptance testing is done, the application will be deployed to the production server. We have done a round of smoke testing on the production server to check whether the application is installed correctly or not. If any real end-user will find any blocker bug, they will get irritated and will not use the application again, which may lead to the loss of customer business as we can see in the below image:



For not getting this problem in the future, the development team manager, the testing team manager, will take the customer login and do one round of smoke testing.

For example, the real user using the Facebook application and every time we got new features updated internally, and the actual user will not affect because they will not aware of the internal changes and use the application properly.

In the production server, smoke testing can be done by **the Business analyst (BA)**, **Development team manager**, **testing team manager**, **build team**, and **the customer**.

Why we do smoke testing?

- We will do the smoke testing to ensure that the product is testable.
- We will perform smoke testing in the beginning and detect the bugs in the basic features and send it to the development team so that the development team will have enough time to fix the bugs.
- We do smoke testing to make sure that the application is installed correctly.

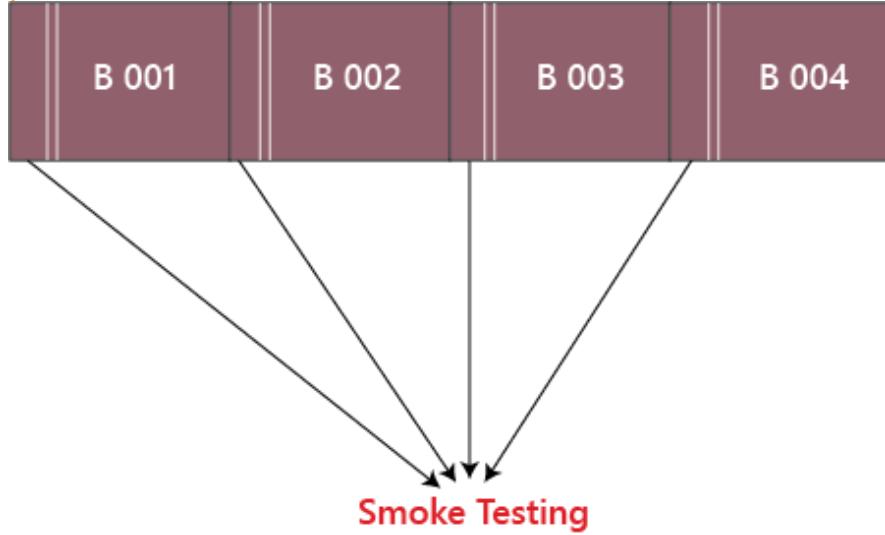
Note:

- In the early stage of application development, if we are doing smoke testing, it will fetch more number of bugs. But in the later stage of application development, if we do smoke testing, the

number of bugs that we are going to catch in smoke testing will be very less. Therefore, frequently the effort spent on smoke testing is less.

- **When we go for smoke testing on every build?**

Whenever a new build is installed, we make sure that build is testable or not, and if it is testable, then we perform smoke testing like as we can see in the below image:



Types of smoke testing

Smoke testing is divided into two types:

Formal smoke testing

In this, the development team sends the application to the Test Lead. Then the test lead will instruct the testing team to do smoke testing and send the reports after performing the smoke testing. Once the testing team is done with smoke testing, they will send the smoke testing report to the test lead.

Informal smoke testing

Here, the Test lead says that the application is ready for further testing. The test leads do not specify to do smoke testing, but still, the testing team starts testing the application by doing smoke testing.

Real Time Example:

Suppose, we are using an eCommerce site, and the core working of this site should be login, specific search, add an item into the cart, add an item into the favorite, payment options, etc. Here we are testing function to place an order. After testing, the tester has to be sure and confident about the functioning of the function of the application.

Steps of the workflow are given below:

- Click on item
- Description page should be open.
- Click on Add to Cart
- The cart should be open
- Click on Buy Now
- Payment options should be displayed. Choose one of them.
- Order placed

01 *Click On Item*

02 *Description page
should be open*

03 *Click on Add to Cart*

04 *Cart should be open*

05 *Click on Buy Now*

06 *Payment options
should be display*

07 *Order placed*

If this function is working correctly, then tester will pass it in testing and test the next function of the same application.

Advantages of smoke testing

- It is a time-saving process.
- In the early stage, we can find the bugs.
- It will help to recover the quality of the system, which decreases the risk.
- It is easy testing to perform because it saves our test effort and time.

Sanity Testing

In this section, we are going to understand the working of sanity testing, which is used to check whether the bugs have been fixed after the build or not.

And we also learn about **its process, why we need to perform the sanity testing, the objective of sanity testing, real-time examples, various attributes of sanity testing, advantages, and disadvantages**.

What is Sanity Testing?

Generally, Sanity testing is performed on stable builds and it is also known as a variant of **regression testing**.

Sanity testing was performed when we are receiving software build (with minor code changes) from the development team. It is a checkpoint to assess if testing for the build can proceed or not.

In other words, we can say that sanity testing is performed to make sure that all the defects have been solved and no added issues come into the presence because of these modifications.

Sanity testing also ensures that the modification in the code or functions does not affect the associated modules. Consequently, it can be applied only on connected modules that can be impacted.

The Objective of Sanity Testing

The key objective of implementing sanity testing is to fulfill the following aspects:

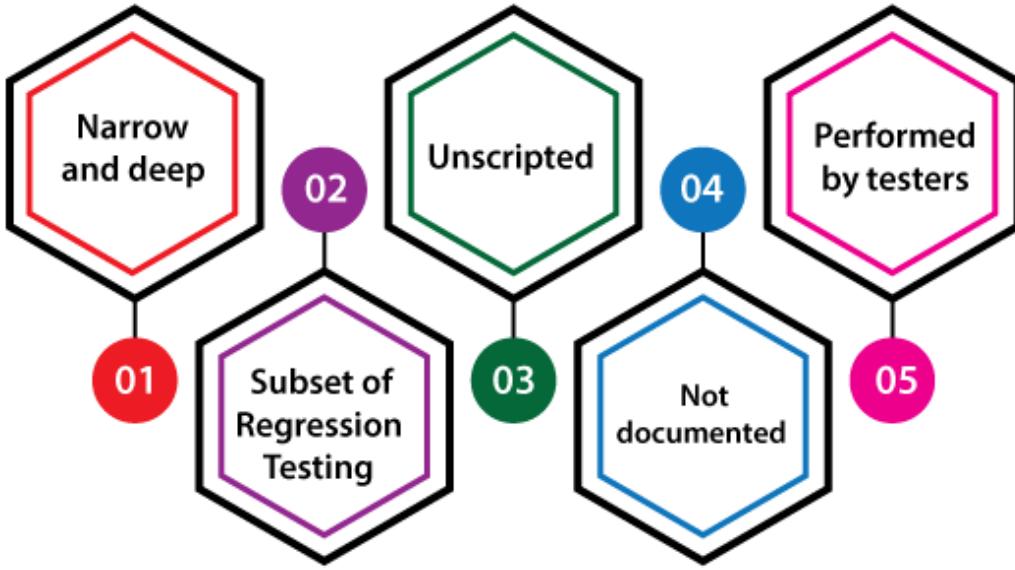


- The primary aim of executing the sanity testing is to define that the planned features work unevenly as expected. If the sanity test fails, the build is refused to save the costs and time complexity in more severe testing.
- The execution of sanity testing makes sure that new modifications don't change the software's current functionalities.
- It also validates the accuracy of the newly added features and components.

Attributes of Sanity Testing

For understanding the fundamental of the **sanity testing techniques**, we have to learn their attribute and several other components. Hence, following are some of the important features of Sanity testing:

- **Narrow and deep**
- **A Subset of Regression Testing**
- **Unscripted**
- **Not documented**
- **Performed by testers**



Narrow and deep

In software testing, sanity testing is a **narrow and deep** method where limited components are tested deeply.

Subcategory of Regression Testing

It is a subdivision of **regression testing**, which mainly emphasizes on the less important unit of the application.

It is used to test the application efficiency under the requirements of the modification or new features that have been executed.

Unscripted

Generally, sanity testing is unscripted.

Not documented

Typically, sanity testing cannot be documented.

Performed by test engineers

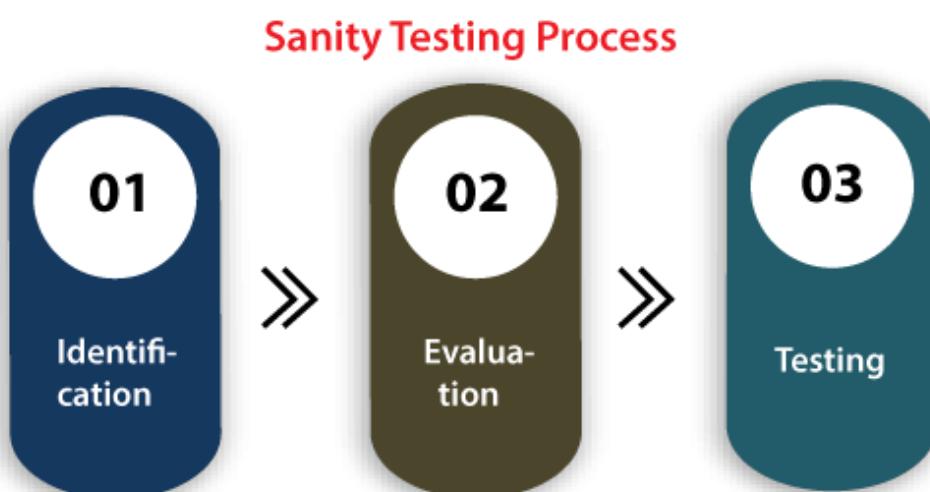
Usually, Sanity testing is done by the test engineers.

Sanity Testing Process

The main purpose of performing sanity testing is to check the incorrect outcomes or defects which are not existing in component procedures. And also, ensure that the newly added features may not affect the functionalities of current features.

Therefore, we need to follow the below steps to implement the sanity testing process gradually:

- **Identification**
- **Evaluation**
- **Testing**



Step1: Identification

The first step in the sanity testing process is **Identification**, where we detect the newly added components and features as well as the modification presented in the code while fixing the bug.

Step2: Evaluation

After completing the identification step, we will analyze newly implemented components, attributes and modify them to check their intended and appropriate working as per the given requirements.

Step3: Testing

Once the identification and evaluation step are successfully processed, we will move to the next step, which is **testing**.

In this step, we inspect and assess all the linked parameters, components, and essentials of the above analyzed attributed and modified them to make sure that they are working fine.

If all the above steps are working fine, the build can be subjected to more detailed and exhausting testing, and the release can be passed for thorough testing.

Who executes the Sanity testing?

Generally, a sanity test case is executed by the **test engineers**.

When do we need to perform Sanity testing?

There are no such hard and fast software testing rules to execute the sanity tests process.

It is a quick process of testing the application as it does not include the scripting any of the test cases.

A Sanity testing is a narrow regression test that emphasizes specific areas of the component. And if we encounter the below two conditions, we needed to execute one round of sanity testing, and those conditions are as follows:

Case1

We go for sanity testing whenever there is an improvement in the functionality of the specified software.

Case2

Whenever the bugs have been fixed, or a new feature added, we need to perform sanity testing in order to check whether the application is still working fine or not.

Examples of Sanity Testing

For our better understanding of sanity testing, we will see the below example:

Example 1

Suppose we have an **e-commerce application**, which contains several modules, but here, we mainly concentrate only a few modules such as **the login page, the home page, the new user creation page, the user profile page, etc.**

- While a new user tries to login into the application, he/she is not able to log in, as there is a bug in the **login page**.
- Because the **password field** in the **login module** accepts less than four alpha-numeric characters and based on the specification, the password field should not be accepted below 7-8 characters.
- Thus, it is considered as bug, which is reported by the testing team to the development team to fix it.
- Once the development team fixes the specified bug and reports back to the testing team, the testing team tests the same feature to verify that the modification that happens in the code is working fine or not.

- And the testing team also verifies that the particular modification does not impact other related functionalities.
- To modify the **password** on the user **profile page** there is a process.
- As part of the sanity testing process, we must authenticate the **login page** and the **profile page** to confirm that the changes are working fine at both the places.

Advantages and Disadvantages of Sanity Testing

Below are some of the vital benefits and drawbacks of Sanity testing.

Advantages of Sanity Testing

Some of the dynamic benefits of performing sanity testing are as follows:

Sanity testing is easy to understand and implement.

- It helps us to find any deployment or compilation issues.
- It is less expensive as compared to other types of software testing.
- It helps in rapidly finding the bugs in the core functionality.
- There is no documentation mandatory for sanity testing, that's why it can be executed in lesser time.
- The execution of sanity testing will help us save unnecessary testing effort and time because it only focused on one or a few functionality areas.
- Sanity testing helps in detecting the missing dependent objects.

Disadvantages of Sanity testing

Following are the drawbacks of sanity testing:

- It's become a very complex process for the developers to understand how to fix the defects acknowledged throughout the sanity testing if they do not follow the design structure level.
- All the test cases are not covered under sanity testing.
- It is emphasized only on the statement and functions of the application.
- We do not have future references since the sanity testing is unscripted.
- It became a complex process to find any other components as sanity testing is executed only for some limited features.

Overview

In this tutorial, we learned that the execution of the sanity testing, real-time examples, benefits, and drawbacks.

Sanity testing is implemented when a new functionality, modification request, or bug fix is executed in the program.

It is a **narrow and deep** testing process that is intensive only on those components where the modification has impacted.

Sanity testing is beneficial as it provides various advantages like, it offers a quick assessment of the quality of software release.

Sanity testing allows us to check the application's small functionality if a minor change occurs in the software.

Static Testing

In this section, we are going to understand **Static testing**, which is used to check the application without executing the code. And we also learn about **static Testing, why we use static Testing, how to perform it, a different technique for static Testing, advantages of static testing, and various Static Testing tools.**

Introduction to Static Testing

Static testing is a verification process used to test the application without implementing the code of the application. And it is a **cost-effective process**.

To avoid the errors, we will execute Static testing in the initial stage of development because it is easier to identify the sources of errors, and it can fix easily.

In other words, we can say that **Static testing** can be done manually or with the help of tools to improve the quality of the application by finding the error at the early stage of development; that is also called the **verification process**.

We can do some of the following important activities while performing static testing:

- **Business requirement review**
- **Design review**
- **Code walkthroughs**
- **The test documentation review**

Note: Static testing is performed in the white box testing phase, where the developer checks every line of the code before giving it to the Test Engineer.

Static testing also helps us to identify those errors which may not be found by **Dynamic Testing**.

Why do we need to perform Static Testing?

We can perform static testing to fulfill the below aspects:

- We can use static testing to improve the development productivity.
- If we performed static testing on an application, we could find the defects in the earlier stages and easily fix them.
- The usage of static testing will decrease the testing cost, development timescales, and time.

What are the different features we can test in Static Testing?

We can test the various testing activities in Static Testing, which are as follows:

- BRD [Business Requirements Document]
- Functional or system Requirements
- Unit Use Cases
- Prototype
- Prototype Specification Document
- Test Data
- DB Fields Dictionary Spreadsheet
- Documentation/Training Guides/ User Manual
- Test Cases/Test Plan Strategy Document

- Traceability Matrix Document
- Performance Test Scripts/Automation

When we performed Static Testing?

To perform static testing, we need to follow the below steps:

Step1: To review the design of the application entirely, we will perform the **inspection process**.

Step2: After that, we will use a checklist for each document under review to make sure that all reviews are covered completely.

We can also implement several activities while performing static testing, which are discussed in the following table:

Activities	Explanation
Architecture Review	<ul style="list-style-type: none"> ◦ The architecture review activities contain all business-level processes such as network diagram, load balancing, server locations, protocol definitions, test equipment, database accessibility, etc.
Use Cases Requirements Validation	<ul style="list-style-type: none"> ◦ It is used to authenticates all the end-user actions along with associated input and output. ◦ If the use case is more comprehensive and detailed, we can make more precise and inclusive test cases.
Functional Requirements Validation	<ul style="list-style-type: none"> ◦ The functional requirement validation activity is used to make sure that all necessary elements identify correctly. ◦ And it also took care of the software, interface listings, network requirements, hardware, and database functionality.
Field Dictionary Validation	<ul style="list-style-type: none"> ◦ In the field dictionary validation, we will test each field in the user interface specified to create field-level validation test cases. ◦ And we can check the fields for error messages, minimum or maximum length, list values, etc.
Prototype/Screen Mockup Validation	<ul style="list-style-type: none"> ◦ The prototype validation activity contains the authentication of requirements and uses cases.

Why we need Static Testing?

We required Static testing whenever we encounter the following situation while testing an application or the software:

- **Dynamic Testing is time-consuming**
- **Flaws at earlier stages/identification of Bugs**
- **Dynamic Testing is expensive**
- **Increased size of the software**

Dynamic Testing is time-consuming

We need static testing to test the application as dynamic testing is time-taking process even though the dynamic testing identifies the bug and provides some information about the bug.

Flaws at earlier stages/identification of Bugs

When we are developing the software, we cannot completely rely on Dynamic testing as it finds the bugs or defects of the application/software at a later stage because it will take the programmer's plenty of time and effort to fix the bugs.

Dynamic Testing is expensive

We need to perform the static testing on the software product because dynamic testing is more expensive than static testing. Involving the test cases is expensive in dynamic testing as the test cases have been created in the initial stages.

And we also need to preserve the implementation and validation of the test case, which takes lots of time from the test engineers.

Increased size of the software

Whenever we test the software, it will increase the size of the software product, which we cannot handle because of the reduction in the productivity of code coverage.

That is why we require static testing to get free from the bugs or defects earlier in the software development life cycle.

Objectives of Static testing

The main objectives of performing static testing is as below:

- Static testing will decrease the flaws in production.
- Static testing will identify, anticipate and fix the bugs at the earliest possible time.
- It is used to save both time and cost.
- It is used to identify defects in the early stage of SDLC, where we can fix them easily.

Some useful points for Successful Static Testing Process

The following guidelines help us to perform a successful static testing process in Software testing.

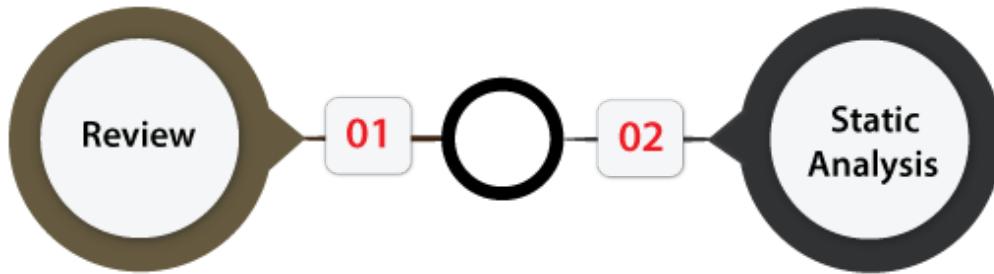
- We can train participants with examples.
- The testing cost and time can decrease if we delete the major delays in test execution.
- We can retain the process formal as per the project culture.
- We can make emphasis only on things that matter.
- As we know that a software walkthrough and review are usually merged into peer reviews; therefore, we can plan explicitly and track the review activities.
- We can resolve the client's problems.

Static Testing Techniques

Static testing techniques offer a great way to enhance the quality and efficiency of software development. The Static testing technique can be done in two ways, which are as follows:

- **Review**
- **Static Analysis**

Static Testing techniques



Review

In static testing, the **review** is a technique or a process implemented to find the possible bugs in the application. We can easily identify and eliminate faults and defects in the various supporting documents such as SRS [**Software Requirements Specifications**] in the **review process**.

In other words, we can say that a **review** in Static Testing is that where all the team members will understand about the project's progress.

In static testing, **reviews** can be divided into **four different parts**, which are as follows:

- **Informal reviews**
- **Walkthroughs**
- **Technical/peer review**
- **Inspections**



Let's understand them in detail one by one:

- **Informal reviews** In **informal review**, the document designer place the contents in front of viewers, and everyone gives their view; therefore, bugs are acknowledged in the early stage.
- **Walkthrough** Generally, the **walkthrough review** is used to performed by a skilled person or expert to verify the bugs. Therefore, there might not be problem in the development or testing phase.
- **Peer review** In **Peer review**, we can check one another's documents to find and resolve the bugs, which is generally done in a team.

- **Inspection** In review, the **inspection** is essentially verifying the document by the higher authority, **for example**, the **verification of SRS [software requirement specifications] document**.

Static Analysis

Another Static Testing technique is **static analysis**, which is used to contain the assessment of the code quality, which is established by developers.

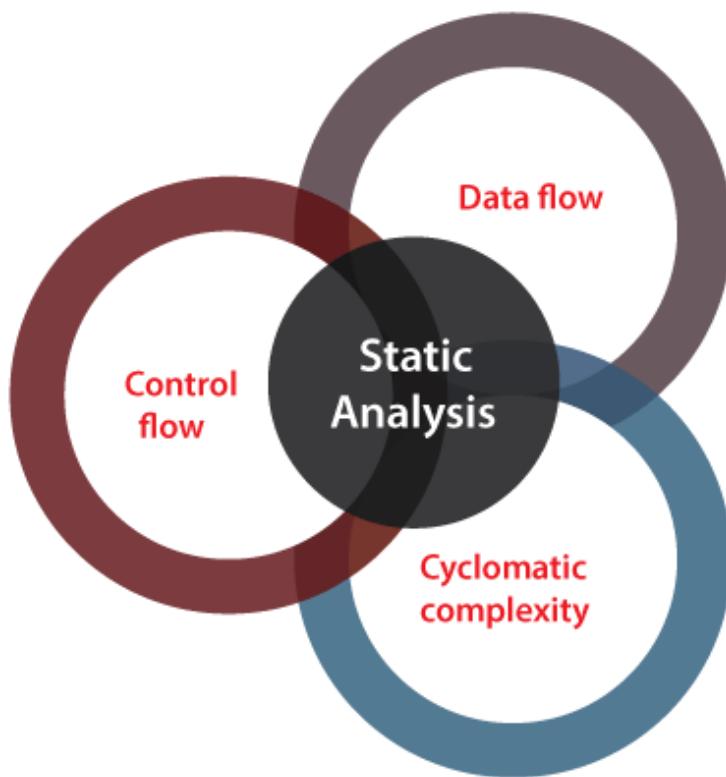
We can use the different tools to perform the code's analysis and evaluation of the same.

In other words, we can say that developers' developed code is analyzed with some tools for structural bugs, which might cause the defects.

The **static analysis** will also help us to identify the below errors:

- **Dead code**
- **Unused variables**
- **Endless loops**
- **Incorrect syntax**
- **Variable with undefined value**

In static testing, **Static Analysis** can be further classified into **three parts**, which are as discuss below:



Data Flow: In static analysis, the data flow is connected to the stream processing.

Control Flow: Generally, the control flow is used to specify how the commands or instructions are implemented.

Cyclomatic Complexity: It is the measurement of the program's complexity, which is mostly linked to the number of independent paths in the control flow graph of the program.

Tools used for Static Testing

In static testing, we have several tools in the market, but here we are discussing the most commonly used tools, which are as follow:

- **CheckStyle**
- **SourceMeter**
- **Soot**

CheckStyle

It is a development tool that is used to help the developers write Java code, which follows a coding standard. The **CheckStyle tool** automates the process of checking Java code.

It is a highly configured tool, which is made to support almost any coding standard. The **Google Java Style, Sun code conventions** are those configuration files, which is supported by CheckStyle.



Feature of CheckStyle

Following are the most common features of CheckStyle:

- It can check various characteristics of our source code.
- The CheckStyle code has the capability to verify the code layout and formatting issues.
- It can also help to identify the method design problems, class design problems.

SourceMeter

It is an advanced tool for the specific static source code analysis of various programming languages such as **C/C++, C#, Java, Python, and RPG projects**.

With the **SourceMeter tool's** help, we can easily identify the vulnerable spots of a system under development from the source code.

The free version with partial functionality of SourceMeter can be accessible for all programming languages.

In SourceMeter, we can use the output of the analysis, the quality of the analyzed source code to enhance and developed both the short and long term in a directed way.



Feature of SourceMeter

The most commonly used features of the SourceMeter tool are as follows:

- It provides the most precise coding error detection.
- The SourceMeter tool will provide a deep static code analysis.
- It improved the user interface with the help of third-party integration.
- It will provide platform-independent command-line tools.

Soot

It is a **Java optimization framework**, which means that it is a framework for analyzing and transforming Java and Android applications where we can test the following aspects:

- Named module and modular jar files.
- Automatic modules, which means the modules are repeatedly created from jars in the module-path.
- Exploded modules
- Resolving modules in Soot's



And a Soot can also produce possibly transformed code in the various output formats such as [Android bytecode](#), [Java bytecode](#) [Jasmin](#), and [Jimple](#).

Advantages of Static Testing

The advantages of static testing are as follows:

- **Improved Product quality** Static testing will enhance the product quality because it identifies the flaws or bugs in the initial stage of software development.
- **Improved the efficiency of Dynamic testing** The usage of Static testing will improve Dynamic Testing efficiency because the code gets cleaner and better after executing Static Testing. As we understood above, static Testing needs some efforts and time to generate and keep good quality test cases.
- **Reduced SDLC cost** The Static Testing reduced the SDLC cost because it identifies the bugs in the earlier stages of **the software development life cycle**. So, it needs less hard work and time to change the product and fix them.
- **Immediate evaluation & feedback** The static testing provides us immediate evaluation and feedback of the software during each phase while developing the software product.
- **Exact location of bug is traced** When we perform the static testing, we can easily identify the bugs' exact location compared to the dynamic Testing.

Overview

In the Static testing section, we have learned the following topics:

- Static testing is used to identify the faults in the early stage of the Software development cycle [SDLC].
- We have understood that Static Testing is not a replacement for dynamic Testing because both testings identify different bug types.
- We have understood the objective of Static Testing.
- In static testing, the reviews are the productive approach to test the application because the reviews help identify the bugs and recognize the design flaws, missing requirements, and non-maintainable code, etc.
- We have understood several static testing tools, which help us enhance testing performance for the software product.

Dynamic Testing

In this section, we are going to understand **Dynamic testing**, which is done when the code is executed in the run time environment.

And we also learn about Dynamic testing, **why we use it, how to perform it, what are a different technique for Dynamic testing, various tools for Dynamic Testing.**

Introduction to Dynamic Testing

Dynamic testing is one of the most important parts of Software testing, which is used to analyse the code's dynamic behavior.

The dynamic testing is working with the software by giving input values and verifying if the output is expected by implementing a specific test case that can be done manually or with an automation process.

The dynamic testing can be done when the code is executed in the run time environment. It is a **validation process** where functional testing [unit, integration, system, and user acceptance testing] and non-functional testing [Performance, usability, compatibility, recovery and security testing] are performed.

As we know that **Static testing** is a **verification** process, whereas **dynamic testing** is a **validation** process, and together they help us to deliver a cost-effective quality Software product.

Why do we need to perform Dynamic Testing?

We can easily understand how to implement dynamic testing during the **STLC [Software Testing Life Cycle]** if we consider the characteristics accessible by dynamic testing.

Using dynamic testing, the team can verify the software's critical features, but some of those can be left without any assessment. And they can also affect the functioning, reliability, and performance of the software product.

Hence, we can perform **Dynamic testing** to fulfill the various below aspects:

- We will perform dynamic testing to check whether the application or software is working fine during and after installing the application without any error.
- We can perform dynamic testing to verify the efficient behavior of the software.
- The software should be compiled and run if we want to perform dynamic testing.
- Generally, Dynamic Testing is implemented to define the dynamic behavior of code.
- The team implements the code to test the software application's performance in a run-time environment during the dynamic testing process.
- It makes sure that the concurrency of the software application with the customer's potentials, needs and the end-user.
- It is an operative technique to measure the effect of several environmental stresses on the software application like **network, hardware**

Characteristic of Dynamic Testing

For understanding the fundamental of the **software testing techniques**, we have to learn their attribute and several other components. Hence, following are some of the important characteristics of **dynamic testing**:

- It is implemented throughout the **validation stage** of software testing.
- Dynamic Testing is done by performing the program.
- Both functional and non-functional testing include in dynamic testing.
- In Dynamic testing, we can easily identify the bugs for the particular software.
- It helps the team in validating the reliability of the software application.

- Unlike static testing, the team implements the software's code to get expected outputs in dynamic testing.
- Dynamic testing is performed directly on the software application as compare to other testing techniques.
- Dynamic testing is a more formal testing approach for different testing activities such as **test execution, coverage consideration, reporting and test case identification**.

Dynamic testing Process

Generally, dynamic testing follows a set process when the approach and test implementation performances are decided, and the team can move to execute the different testing activities.

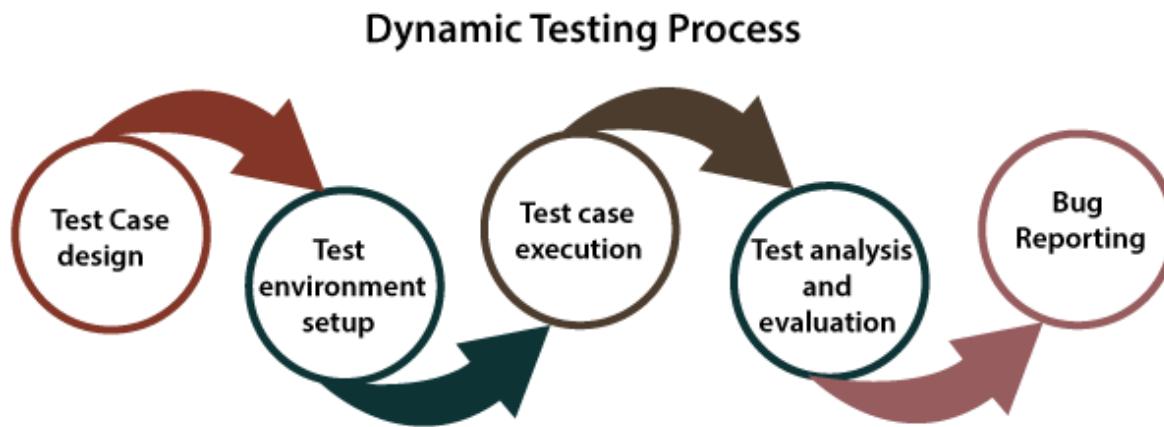
With the help of this process, the team can find any irregularity from the approaches and strategies and help us display all the testing steps.

In the **STLC**, the process of Dynamic Testing involves different functions. And all the functions in the dynamic testing process rely on the conclusion of the earlier task in the testing process.

The Dynamic testing process will complete in the following steps:

- **Test case design**
- **Test environment step-up**
- **Test case execution**
- **Test analysis and evaluation**
- **Bug Reporting**

The actual Dynamic Testing Process begins from Test Case Design in the [software testing](#) life cycle. Now, we discuss each step one by one to get complete knowledge of the dynamic testing process.



Step1: Test Case Design

In the first step of the dynamic testing process, the teams will design the test cases. Here, we are creating those test cases that depend on the requirements and scope of testing established before the start of the project.

In this step, we can **originate the test conditions, obtain the test cases, extract the coverage Items, and identify those features that need to be tested**.

Step2: Environment Setup

In the **test environment phase**, we will make sure that the testing environment should always be parallel to the production environment because the testing is implemented directly on the software product.

In this step, the dynamic testing process's main objective is to install the test environment, which helps us succeed in the test machines.

Step3: Test Execution

Once we successfully install the test environment, we will execute those test cases prepared in the primary stage of the dynamic testing process.

Step4: Analysis & Evaluation

After executing the test cases, we will analyse and evaluate the outcomes derived from the testing. And we will compare those outcomes with the expected results.

If expected and actual results are not the same according to executing, we will consider those test cases as fail, and log the Bug in the bug repository.

Step5: Bug Reporting

After analyzing the test cases, we will be reported and recorded any bugs or defects between the actual result and expected result to the concerned person. And the concerned person will make sure that the issue has been solved and delivering a quality product.

Example of Dynamic Testing

Let us take one sample example where we understand how dynamic testing will work.

So, for this, we will understand the login module of any application, such as **www.Twitter.com**.

Suppose we want to create one new account with a secure password, so we need to follow some pre-defined rules in the **password field**.

And the password should have **eight characters long, capital letters and at least one special character**.

If we are testing this functionality, we would take all the input conditions to test this and then verify the output.

We can also put the non-working constraints, such as input a **4-character password**, and validate if there is an error occurred or not.

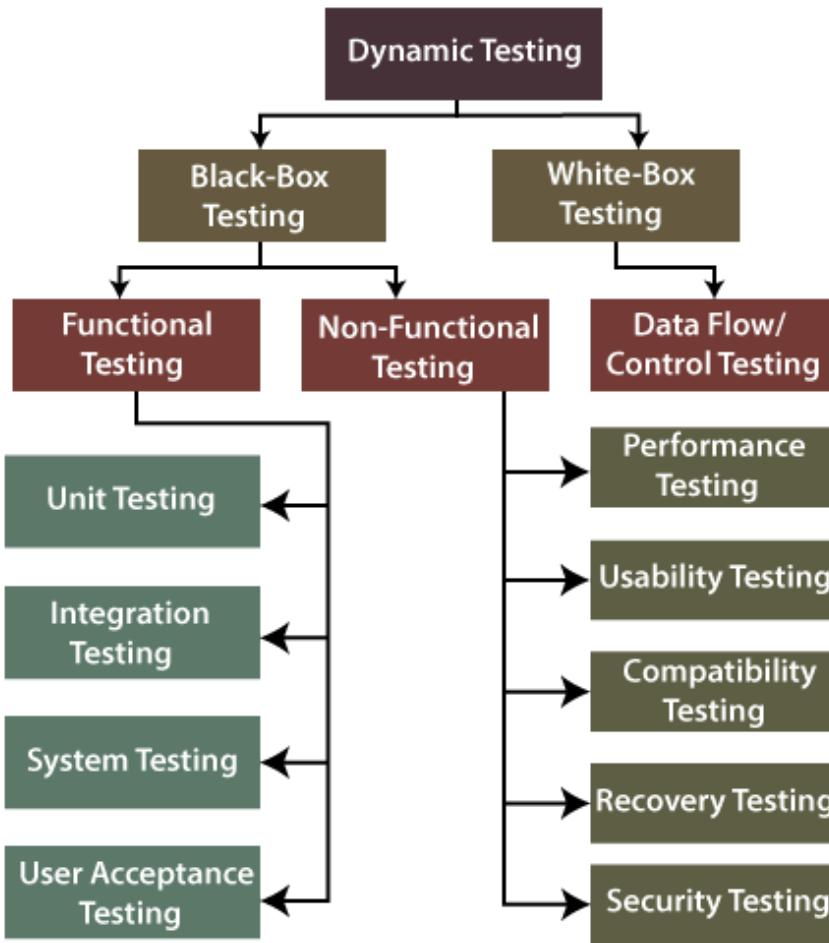
Types of Dynamic testing

Dynamic testing divided into two different testing approach, which are as follows:

- **White-box testing**
- **Black-box testing**

Both the testing techniques will help us execute the dynamic testing process efficiently as they play an important role in verify the performance and quality of the software.

Let's understand them one by one in detail and also see the below diagram of it:



White-box testing

The word **white box** is used to describe the core perspective of the system. The developers will perform the **white box testing**, where they will test every line of the program's code.

When the developers perform the White-box testing and then send the software application to the testing team, the testing team will do the black box testing, validate the application as well as the requirements. The white-box testing is further divided into **data flow/control testing**.

Data flow Testing

The **data flow testing** is used to identify the program's test paths as per the settings of descriptions and uses of variables in the program. And it does not relate to data flow diagrams.

Black-box testing

The **black-box testing** is a testing technique where the test engineer selects a module and gives an input value to observe its functionality and analysis of whether the function is giving the expected output or not. If the function produced the correct output, then the particular function will be marked as pass.

To perform black-box testing, the test engineer should have specific knowledge about the software's requirement rather than programming knowledge of the software.

And then, they can develop the test cases to check the correctness of the software's functionality.

Black-box testing is further classified into two types, which are as follows:

- **Functional testing**
- **Non-function testing**

Functional testing

Functional testing is one of the most important parts of black-box testing. It mainly focuses on application specification rather than the actual code, and the test engineer will test the program rather than the system.

The functional testing is used to validate the software application's functionality, whether the function is working as per the requirement specification.

In functional testing, each module has been tested by giving the value, determining the output, and verifying the actual output with the expected value.

The functional testing is classified into four different type of testing, which are as follows:

- **Unit testing**
- **Integration testing**
- **System testing**
- **User acceptance testing**

Unit testing

- The unit testing is the first level of functional testing to perform any testing on the software application.
- We will perform the unit testing whenever the application is ready and given to the Test engineer. He/she will start checking every component of the module or application independently or one by one. And this process is known as **components testing**.
- The primary objective to perform unit testing is to test the correctness of remote code and validate the unit components with their performance.

Integration testing

- When we have successfully done the unit testing on the specific software, we will go for the integration testing. The integration testing will help us to combined individual units and tested as a group. And it is **the second level** of functional testing.
- When all the components or modules are working independently, we will check the data flow between the dependent modules, which is known as integration testing.
- The developers and the test engineer perform the integration testing. And the main purpose of the integration is to identify the faults in the interaction between the integrated units.

System testing

- System testing is used to check the end-to-end flow of an application or the software as a user.
- System testing is also known as **end-to-end testing** as the testing environment is similar to the production environment.
- In the **third level (system testing)** of functional testing, we go through all the necessary modules of an application and check if the end features or the end business works fine, and test the product as a whole system.

User acceptance testing

- The user acceptance testing is performed to certify the system according to requirements. The customer or client does it before accepting the final product.
- In other words, we can say that the **UAT** is done by the customer (domain expert) for their satisfaction and check whether the application is working according to given business scenarios and real-time scenarios.
- It is the last level of functional testing, which is execute before releasing the software to the market or production environment where two or more end-users will involve.

Non- Functional testing

Another part of black-box testing is **non-functional testing**. It is used to test non-functional constraints like **load test, reliability, performance, and software accountability**.

The main objective of performing the non-functional testing is to test the software system's reading speed according to the non-functional parameters because these parameters are never tested before the functional testing.

Non-functional testing plays a vital role in customer satisfaction while testing the software or the application.

It reduces the risk of production and related costs of the software, and it provides a thorough knowledge of product behavior and used technologies.

Furthermore, the non-functional testing is divided into various parts, which can be performed at the test level.

- **Performance testing**
- **Usability testing**
- **Compatibility testing**
- **Recovery testing**
- **Security testing**

Let's understand them in details one by one:

Performance Testing

- The performance testing is the most importantly used type of **non-functional**
- Once the software is stable and moved to the production, and it may be accessed by multiple users concurrently, we will do **performance testing**.
- The **performance testing** is testing where we check the *behavior of an application by applying some load*.
- As we know it is non-functional testing, which doesn't mean that we always use performance testing when the application is functionally stable; only then we go for performance testing.

Usability Testing

- In usability testing, we will check the user-friendliness, efficiency, and accuracy of the software application.
- If we are using usability testing, it ensures that the developed software is easy to test while using the system without facing any problem and makes end-user life easier.

Compatibility testing

- The next type of **non-functional testing** is **compatibility testing**, which is used to check the functionality of an application on different **software, hardware platforms, network, and browsers**.
- The compatibility testing is not performed for all the applications; we will use the compatibility testing only for those applications where we don't have control over the platform used by users.

Recovery testing

- In **recovery testing**, we can verify how well a system can recover from hardware failures and crashes.
- It reproduced the failure modes or essential producing failures in a controlled environment.
- The recovery testing is performed to confirm that a system is fault-tolerant and can improve well from failures.

Security testing

- The security testing is used to discover the weaknesses, risks, or threats in the software application and help us stop the nasty attack from outsiders and ensure our software applications' security.
- The main purpose of security testing is to identify all the possible uncertainties and vulnerabilities of the application so that the software does not stop working.

Advantages and disadvantages of Dynamic Testing

From detecting and evaluating several bugs and errors in the software to verifying the software's performance, dynamic testing provides several benefits to the users and the testing team.

However, we have various **advantages** of dynamic testing as well as some **disadvantages**.

Therefore, below we listed some of the advantages and disadvantages of dynamic testing:

Advantages

Following are the advantages of dynamic testing:

- It validates the performance of the software application.
- The usage of dynamic testing ensures the reliability and constancy of the software product.
- It can automate with the help of tools that detect the problematic and complex bugs in the testing process, which cannot be covered through static Analysis.
- It helps the testing team to identify the weak areas of the run-time environment.
- The most important benefit of using **dynamic testing** over static testing is the relatively higher number of bugs can be found.
- As compared to **static testing**, **dynamic testing** requires a smaller number of meetings at the planning level of testing.
- It implements the software, end to end, and delivers Bug-free software.
- It becomes an essential tool for identifying any security threats.
- In dynamic testing, we can detect the problematic bugs which may have escaped the review processes.
- It also identifies those bugs which cannot be noticed by static testing.
- Dynamic testing can also find security threats, which ensure a better and secure application.

Disadvantages

Following are drawbacks of dynamic testing:

- It is a **time-consuming** process as it implements the software application or code, which needs a massive resource.
- The dynamic testing process is a **bit costlier** as it increases the budget of the software.
- The dynamic testing needs more human resources to complete the task, which makes its implementation costlier.
- Generally, dynamic testing is executed after the coding phase is completed, and therefore, the bugs are identified later in the life cycle.

Overview

In the dynamic testing section, we have learned the following topics:

- After understanding the dynamic testing above, we can easily say that the importance of dynamic testing is massive in the **software testing life cycle (STLC)**.
- Dynamic testing is used to perform the dynamic behavior of the code.
- We have understood the process of dynamic Testing and the various types of dynamic testing.
- In dynamic testing, we can directly implement the software tests to verify the **functional performance, behavior, reliability, and other significant features of the software**.
- We have understood the **advantages and disadvantages** of dynamic testing.

Non-Functional Testing

Non-functional testing is a type of software testing to test non-functional parameters such as reliability, load test, performance and accountability of the software. The primary purpose of non-functional testing is to test the reading speed of the software system as per non-functional parameters. The parameters of non-functional testing are never tested before the functional testing.

Non-functional testing is also very important as functional testing because it plays a crucial role in customer satisfaction.

For example, non-functional testing would be to test how many people can work simultaneously on any software.

Why Non-Functional Testing

Functional and Non-functional testing both are mandatory for newly developed software. Functional testing checks the correctness of internal functions while Non-Functional testing checks the ability to work in an external environment.

It sets the way for software installation, setup, and execution. The measurement and metrics used for internal research and development are collected and produced under non-functional testing.

Non-functional testing gives detailed knowledge of product behavior and used technologies. It helps in reducing the risk of production and associated costs of the software.

Parameters to be tested under Non-Functional Testing



Performance Testing

Performance Testing eliminates the reason behind the slow and limited performance of the software. Reading speed of the software should be as fast as possible.

For Performance Testing, a well-structured and clear specification about expected speed must be defined. Otherwise, the outcome of the test (Success or Failure) will not be obvious.

Load Testing

Load testing involves testing the system's loading capacity. Loading capacity means more and more people can work on the system simultaneously.

Security Testing

Security testing is used to detect the security flaws of the software application. The testing is done via investigating system architecture and the mindset of an attacker. Test cases are conducted by finding areas of code where an attack is most likely to happen.

Portability Testing

The portability testing of the software is used to verify whether the system can run on different operating systems without occurring any bug. This test also tests the working of software when there is a same operating system but different hardware.

Accountability Testing

Accountability test is done to check whether the system is operating correctly or not. A function should give the same result for which it has been created. If the system gives expected output, it gets passed in the test otherwise failed.

Reliability Testing

Reliability test assumes that whether the software system is running without fail under specified conditions or not. The system must be run for a specific time and number of processes. If the system is failed under these specified conditions, reliability test will be failed.

Efficiency Testing

Efficiency test examines the number of resources needed to develop a software system, and how many of these were used. It also includes the test of these three points.

- Customer's requirements must be satisfied by the software system.
- A software system should achieve customer specifications.
- Enough efforts should be made to develop a software system.

Advantages of Non-functional testing

- It provides a higher level of security. Security is a fundamental feature due to which system is protected from cyber-attacks.
- It ensures the loading capability of the system so that any number of users can use it simultaneously.
- It improves the performance of the system.
- Test cases are never changed so do not need to write them more than once.
- Overall time consumption is less as compared to other testing processes.

Disadvantages of Non-Functional Testing

- Every time the software is updated, non-functional tests are performed again.
- Due to software updates, people have to pay to re-examine the software; thus software becomes very expensive.

Stress Testing

In this section, we are going to understand **Stress Testing**, which is an important part of **Performance testing** and used to check the behavior of an application by applying a load greater than the desired load.

And we also learn about **its process, why we need to perform the Stress testing, the objective of Stress testing, examples, various features of stress Testing, advantage and disadvantage.**

Introduction of Stress Testing

In software testing, stress testing is an important part of performance testing under non-functional testing.

Stress Testing is testing used to check the accessibility and robustness of software beyond usual functional limits. It mainly considers for critical software but it can also be used for all types of software applications.

It is also known as **Endurance Testing, fatigue testing** or **Torture Testing**.

The stress testing includes the **testing beyond standard operational size**, repeatedly to a **breaking point**, to get the outputs.

It highlights the error handling and robustness under a heavy load instead of correct behavior under regular conditions.

In other words, we can say that **Stress testing** is used to verify the constancy and dependability of the system and also make sure that the system would not crash under disaster circumstances.

To analyze how the system works under extreme conditions, we perform **stress testing** outside the normal load.

The Objective of Stress Testing

The main objective of using stress testing is to fulfill the following aspects:

- The primary purpose of executing the stress testing is **to confirm that the software does not crash in lacking computational resources like disk space, memory, and network request.**
- The implementation of stress testing certifies that the system fails and improves effortlessly, known as **the recoverability process.**
- We can use stress testing to discover hardware issues, data corruption issues.
- Stress testing will help us to identify the security weaknesses that might sneak-in throughout constant peak load.
- It helps determine the software application's data integrity throughout the extreme load, which implies that the data should be in a dependable state after a failure.

Features of Stress Testing

Following are the basic features of stress testing:

- Stress testing also makes sure that unpredicted failures do not cause security issues.
- It is used to analyze the system works under rare circumstances and the system's behavior after a failure.
- Stress testing is used to check the system has saved the data before crashing or not.
- Stress testing guarantees to display a suitable error message when the system is under stress.

Why do we need to perform the Stress Testing?

We need to perform stress testing if we encounter the following situations:

Whenever **e-commerce or online shopping sites** announce a sale during the festival may witness a spike in traffic. Or when an article is mentioned in a top newspaper, its knowledges an unexpected flow in traffic.

If we fail to assist this sudden traffic can result in loss of profits and status. So, in that case, we need to execute the **Stress Testing to integrate such irregular traffic spikes**.

Stress testing is also needed to be performed for the below scenarios:

- When the system is under stress, it should display a proper error message.
- To inspect whether the system works under bizarre circumstances.
- If the system is failed under risky situations could result in huge profits loss.
- By implementing Stress Testing, we will be prepared for extreme conditions.

Example of Stress Testing

Let's see some **real-time examples** where we can discover the usage of Stress Testing.

Example 1: E-commerce website/ application

Throughout the new product releases, sales, holidays, promotions, and festival offers, the e-commerce application tends to get many users in a very short time.

Example 2: News website at the time of some major/viral event

The news website will crash or slow down when a major event happens; **for example**, when Michael Jackson passed away, a maximum number of news websites are slow down, or some of them also crashed.

To overcome this situation, we need to perform stress testing on the particular application and be prepared to recover quickly in case of any crashes or failure.

Example 3: Education Board's result website

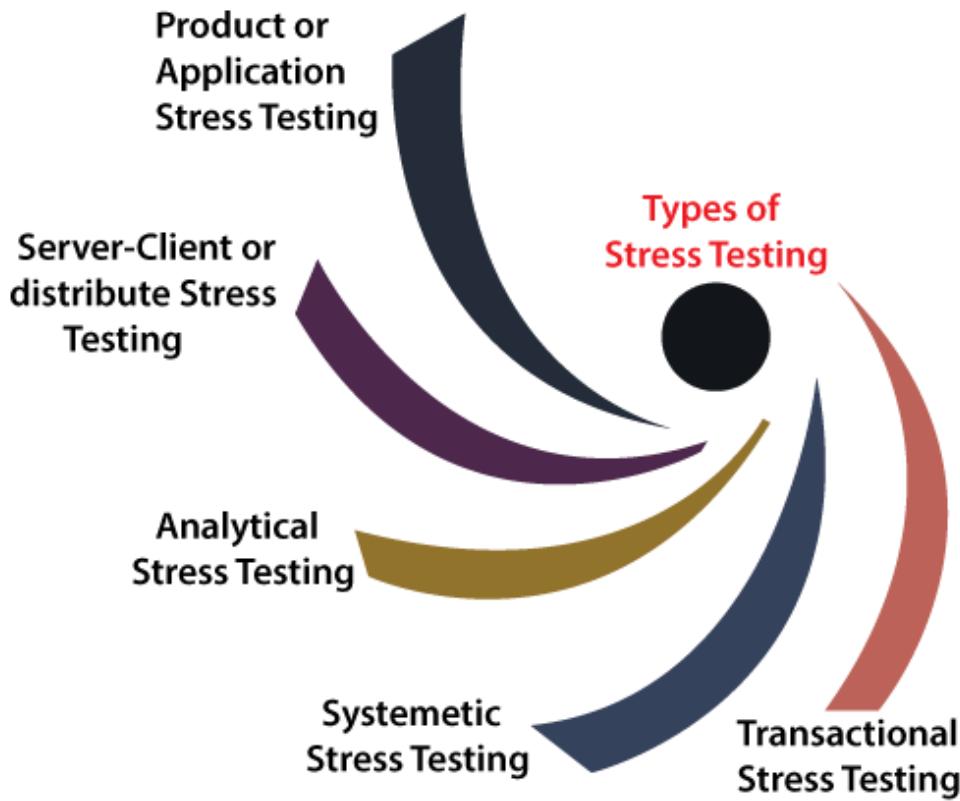
Stress testing is important to perform on the education board's result website. On the day of some results, many students, users, and applicants will logins to the particular to check their grades.

Therefore, the execution of stress testing helps identify the failure of the application and evaluate the performance and recoverability when the ultimate load occurs in a short duration or when the result is out.

Note: Stress testing of the website or an application is very significant to handle such an unexpected increase in several visitors or users.

Types of Stress Testing

Stress testing can be categories into various parts, which are as follows:



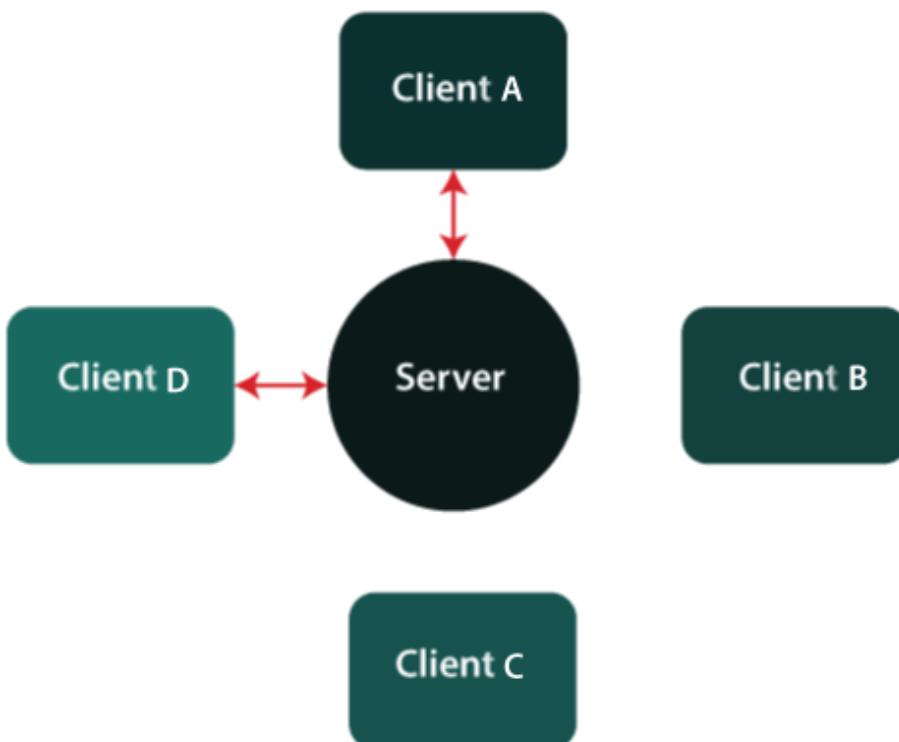
- **Product or Application stress testing**
- **Server-client or distribute Stress Testing**
- **Analytical Stress Testing**
- **Systematic Stress Testing**
- **Transactional Stress Testing**

Product or Application Stress Testing

- The application or product stress testing is mainly focused on determining the faults related to network issues, data locking, blocking, and a performance bottleneck in a software product.

Server-client or Distribute Stress Testing

- In this type of stress testing, all the clients related to the server are tested.
- The distribute stress testing is used to perform across all clients from the server.
- The server can communicate with **clients A and B**, but it cannot link with **clients C and D** when there is stress on the client-server system as we can see in the below image:



Analytical/ Exploratory Stress Testing

- Analytical or exploratory testing is used to execute the system with unusual constraints unlikely to occur in a real scenario.
- It is mainly used to identify the bugs in rare situations such as **a large number of users logged simultaneously or a database went offline when retrieved from a website.**

Let see some examples **of analytical Stress Testing** where such irregular conditions are used:

- When a large number of parallel users try to log into the application.
- Data is added in enormously large quantity in the database.
- When the website tries to reach it from the front end, and database linked to the website shuts down.

Systematic Stress Testing

- It is combined testing used to execute the test across various systems running on a similar server.
- Using **systematic stress testing**, we can easily detect the bottleneck where data of one application blocks another application.

Transactional Stress Testing

- Another type of stress testing **is transactional stress testing**, which is used to implement one or more transactions between various applications.
- The main objective of performing the transactional stress testing is **to enhance the system performance.**

Process of Stress testing / how to perform stress testing

The stress testing process will be completed into the following steps:



Step1: Detect the testing environment

In the first step of stress testing, we will identify the network, software, and hardware configurations and tools available to achieve the stress test.

Step2: Find performance acceptance criteria

After identifying the testing environment, we will find the performance acceptance criteria, which help us categorize the metrics used to test the application's performance under stress.

And also, identifying the success criteria for a stress test, for example, the maximum load can apply to the application for it to fail.

Step3: Plan and design stress tests

In the next step of the stress testing process, we will plan and design a stress test plan, identify test scenarios etc.

Step4: Configure the test environment

Once the stress test plan has been created successfully, we will move to our next step where we create the test environment, tools and resources essential to perform each approach as features and components become available for test.

Step5: Implement test design

After the test environment's configuration, we will develop the stress tests resulting the test design best performs.

Step6: Execute tests

In the next step, we will execute the particular test, observe and confirm the tests along with test data and output collection.

Step7: Analyze the results

In the last step of the stress testing process, we will analyze the outcomes, combine and share the respective teams' output data.

Stress testing tools

As we know that **stress testing** is part of **performance testing**, the tools used for performance testing can be used for stress testing. Therefore, we have various types of **Stress testing** tools available in the market, where some are commercial tools and open-source tools. Some of the most commonly **Stress testing** are listed below:

- **Apache JMeter**
- **NeoLoad**
- **Stress tester**
- **LoadRunner**

To get detailed information about the above Stress testing tool, refer to the below link:

<https://www.javatpoint.com/performance-testing-tools>

Advantages and disadvantages of Stress Testing

Advantages

Some of the vital **benefits** of performing Stress testing is as follows:

- Stress testing signifies the system's behavior after failure and makes sure that the system recovers quickly from the crashes.
- The most important advantage of executing the stress testing will make the system work in regular and irregular conditions in a suitable way.
- It determines the scalability and enhance the performance of the software.

Disadvantages

Some of the most common **drawbacks** of Stress testing are as follows:

- Even in open-source tools like JMeter, a load testing environment is required, which should be as close to the production environment setup as possible.
- If we are writing the Stress test script, the person should have enough scripting knowledge of the language supported by the particular tool.
- If we are using stress testing, it will require additional resources, which makes this testing bit costlier.
- If we perform the **Stress Testing** manually, it became a tedious and complicated task to complete, and it may also not produce the expected results.

Overview

In this tutorial, we have understood that **stress testing** is used to assess the system under extreme situations. It can verify the system to recover back to normal status.

It is a type of **non-functional testing** and generally executed after the **functional testing**.

Stress testing entirely concentrate on testing the system under extreme load situations to detect its **breaking point** and see if suitable messages are shown when the system is not responsive.

Recovery testing

In this section, we are going to understand **recovery testing**, which is an important part of **Non-functional testing** and used to test how soon the application is recovered from the hardware crashed or fails.

And we also learn about its **process, why we need to perform the recovery testing, who perform the recovery testing, example, advantage and disadvantage**.

Introduction of Recovery Testing

Recovery testing is testing where the test engineer will test the application to check how well the Software or the application recovers from disasters or crashes.

It is one of the most important types of **non-functional testing** categorized under performance testing.

In other words, we can say that the recovery testing is done to verify how fast and better the application can improve or learn the capability of the software after it has gone through any **software, hardware crashes or network failures** etc.

It is the software's required **failure** in a diversity of ways to confirm that recovery is properly performed.

While executing the recovery testing, we should first take the backup and save it to a secured location to keep away from any data loss if the data is not recovered successfully.

The software/ hardware is forcefully failed to verify the following aspects while executing the recovery testing:

- Lost data can be retrieved completely or not.
- It fails to verify that the fraction of scenarios in which the system can recover back.
- It is failing to verify that if any other additional operations of the software can be done or not.

Note: A good software application is the one that recovers in no time from crashes, hardware failures or other such types of failures.

Example of Recovery testing

Let us see a scenarios where we can understand how the recovery testing is performed:

Scenario 1

Suppose we are using the browser, let say **Google Chrome**, and the power goes off. When we switch on the system again and re-open Google Chrome, we get a message window that displays whether we want to start a new session or restore the previous session.

So, in this situation, while we are restarting the system while a browser has a definite number of sessions and check if the browser can recover all of them or not.

Some of the most common failures which need to test for recovery:

Here, we list out some of the most frequent failures which needs to test, while performing the recovery testing:

- **External device not responding**
- **Network issue**
- **Wireless network signal loss**
- **Power failure**
- **Server not responding**
- **Database overload**

- **External server not reachable**
- **DLL file missing**
- **Physical conditions**
- **Stopped services**

Why recovery testing is important?

The recovery testing is significant if we are developing the application for a user who will decides the difference between success and failure for our organization. Therefore, we need to develop software, which has enough consistency and recoverability.

When do we need to perform the recovery testing?

- Whenever we have a release or upgrade, we need to review the system-specific recovery testing.
- To perform recovery testing, we need to ensure that everyone is included in each other's roles and responsibilities.
- Critical business functions can normally run when there is a failure or disaster.

Who implements Recovery testing?

The Recovery testing is a part of **Business Continuity Planning (BCP)** and involves a host of roles. The following concerned persons can perform the recovery testing:

- It can be executed by the **Production Service and Service Management teams** who have the knowledge and experience in managing such outages.
- The **Technical SMEs** recognize the core systems maintained by the hardware.
- The recovery testing can perform by the **IT operations teams** which manage servers and other hardware infrastructure.

Recovery testing life cycle

The recovery testing life cycle includes the various phase, which is as follows:

- **Standard operations**
- **Disaster and failure occurrence**
- **Interruption to Standard Process**
- **Recovery Process**
- **Rebuild Process**

Let's understand them one by one in details:



- **Standard Operations**

In the standard operations phase, we will establish the system according to the Software and hardware requirements, where the particular system can execute as expected. It is used to define the way system is planned to work.

- **Disaster and failure Occurrence**

In the next phase of recovery testing, we can identify the several failures of the system and those failures are as follows:

- **Power failure**
- **Hardware failure**
- **Physical conditions**
- **Server not reachable and many more.**
- **Interruption to Standard Process**

It leads us to losses in terms of business, financial losses, relations with the client, reputation in the market, etc.

- **Recovery Process**

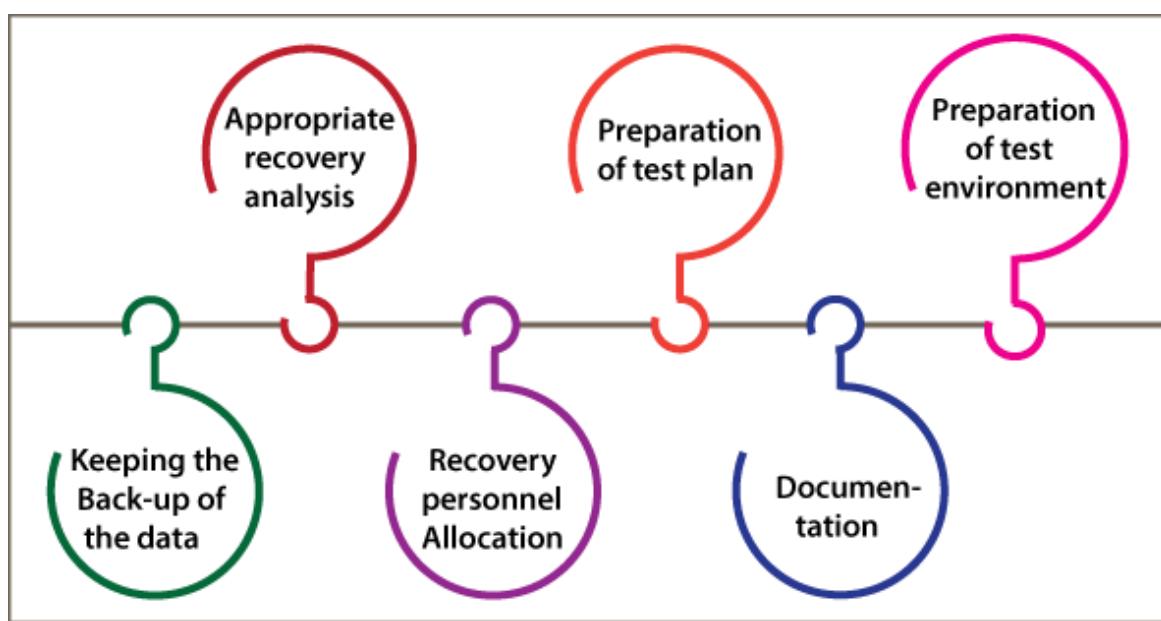
In recovery testing, the **recovery process** is used to keep away from the most important losses in companies and have backup plans with minimal impact on the interruption system.

- **Rebuild Process**

The last phase of the recovery testing process is the rebuild process, which contains the already specified documents and processes that have to be followed. And in this phase, all configuration files and folders are restored to retrieve the lost data.

Steps to be performed before implementing a Recovery Testing:

The following steps need to be performed before executing the recovery testing process to ensure the performance of recovery testing:



Step1: Appropriate recovery analysis

Before implementing the recovery testing, we should make sure that the proper analysis has to be done to verify the possibility of recovery. The recovery analysis is necessary for our better understanding of the recovery-related modification, which can impact the system's working.

To complete the appropriate analysis, we can observe the below aspects:

- The system's capability to allocate additional resources such as servers in case of critical failures or additional CPUs.
- How to track the failures.
- The effect of the failures, failures that can occur, and solutions to the failure.

Step2: Preparation of Test Plan

In the next step, we will prepare the Test cases according to the analysis results, which we discussed in the above step.

Step3: Preparation of Test environment

After preparing the test cases, we moved to our next step to design the test environment as per the **recovery analysis results**.

Step4: Keeping the Back-up of the data

After the preparing the test cases and test environment, we can go to our next step to keep the back-up data related to the software, for example, **several states of the software and database**.

Similarly, if the data is significant, then **we can keep the backup data depending upon the criticality, with the help of the below strategies:**

- The backing up of the data at one or various locations.
- Single back up or Multiple back-ups.
- Automatic set up for back up at every **n** minute, for example, 20 minutes.
- Online and Offline backups.
- To perform and track the backups, we can have a separate team.
- Allocation of resources for recovery testing.

Step5: Recovery personnel Allocation

Once the backup of data is sustained successfully, we will have enough knowledge for the recovery testing being conducted and allocate the recovery personnel.

Step6: Documentation

In the last step, we will document all the steps performed before and throughout the recovery testing because, in case of a failure, the system can be tested for its performance.

Advantages and Disadvantages of Recovery testing

Advantages

Some of the major **benefits** of performing the recovery testing are as follows:

- One of the most important advantages of **recovery testing** is to recovers system quality because whenever the bugs are detected and fixed, it improves the system's quality.
- The recovery testing will help us to remove risks.
- It helps us to decreases the risk of failure of the software product in the market.
- When we implement the recovery testing, we can easily identify the performance-related issues and fix them before the software product goes live in the market.
- The system is more stable, reliable and bug-free once we performed the recovery testing.

Disadvantages

Following are the **disadvantages** of recovery testing:

- Recovery testing is a **time-consuming process** as the test cases are random.
- It is an expensive process.

- A skilled person is essential to test the recovery testing; if the untrained test engineer implements the recovery testing, they should have all the data for testing: data and backup files.
- In recovery testing, we may not detect all the potential bugs in a few cases because sometimes the issues are unpredictable.

Overview

In the above **recovery testing** tutorial, we have understood the numerous recovery testing characteristics, which helps to learn whether the system or program meets its requirements after a failure.

Recovery testing is necessary if we have any project that overhauls business processes or technology. In case of failures it helps us to validate that recovery.

The occurrence of executing **recovery testing** is in reverse proportional to the impact of failure on the system. Therefore, recurrent testing plays an important role in minimizing the impact.

As we know that the failures can happen anytime due to many common reasons, such as **recovery testing removes critical bugs**, which makes the system ready to recover from those failures.

Exploratory Testing

In this section, we will learn about exploratory testing, its types, when we use it, the advantage and disadvantages of it.

What is exploratory testing?

If requirement does not exist, then we do one round of exploratory testing.

So, for this first, we will be exploring the application in all possible ways, understanding the flow of the application, preparing a test document and then testing the application, this approach is known as exploratory testing.



When we use exploratory testing

We will use this testing for the following aspects:

- When the requirement is missing
- Early iteration is required
- The testing team has the experienced testers when we have a critical application, and new testers entered into the team.

For example, to test any software or the application, first, we will perform unit, integration, and **system testing**.

So if we want to understand any application first, we will perform unit or component testing, suppose the application is having a login page having many elements, and we will understand each part and doing the component testing, but actually, we are doing the exploratory testing because we are exploring the application.

Suppose we have many modules in the application, and we are trying to do some integration scenarios.

Indirectly we are just doing exploratory testing while performing the **integration testing**.

And, even if we are performing system testing, indirectly, we are performing exploratory testing because here we are also understanding and exploring the application.

Why the requirement is missing

The requirement is missing because of the following reasons:

If the project is very old, the test engineer can't understand each scenario from the starting, and it might happen that the requirements will be missing.

For example, in each company, we don't see any fast process which means, we cannot expect the release to be done in just one month, and the product should be delivered in very less duration of time.

Many companies are still in the development phase for a particular product from the last 6 to 12 years.

Suppose one company has a 15-year-old project, and they hired a new test engineer now. The new test engineer faces many difficulties in understanding every scenario or requirement from scratch or starting because he/she is new with the application.

In that case, what test engineer will do with software that is 15 years old?

So firstly, he/she will take the application and start exploring the application. Once the test engineer starts using the application, he/she will get to know how the application is working. And, this process is nothing but exploratory testing.

How to perform exploratory testing

To perform exploratory testing, first, we will start using the application and understand the requirement of the application from the person who has a good product knowledge such as senior test engineer, and developers.

Then we will explore the application and write the necessary document, and this document is sent to the domain expert, and they will go through the document.

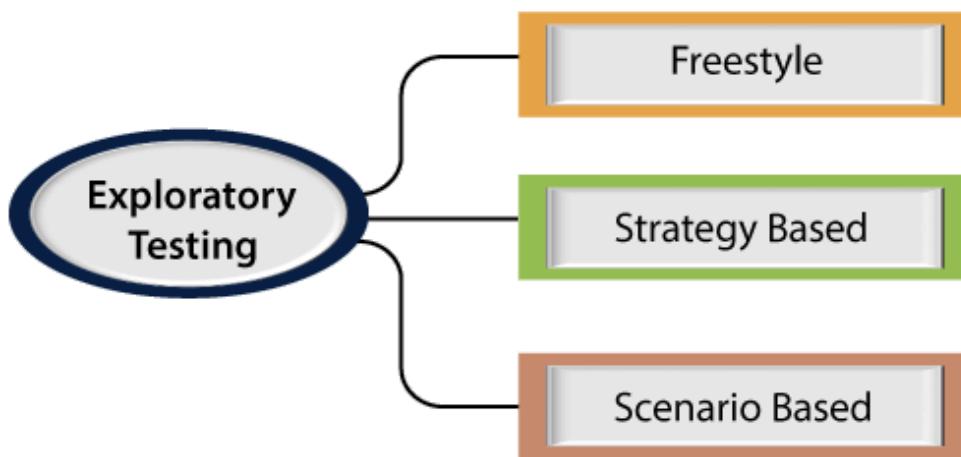
And we can test the application based on our knowledge, and taking the help of the competitive product, which is already launched in the market.

Types of exploratory testing

Exploratory testing can be divided into three parts, which are as follows:

- **Freestyle**
- **Strategy based**
- **Scenario-based**

Types of Exploratory Testing



Freestyle

In freestyle testing, we did not follow any rules, there is no maximum coverage, and we will explore the application just like Adhoc testing.

If we want to get friendly with the software and checks the other test engineer's works, we can use freestyle exploratory testing.

Strategy based

Strategy based exploratory testing can be performed with the help of multiple testing techniques such as risk-based, boundary value analysis, and equivalence partitioning.

It is done by the experienced tester and who is using the application for the longest time because he/she is known the application very much.

Scenario-based

Scenario-based exploratory testing is performed with the help of multiple scenarios such as end-to-end, **test scenarios**, and real user scenarios.

The test engineer can find defects and also checks various set of possibilities for the multiple scenarios with their application knowledge while they were exploring the application.

Advantages and Disadvantages of Exploratory Testing

Advantages

Following are some benefits of exploratory testing:

- If the test engineer using the exploratory testing, he/she may get a critical bug early because, in this testing, we need less preparation.
- In this testing, we can also find those bugs which may have been missed in the test cases.
- This testing can be used to test the new features, whereas, for the existing functionality, we will use the regression testing if we have less time to test the application.
- For the test engineer, this testing requires a lot of concentration to explore the application.

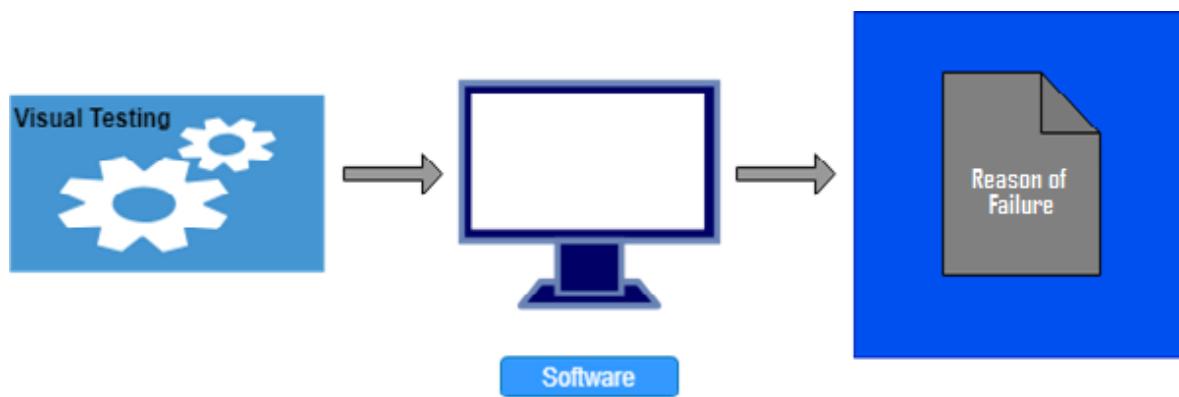
Disadvantages

Following are the disadvantages of exploratory testing:

- **Time Consuming** It is a time taking process because we don't know the requirement, and which feature has to be tested first because we are just exploring the application.
- **The test engineer will misunderstand the feature as a bug. For example**, suppose we have one login page and requirement says we have to provide the necessary details like **username**, **password**, and **employee id** then click on the **login** button. But while performing exploratory testing, we only provide the details of **username**, **password**, and then click on the **login** button, but we have not entered the employee id. Since we don't have the requirement, and doing exploratory testing, that's why we feel that the employee id component is a bug, but it is a feature.
- **Bugs can be misunderstood as a feature For example**, suppose we have one registration page where we have to provide details like the **username**, **password**, **mobile number**, and the **email id**. And the requirement says that when we are providing the mobile number and email id, the same code will be sent to the registered email id and mobile number to verify whether it is correct or not. But when we are performing exploratory testing on the registration page and provide all the details (user name, password, mobile number, and email id), the code will only be sent to our mobile number, not to the email id. It is happening because the requirement is missing, and we will be misunderstood that this bug is a feature, and we never come to that this is a bug.

Visual testing

Visual testing is used to examine what happened at the point of software failure by defining the data in such a way that the developer can quickly identify the reason of failure, and the information is expressed clearly so that any other developer can utilize this information.



Visual testing aims to show actual problem rather than just to describe it, remarkably it increases understanding and clarity so that the problem can be solved quickly.

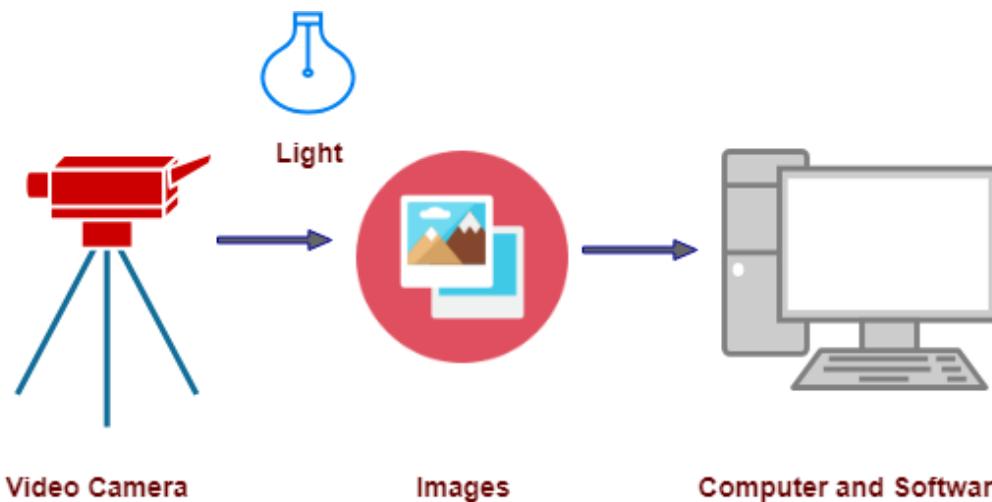
The general meaning of visual is optical means what we can see. Therefore, visual testing requires the video recording of the entire process. It captures everything that happens at the time of system testing in video format. Tester gives a picture in a picture webcam and audio commentary from microphones as an input value.

Visual Inspection System

System for visual inspection consists of a high-quality video camera for collecting data and software and computer to analyze data. The video camera is used to capture a picture of the object during the testing process. These object pictures are sent to a computer via a frame grabber.

The computer has software that analyze the pictures and decide whether the object fails or passes the inspection.

The conditions under which the video testing system works necessarily well controlled and easy to maintain testing persistence.



Visual Inspection System

Visual testing offers a number of advantages. It increases the quality of communication drastically because testers can optically present the problem to the developer as opposed to describing it in written document form. The developer has all the required evidence of a test failure so, the focus is only on the cause of the failure and how to fix it.

Some remarkable advantages and disadvantages are given below:

Advantages of Visual Testing

- Visual testing is cheap because information is recorded in video form. So, we don't need to replicate the information in any other form. It saves money.
- Visual testing provides portability. A tester can provide video to any other tester if the type of software is same. So, in the case of system failure, we don't loss the data.
- Visual testing saves the time of testing as if once the testing process is done and saved in visual form so, we do not need to test software again. Developer can identify the defect by seeing the video.
- Visual testing requires the minimum number of special skills.
- Visual testing requires minimum part preparation because there is a need to find only the reason for system failure.

Disadvantages of Visual Testing:

- Visual testing is suitable only for the surface which can be visible so, need to arrange suitable surface.
- Visual testing cannot detect hidden defects; it can detect only larger defects.
- To record clear visible video lighting must be well implemented.
- It follows only rules does not emulate human inspections.
- Scratches and cracks can create misinterpretation.
- Visual testing does not provide component variations on the product if there is variation in software components it cannot be tested via visual testing.

Summary

Visual testing is used when we test software with easily detectable defects and do not allow component variations.

Acceptance testing

Acceptance testing is formal testing based on user requirements and function processing. It determines whether the software is conforming specified requirements and user requirements or not. It is conducted as a kind of Black Box testing where the number of required users involved testing the acceptance level of the system. It is the fourth and last level of software testing.



User acceptance testing (UAT) is a type of testing, which is done by the customer before accepting the final product. Generally, UAT is done by the customer (domain expert) for their satisfaction, and check whether the application is working according to given business scenarios, real-time scenarios.

In this, we concentrate only on those features and scenarios which are regularly used by the customer or mostly user scenarios for the business or those scenarios which are used daily by the end-user or the customer.

However, the software has passed through three testing levels (Unit Testing, Integration Testing, System Testing). But still there are some minor errors which can be identified when the system is used by the end user in the actual scenario.

Acceptance testing is the squeezing of all the testing processes that have done previously.

Note:

It is done in the separate environment at the customer place, which is known as the UAT environment. The user acceptance testing is done by a different team called as domain expert who is known to the application.

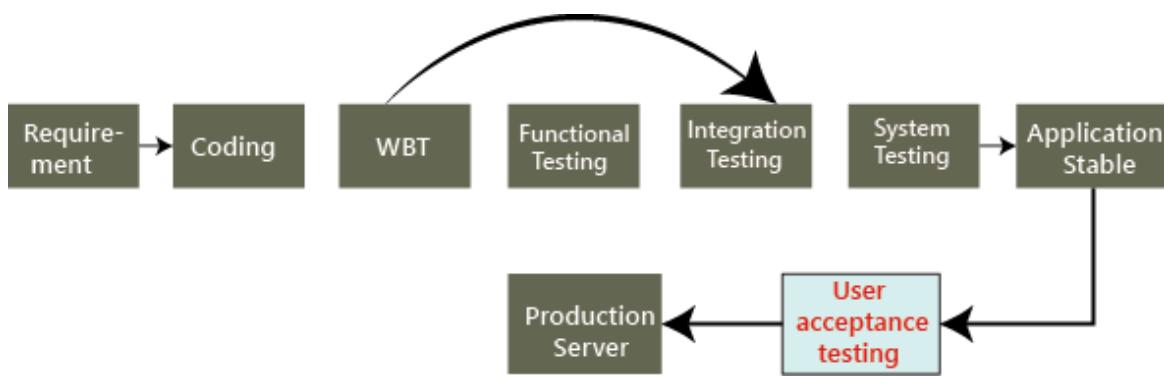
Generally, small companies do not have a domain expert because there is no frequent changes happen in the application.

Reason behind Acceptance Testing

Once the software has undergone through Unit Testing, Integration Testing and System Testing so, Acceptance Testing may seem redundant, but it is required due to the following reasons.

- During the development of a project if there are changes in requirements and it may not be communicated effectively to the development team.
- Developers develop functions by examining the requirement document on their own understanding and may not understand the actual requirements of the client.
- There's maybe some minor errors which can be identified only when the system is used by the end user in the actual scenario so, to find out these minor errors, acceptance testing is essential.

Note: Once we collect the requirement from the customer and done the coding process completely then the test engineer starts all different types of testing until the application becomes stable.

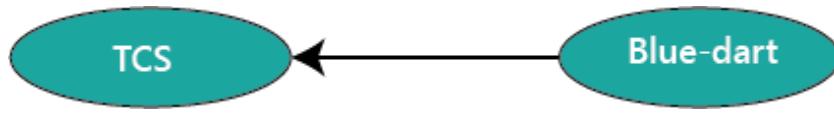


Once the application is bug-free, we handover it to the customer, no customer accept the application blindly before using it. Hence, they do one round of testing for their satisfaction, which is known as user acceptance testing.

Who performs user acceptance testing?

The acceptance testing can be performed by different persons in different cases.

For example, the blue-dart company gives the requirement to TCS for developing the application, and the TCS will accept the needs and agree to deliver the application in the two releases as we can see in the below image:



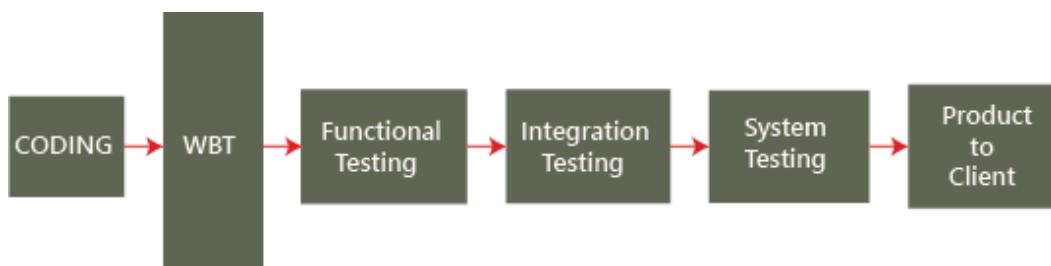
On August 10, the test manager tells the project manager that there is a critical bug in the application, and that will take another four days to fix it.

Jan 2019 ————— 30crores ————— Aug 2019 ————— Aug 2019 ————— 15crores ————— Jan 2020

But the project manager said we have to deliver the software within a given time. It takes another 30 days to fix the defect, or otherwise, we will have to pay the penalty (fine) for each day after the given release date. Is this the real situation? NO, let us see three different cases and understand who perform the acceptance testing.

Case1

In this, we will discuss how the acceptance testing is performed, and here the test engineer will do the acceptance testing.



Mostly, the actual flow for testing the application will be seen in the above image, but here it is little difference, as we know where the end-to-end testing or system testing ends and the acceptance testing will proceed. To understand this scenario, follow the below process:

The blue-dart provides the requirements, and TCS develops the application and performs all the testing and handover to the blue-dart company.

Now the question arises the blue-dart will use the application as soon they get it from TCS? NO, the blue dart company has a group of test engineers after they get the software, and this team will start testing the application, and this end-to-end testing is done at the customer environment, which is called the **User Acceptance Testing**.

Let us see the difference between **TCS test engineers** and **Blue-dart Engineers**:

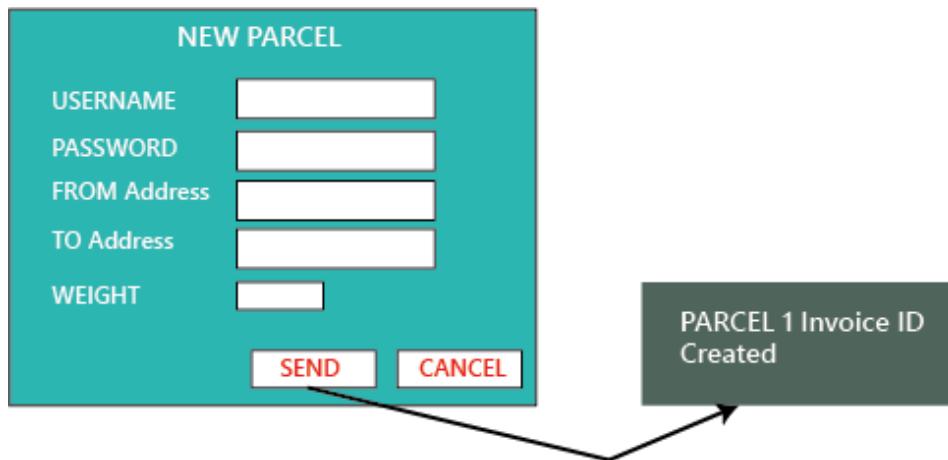
In **TCS**, the tester will perform the **functional testing, integration testing, and system testing** and whereas in **Blue-dart**, the tester will do only the **end-to-end or system testing, which is known as acceptance testing**.

The difference between end-to-end testing at TCS and blue-dart is as follows:

- The blue-dart test engineer is the one who gave the requirements
- The blue-dart engineer understands the product well
- The blue-dart engineer is a domain expert.
- They test the real-time data on the application.

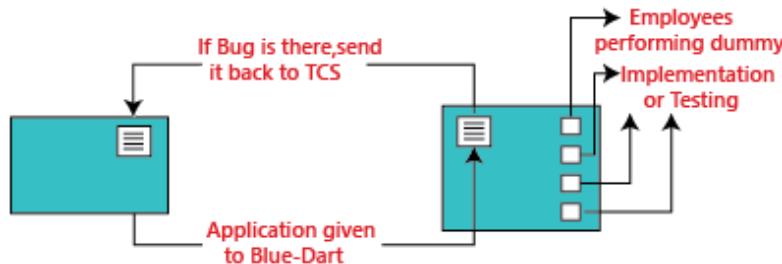
To understand this, we can see the below example, or if we have the application format is like this:

When the application is given to blue-dart test engineers, and they will perform testing and the application should generate a text message "**Parcel 1 invoice Id created.**" It was not mentioned in the requirement, or it is there, and TCS does not fix it. Then fine(penalty) counts for TCS from that only, and whereas the test engineers at TCS will not know this, due to that, we can see the difference between the testing done at TCS and Blue-dart.



Case2

In this case, we will see how the Employee is becoming end-users and performing acceptance testing.



The application is developed and tested at the TCS environment and then sent to blue-dart. And in the Blue-dart, they have fewer test engineers, so they can't do acceptance testing. So for this, out of 300 employees of blue-dart, they will provide the application to the 30 employees and install the application to their systems and ask them to start using the application and find any defect or issues.

Now 30 employees will do the dummy implementation, which means they provide the data into the application and also write that data manually. And here, the employee becomes the end-user and also identify the bugs and issues while using the application.

These issues are verified against the requirements, and now the fine is charged for TCS (sometimes the penalty is charged on an hourly basis). If the identified bug is not as per requirement, then blue-dart can go for the **Request For Enhancement [REF] and Change Request [CR]**.

Where **Request for enhancement** means that if the blue-dart feels that a particular module can be improved and developed in a better way, and then they can send the **Customer Requirement Specification [CRS]** as REF and TCS will follow the CRS and also make sure to do the necessary changes.

And the **Change Request** means, if the requirement has not been specified accurately, then blue-dart provides the exact needs and Request for changes.

Therefore, the acceptance testing can also be defined as end-to-end testing, which can be done by the engineers who are working in the client environment. Here, they take real-time scenarios and check whether the application is working fine or not, and also we can make real-time business scenarios because the end-user knows how the business flow works.

Note:

If we are getting more builds for acceptance testing, this means that:

- After receiving the application, the customer is getting more and more ideas, so they are asking for more and more changes.
- The quality of the software, which we delivered to customers, is not appropriate, and the development and testing both are not correctly done.
- The requirement which was given in the starting is not clear.

Case3

In this case, if the blue-dart customers become the end-users.

Here, the application is developed and tested and implemented at a blue-dart production server, and n-numbers of users start using the application, which is in the first release. While using the application, the blue-dart comes up with more number of features and enhancements, which is sent with the CRS to the TCS after that TCS will do the further changes in modules and sent it back to the blue-dart.

Hence, what is happening here, the application was developed when the requirement is collected by blue-dart from their end-users and customers.

The numbers of releases depend on the following facts:

- Difficulty of modules
- The number of modules.
- How the new module affects the old module.

Note:

Hotfix: In the production environment, whenever the customer identify the critical bug, we will do the following

- The developers fix the bugs.
- Small teams of test engineers will test the software.
- Re-install the application on the client environment.
- The client starts using the new software.

This entire process is known as a hotfix, and it can be done in a few hours or one day.

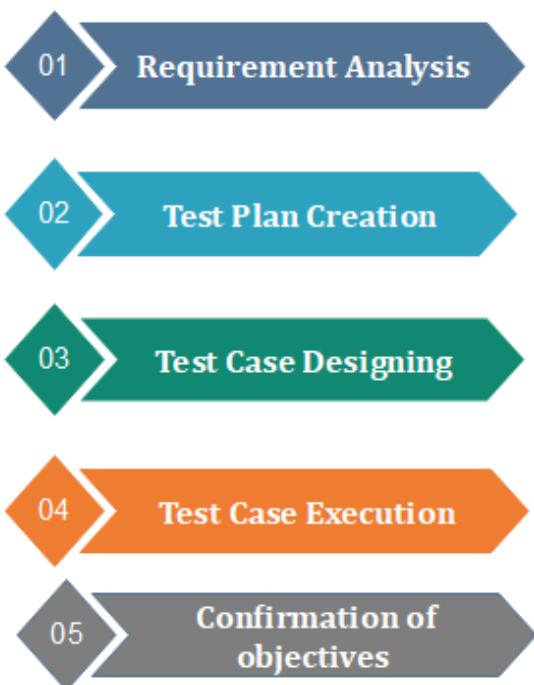
For example: If the significant module, suppose the Login module itself is not working at the production server, then the client will send it immediately for fixing it, and that has to be done as soon as possible.

Short release

Between two major releases, this is a short release of improvements, and it happens when the client needs some small features to change on an urgent basis.

For example, if we have 60 developers, where the ten developers will come out, and out of 40 test engineers, the 3 test engineers will come out, and they develop and test the application. And before adding it to the production server, the customer does one short round of acceptance testing.

Steps to Perform Acceptance Testing



Requirement Analysis:

In this step, the testing team analyzes requirement document to find out the objective of the developed software. Test planning accomplished by using requirement document, Process Flow Diagrams, System Requirements Specification, Business Use Cases, Business Requirements Document and Project Charter.

Test Plan Creation:

Test Plan Creation outlines the whole strategy of the testing process. This strategy is used to ensure and verify whether the software is conforming specified requirements or not.

Test Case Designing:

This step includes the creation of test cases based on test plan documents. Test cases should be designed in a way that can cover most of the acceptance testing scenario.

Test Case Execution:

Test Case Execution includes execution of test cases by using appropriate input values. The testing team collects input values from the end user then all test cases are executed by both tester and end user to make sure software is working correctly in the actual scenario.

Confirmation of objectives:

After successful completion of all testing processes, testing team confirms that the software application is bug-free and it can be delivered to the client.

Tools used in Acceptance Testing

Acceptance Testing can be done by using several tools; some are given below:

done by using several tools; some are given below:

Watir:

Acceptance testing uses this tool for the execution of automated browser-based test cases. It uses Ruby language for the inter-process communication.

Fitness tool:

This tool is used to enter input values and generate test cases automatically. The user needs to input values, these values used by the tool to execute test cases and to produce output. It uses Java language for the inter-process communication. This tool makes it easy to create test cases as well as record them in the form of a table.

Advantages of Acceptance Testing

- It increases the satisfaction of clients as they test application itself.
- The quality criteria of the software is defined in an early phase so that the tester has already decided the testing points. It gives a clear view to testing strategy.
- The information gathered through acceptance testing used by stakeholders to better understand the requirements of the targeted audience.
- It improves requirement definition as client tests requirement definition according to his needs.

Disadvantages of Acceptance Testing

According to the testing plan, the customer has to write requirements in their own words and by themselves but

- a. Customers are not willing to do that; it defeats the whole point of acceptance testing.
- b. If test cases are written by someone else, the customer does not understand them, so tester has to perform the inspections by themselves only.

If the process is done in this manner, it destroys the existence of the Acceptance Testing.

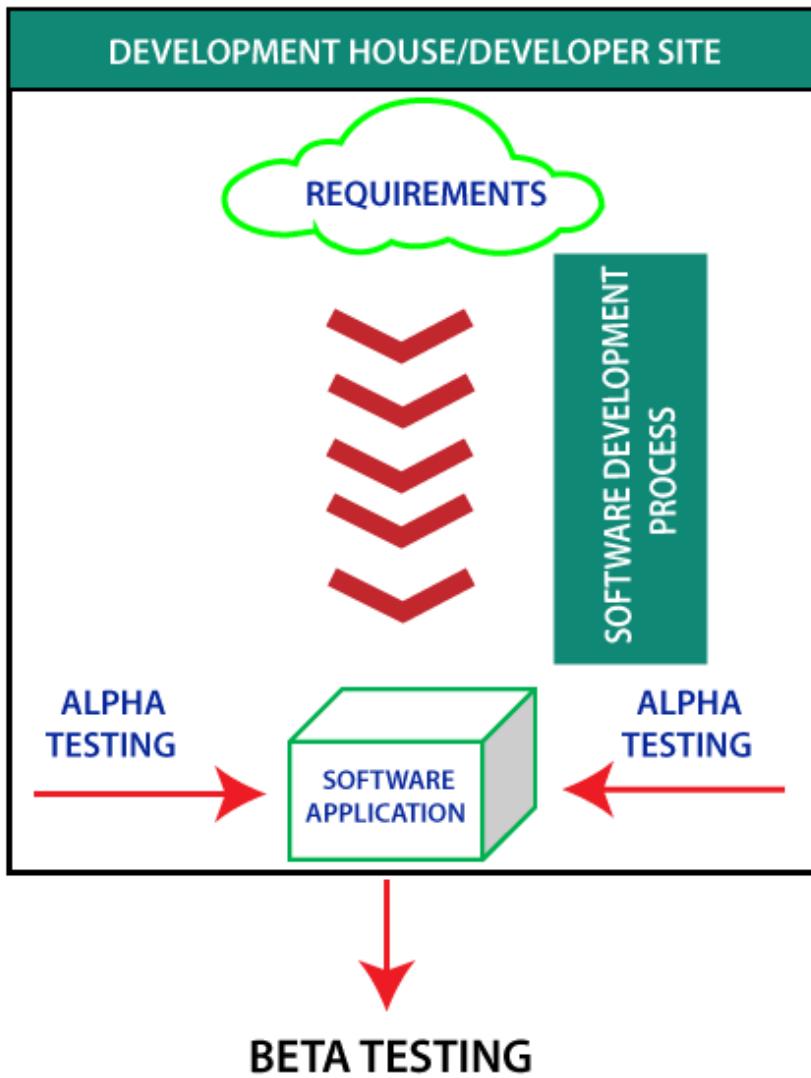
Alpha Testing Introduction



Alpha testing is conducted in the organization and tested by a representative group of end-users at the developer's side and sometimes by an independent team of testers.

Alpha testing is simulated or real operational testing at an in-house site. It comes after the unit testing, integration testing, etc. Alpha testing used after all the testing are executed.

It can be a white box, or Black-box testing depends on the requirements - particular lab environment and simulation of the actual environment required for this testing.



What is the alpha testing process?

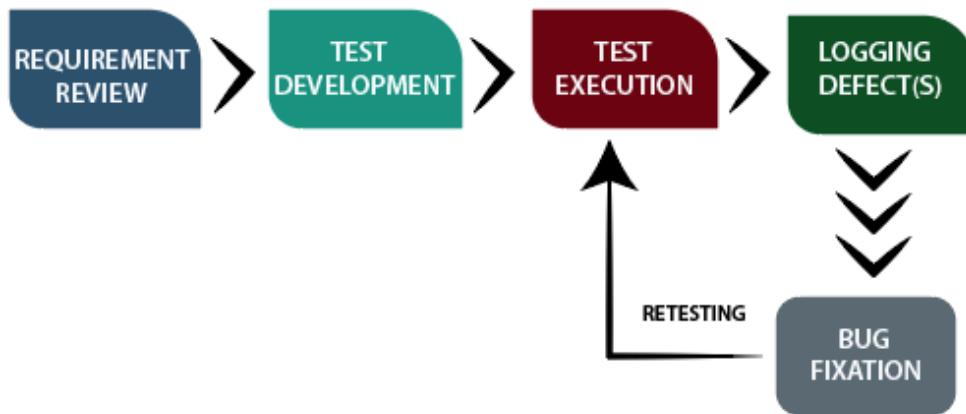
Alpha testing follows the following process:

1. **Requirement Review:** Review the design of the specification and functional requirement
2. **Test Development:** Test development is base on the outcome of the requirement review. Develop the test cases and test plan.
3. **Test case design:** Execute the test plan and test cases.
4. **Logging Defects:** Logging the identified and detected bug found in the application.

5. **Bug Fixation:** When all the bugs are identified and logged, then there is a need to fix the bug.

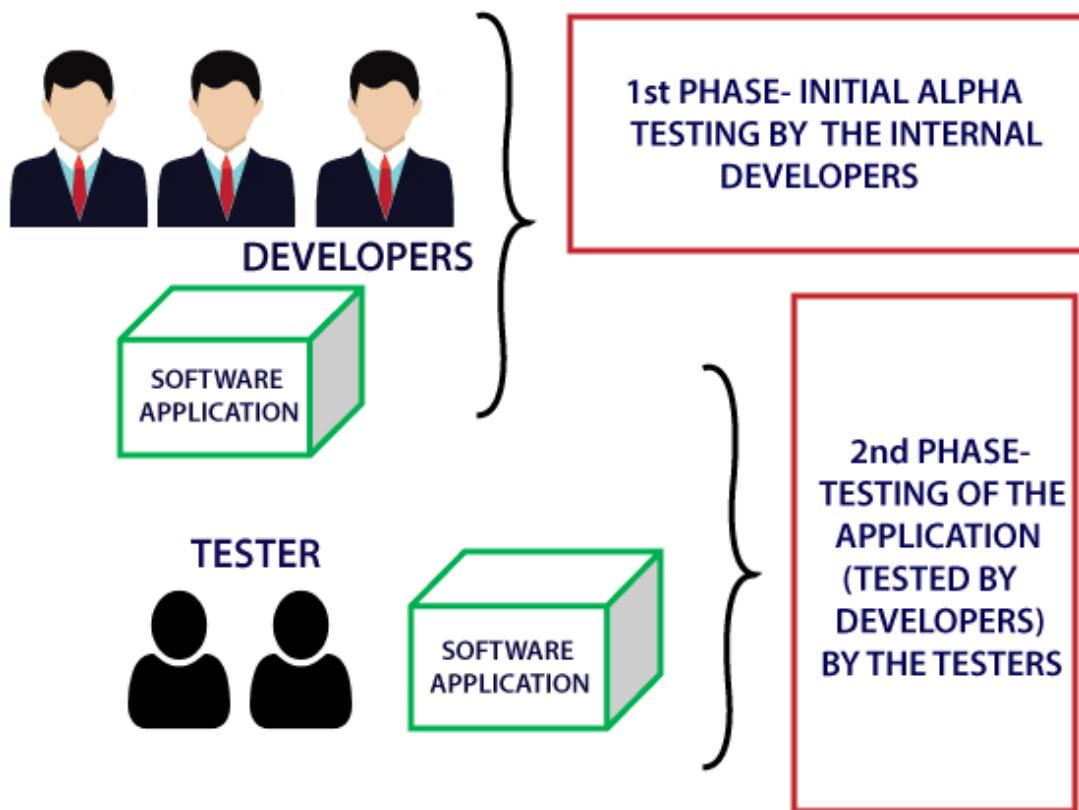
6. **Retesting:** When all the issues are solved, and fixed retesting is done.

ALPHA TESTING



What are the phases of alpha testing?

Alpha testing ensures that the software performs flawlessly and does not impact the reputation of the organization; the company implements final testing in the form of alpha testing. This testing executed into two phases.

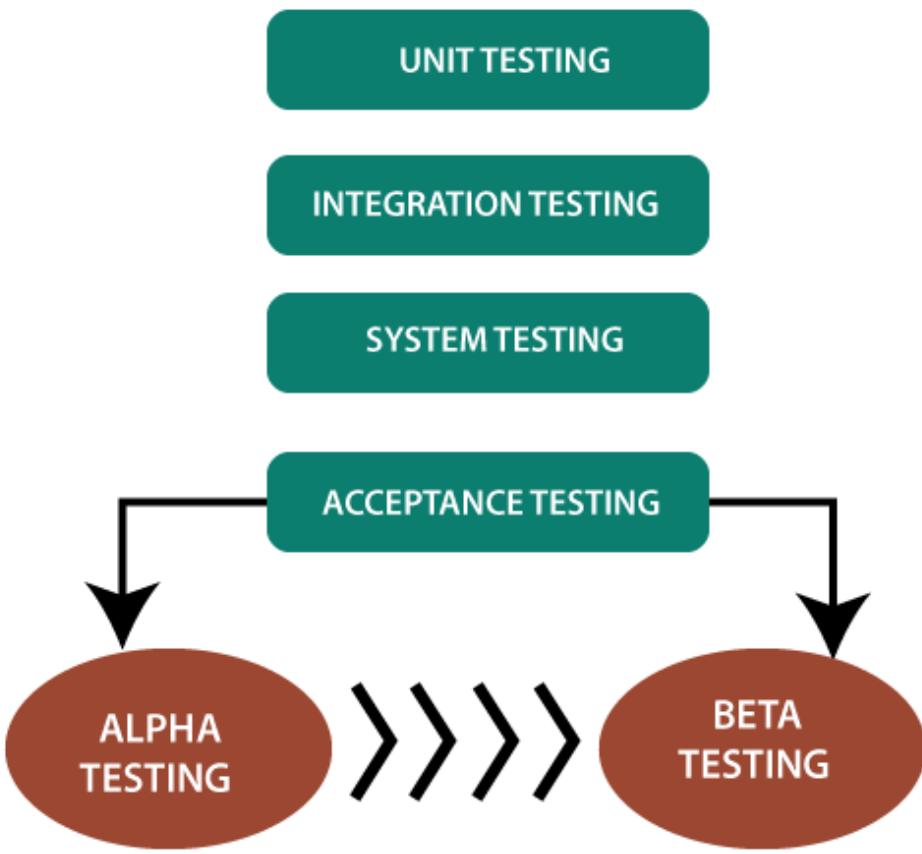


There are two phases of alpha testing.

First Phase: In-house developers of software engineers do the first phase of testing. In this phase, the tester used hardware debugger or hardware aided debugger to catches the bugs quickly. During the alpha testing, a tester finds a lot of bugs, crashes, missing features, and docs.

Second Phase: The second phase involves the quality assurance staff performs the alpha testing by involving black box and white box techniques.

When to perform alpha testing?



Alpha testing is user acceptance testing. Alpha testing performed once the product has gone through stages of testing and prepared for release. It is executing before beta testing, which is also a part of acceptance testing and can define as field testing. During this testing, we can make changes in the software to improve its quality and functionality. Alpha testing done from the developer's site where independent developers can monitor and record user experience and make necessary changes to enhance the performance.

What are the reasons to perform Alpha Testing?

Alpha testing is the final stage of the testing. Alpha testing is an essential and popular testing technique that helps the team to deliver quality and useful software. This testing performed before the release of the product. Alpha testing can define as the first round of independent testing that ensures that the software run as per the requirement plan.

Reasons for alpha testing are:

- Refines the software product by finding and rectifying bugs that weren't discovered through previous tests.
- Alpha testing allows the team to test the software in a real-world environment.
- One of the reasons to do alpha testing is to ensure the success of the software product.
- Alpha testing validates the quality, functionality of the software, and effectiveness of the software before it released in the real world.

What are the features of Alpha Testing?

- Alpha testing is a type of acceptance testing.
- Alpha testing is happening at the stage of the completion of the software product.
- Alpha testing is in the labs where we provide a specific and controlled environment.
- Alpha testing is in-house testing, which is performed by the internal developers and testers within the organization.
- There is not any involvement of the public.
- Alpha testing helps to gain confidence in the user acceptance of the software product.
- With the help of black box and white box technique, we can achieve the alpha testing.

- Alpha testing ensures the maximum possible quality of the software before releasing it to market or client for beta testing.
- Developers perform alpha testing at developer's site; it enables the developer to record the error with the ease to resolve found bugs quickly.
- Alpha testing is done after the unit testing, integration testing, system testing but before the beta testing.
- Alpha testing is for testing the software application, products, and projects.

What are the advantages of Alpha Testing?

Advantages of alpha testing are:

- One of the benefits of alpha testing is it reduces the delivery time of the project.
- It provides a complete test plan and test cases.
- Free the team member for another project.
- Every feedback helps to improve software quality.
- It provides a better observation of the software's reliability and accountability.

What are the disadvantages of Alpha Testing?

Disadvantages of alpha testing are:

- Alpha testing does not involve in-depth testing of the software.
- The difference between the tester's tests the data for testing the software and the customer's data from their perspective may result in the discrepancy in the software functioning.
- The lab environment is used to simulate the real environment. But still, the lab cannot furnish all the requirement of the real environment such as multiple conditions, factors, and circumstances.

Wrap Up:

Every software product needs to undergo a vital methodology before going into a highly competitive market. Alpha testing is one of the vital testing. It needs to be considered by going through the functionality of the software and achieve the confidence in its user acceptance for the real environment, before releasing it into the market.

Introduction of testing

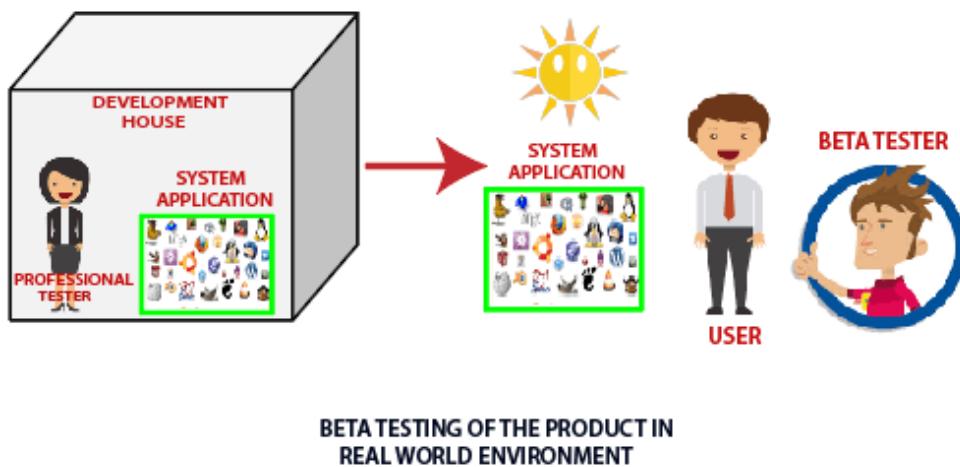


If we compare the various activity performed to develop ideal software, we will find the importance of software testing similar to that of the software development process. Testing is one of those activities which ensure the accuracy of the development process while validating its functionality and performance.

What is Beta Testing?

Beta testing is a type of **User Acceptance Testing** among the most crucial testing, which performed before the release of the software. Beta Testing is a type of Field Test. This testing performs at the end of the **software** testing life cycle. This type of testing can be considered as external user acceptance testing. It is a type of salient testing. Real users perform this testing. This testing executed after the alpha testing. In this the new version, beta testing is released to a limited audience to check the accessibility, usability, and functionality, and more.

- Beta testing is the last phase of the testing, which is carried out at the client's or customer's site.



What are the features of beta testing?

Testing of the product performs by the real users of the software application in the real environment. Beta version of the software is released to a restricted number of end-users to obtain the feedback of the product quality. Beta testing reduces the risk of failure and provides the quality of the product through customer validation. It is the final testing before shipping the product to the customers. Beta testing obtains direct feedback from the customers. It helps in testing to test the product in the customer's environment.

Features of beta testing are:

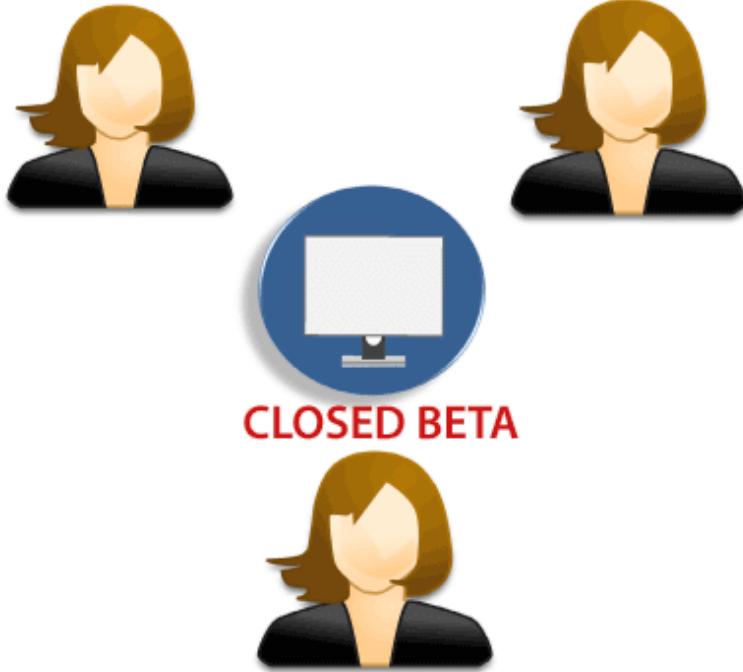
- Beta testing used in a real environment at the user's site. Beta testing helps in providing the actual position of the quality.
- Testing performed by the client, stakeholder, and end-user.
- Beta testing always is done after the alpha testing, and before releasing it into the market.
- Beta testing is black-box testing.
- Beta testing performs in the absence of tester and the presence of real users
- Beta testing is performed after alpha testing and before the release of the final product.

- Beta testing generally is done for testing software products like utilities, operating systems, and applications, etc.

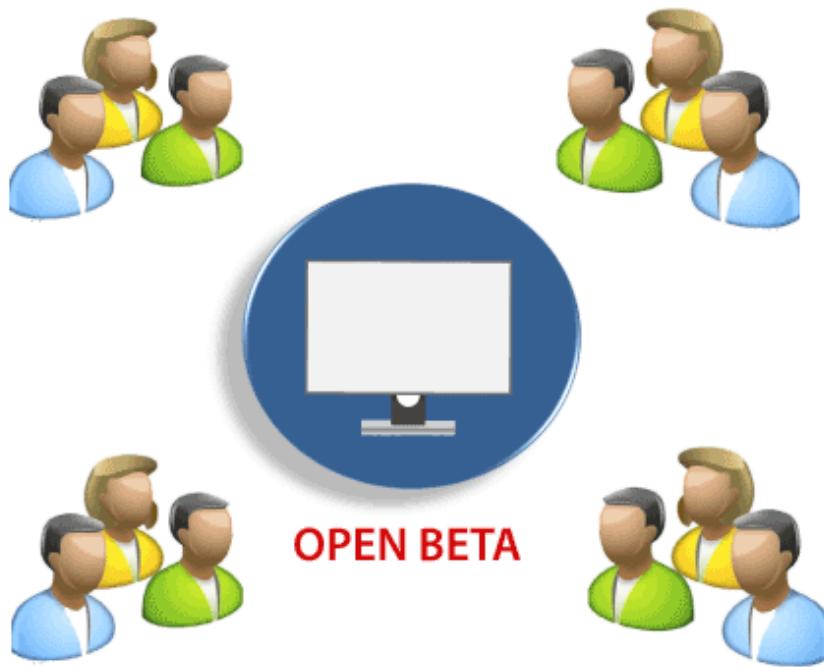
What is a beta version of the software?

The beta version of the software is delivered to a restricted number of users to accept their feedback and suggestions on quality improvement. Hence, there are two types of beta version:

1) Closed beta version: Closed beta version, also known as a private beta, it is released to a group of selected and invited people. Those people will test the software and evaluate their features and specifications. This beta version represents the software which is capable of delivering value, but it is not ready to be used by everyone. Because it shows the issues like lack of documentation or missing vital features.



2) Open beta version: Open beta is also known as a public beta. The open beta opened to the public. Any user as a tester can assess the beta version to provide the relevant feedback and reviews. Open beta version improves the quality of the final release. This version helps to find the various undetected errors and issues.



The beta testing process orients this beta version.

What is the lifecycle of Beta Testing?

A group of end-users performs beta testing. This process can't execute without any strategy or test plan. Before the testers, the end-user executes this type of testing.

The process of beta testing follows the following steps:

1. **Planning:** Like another testing process, beta testing also supports proper planning. In this stage, the team prepares a testing strategy and defines the goal of testing. In this case, the team establishes the need of users for testing, duration, and necessary details related to the process.
2. **Participant Recruitment:** This is the second stage of the beta process in which the team recruits a group of selected end-users for testing. This group can change as per the requirement of the organization and the product.
3. **Product Launch:** When a team of users (testers) recruited. The beta version of the product is launched or installed at the client or user side, and users will test the product for quality assurance.
4. **Collect and Evaluate Feedback:** When the testing finished, developers will collect the feedback provided by the testers and evaluate it. In the end, based on the feedback, issues, and bugs are fixed and resolved by the responsible individual team.
5. **Closure:** When all the problems fixed and the organization meets the exit criteria, beta testing achieved, and the rewards offered to the testing team.

What are the types of beta testing?

Beta testing has six types. Each type has different aspects of the software. All these help developers to improve the quality of the software and allow them to deliver a product that offers excellent user experience. Here are the different types of beta testing:

1. **Open Beta Testing:** Open beta testing involves testing the software product by a large number of people before the final release. The organization decides to make a software product open to the public before releasing the product. Open Beta includes the extensive participation of the public to use and evaluate software product accordingly. Users report the bug to the organization, along with a suggestion to improve the quality of the software.
2. **Closed Beta Testing:** Opposite to the open beta testing. Closed beta testing performed by the selective and limited number of persons. The organization recruits these. In this testing software product is not open to the public.
3. **Traditional Beta Testing:** In this testing, a software product delivered to the target market, and the feedback from the users collected. This type of testing assistance the beta testing, the quality of the software is improved, and developers can make the changes.
4. **Public Beta Testing:** This type of testing is similar to open testing. Public beta testing also allows the product is delivering to the end-users worldwide, with the aid of various online channels available in the world. From this, the feedback and evaluated data also collected and based on the requirement changes, and the development team implements modifications.
5. **Technical Beta Testing:** Technical beta testing is also an essential type of beta testing. This testing involves delivering the software product to the internal groups of the organization. However, the data and feedback provided by the employees of the organization.
6. **Focused Beta Testing:** This type of testing focused on monitoring and evaluating a specific feature or component of the software. In focused beta testing, the software released to the market and user's experience assessed and collected to make the required changes.
7. **Post-Release Beta Testing:** In this testing, the product delivered to the market for the use of the end-users. Their feedback, reactions, and experience are collect for the future release of the software.

When to perform Beta Testing?

Acceptance testing is the final phase of the testing, which combines both alpha and beta testing to ensure that the product released flawlessly. Beta testing performed at the user's end. This testing always performed after the alpha testing, but before the product released to the market. In this stage, the product is expected to be 90% to 95% completed.

Any product undergoing to beta test should be reviewed for the entire checklist before launching it.

Some of them are:

- All the component of the product is ready to start this testing.
- Documentation which is going to end-user should be kept ready - Setup, installation, usage, Uninstallation should be in detail.
- The product management team should review that all the functionality is in good condition.
- Procedure to collect bugs, feedback, etc. should be identified before publishing it.

What are the stakeholders and participants in the Beta Testing?

The product management, quality management, and user experience teams are the stakeholder in beta testing, and they closely monitor every move of the phase.

The real users who use the product are the participants.

Beta test strategy

- Business objective for the product.
- Beta test plan
- The testing approach followed by participants.
- Tools used to detect bugs, measure productivity, collect feedback.
- When and how to end this testing phase?

What is a Beta Test plan?

A beta test plan can be written in many ways,

Objective: We should have to mention the aim of the project why there is a need for beta testing even after performing the internal testing.

Scope: In this plan, we should mention the areas to be tested or not.

Test Approach: We should have to mention clearly that the testing is in the deep, what to focus on - functionality, UI, response, etc.

Schedule: We have to specify, clearly the start and ending date with time, number of cycles, and duration per cycle.

Tools: Bug logging tools and the usage of the machines should identify.

Budget: Incentive of the bugs based on the severity.

Feedback: Collecting feedback and evaluating methods.

- ***Identify and review the entry and exit criteria.***

What are the entry criteria for Beta Testing?

- Sign off the document from alpha testing.
- Beta version of the software should ready.
- The environment should be ready to release the software application to the public.
- To capture the real-time faults environment should be ready.

What are the exit criteria for Beta Testing?

- All the major and minor issues resolved.
- The feedback report should prepare.
- The delivery of beta test summary report.

What are the advantages of Beta Testing?

Beta testing performed at the end of the software testing lifecycle. Beta testing offers numerous benefits to testers, software developer, as well as the users. In the assistance of this type of testing, it enables developers, testers to test the product before its release in the market. The

1. Beta testing focuses on the customer's satisfaction.
2. It helps to reduce the risk of product failure via user validations.
3. Beta testing helps to get direct feedback from users.
4. It helps to detect the defect and issues in the system, which is overlooked and undetected by the team of software testers.
5. Beta testing helps the user to install, test, and send feedback regarding the developed software.

What are the disadvantages of Beta Testing?

Disadvantages of beta testing is:

1. In this type of testing, a software engineer has no control over the process of the testing, as the users in the real-world environment perform it.
2. This testing can be a time-consuming process and can delay the final release of the product.
3. Beta testing does not test the functionality of the software in depth as software still in development.
4. It is a waste of time and money to work on the feedback of the users who do not use the software themselves properly.

Wrap Up

Keeping in mind the characteristics of beta testing can be concluded that beta testing may be considered desirable for the organization. Beta testing provides the feedback of the real-users, which helps improve the software quality before the product released in the market.

Database Testing

In this section, we are going to understand **Database testing**, which checks the schema, tables, triggers, etc. of the database under test.

And we also learn about following concept of database Testing:

- **Why we need to use database Testing**
- **Database testing process**
- **Types of database testing**
- **How to perform database testing manually and with the help of automation tool**
- **What are the different challenges we may face during the database testing?**
- **Components of database testing.**

Before we discuss **database testing**, firstly, we will understand the definition **database**.

What is a Database?

A **database** is a prearranged collection of data containing the information and helps in data manipulation. A database can easily manage and retrieved by the user. We can establish data into tables, rows, columns, and indexes, making it easier to identify the appropriate data.

In a Database, data management becomes a very easy task because we can use the databases as **databases to retrieve** the information like **tables for storing data, function, triggers for data manipulation and view for data representations**.

Note: The Database is become more difficult over time due to the massive amount of data stored in a software system.

After understanding the database concept, we have now come to our main discussion on **database testing**.

Introduction of Database Testing

In **software testing**, **Database Testing** is testing, which is used to analyze the schema, tables, triggers, etc., of the database under test. It also assesses data integrity and consistency, which might include creating difficult queries to load and stress test the Database and review its responsiveness.

Generally, it contains the layered process, which involves **the data access, User Interface [UI], the business layer, along with the database layer**.

During the database testing, we can cover the following database activities, such as:

- **Testing data integrity**
- **Checking data validity**
- **Performance check relate**
- **Triggers and Functions in the database**
- **Testing various procedures**

Why do we need to perform database testing?

If we performed the database testing, it would ensure the database's efficiency, maximum stability, performance, and security.

And these features can be set aside on a check occasionally to confirm that the software application is stable as soon as deployed in a competitive environment. To perform database testing, we must have a basic knowledge of **SQL**.

To get more information about the SQL, we can refer to the following link: <https://www.javatpoint.com/sql-tutorial>

What is the purpose of the Database Testing?

The main objective of performing database testing is to make sure they follow the various aspects:

- **Transaction's ACID Properties**
- **Data mapping**
- **Accuracy of Business Rule**
- **Data integrity**



1. Transaction's ACID Properties

The database testing will ensure **the ACID properties of the transaction**.

A Database performs these four ACID properties. The ACID properties are as follows:

- **Atomicity**
- **Consistency**
- **Isolation**
- **Durability**

Atomicity

- The term **atomicity** in transaction specifics that the data remains atomic, which implies that if any operation is performed on the data, it should be performed or implemented entirely or should not be implemented at all.
- It is also known as **All-or-Nothing**.

Consistency

- The term **consistency** specifies that the value should remain always preserved after the transaction is completed in the transaction.
- And the integrity of the data is very important therefore, the database remains consistent before and after the transaction. The data should always be correct.

Isolation

- In the transaction, the term **isolation** means separation, which specified that the multiple transactions could implement all at once without impacting one another and changing the database state.
- Or if two or more transactions occur concurrently, the consistency should remain preserved.

Durability

- The word **durability** makes sure the permanency of something, which further implies if a transaction is committed, it will keep the modifications without any fail irrespective of the effect of external factors.
- And the durability of the data should be so faultless even though the system fails, the database still survives.

2. Data Mapping

Data mapping is an essential feature in database testing, which is mainly focused on verifying the data that pass through back and out between the application and the backend database.

Below are some of the important features tested in the data mapping:

- We analyze whether the user interface or front-end methods are mapped constantly with the equivalent fields in the database table.
- Characteristically, this mapping information is specified in the requirements documents.
- When a specific action is done at the front-end of an application, an equivalent **Create, Retrieve, Update and Delete [CRUD]** activity gets used at the back-end.
- And then, the test engineer will have to assess if the correct activity is used and whether the user action in itself is effective or not.

3. The Accuracy of the Business Rules

- Database testing ensures the accuracy of the business rules as we know that complex databases lead to complex components such as **stored procedure triggers and relational constraints**.
- Therefore, the test engineer will come up with appropriate SQL commands to verify the complex objects.

4. Data Integrity

- The database testing also makes sure the data integrity, where we can update, and the most recent shared data values should appear on all the forms and screens.
- And if the value should not be modified on one screen and show an older value on another, then the status can be updated concurrently.

How to perform Database Testing

We can perform the database testing either manually or with the help of some automation tools.

How to perform database testing manually

To perform database testing manually, which need to follow the below process:

- Firstly, we will Open the **SQL server** in our local system.
- After that, we will open the **query analyzer** to write the command and retrieve the data.
- Once we can retrieve the specified data, we will compare the detailed data with the expected result.
- Then we can update or delete the data to check how the software application performs.
- The general test process of the testing database is not dissimilar from any other application. Therefore, to run the test, we can follow the below steps:

How to perform Database Testing Manually



Step1: Set up the test environment

Firstly, we need to prepare the test environment to test the software application.

Step2: Execute the test

Once we set up the test environment, we will run particular test case.

Step3: Check the results

When the test case is executed successfully without having any issues, we will check the specified test case results.

Step4: Validate the output with the expected one

After checking the test case result, we will validate the same output with the expected one. If the results meet the expected output, the test case will consider as a pass; otherwise, it will be marked as a fail.

Step5: Report the results to the stakeholders

And at last, we will report the results to the stakeholder of the particular software application.

Note: If we set-up the environment, the test engineer and developers will develop all possible scenarios, which can execute through the application.

Then, the test will involve running through these queries and checking the data integrity, which means that the resulting data will need to be truthful, accurate, complete, retrievable, and verifiable.

And the test could also include monitoring data mapping, the different ACID properties, and ensuring the implemented business rules' accuracy.

How automation can help in Database Testing

In software testing, **Automation testing** is used to decrease the repetitive manual work, which helps the test engineer focus more on critical features, which work the same for the database testing.

Let's see a few scenarios where automation can be very useful for the test engineer:

- **Modification in the database schema**

When every schema is modified, the database needs in-depth testing to ensure that the things are in place. And the number of scenarios to be covered based on the size of the database. And this process is time-consuming if we have done it manually.

- **Monitoring for data integrity issues**

There can be a condition where a set of data gets corrupted in recovery or other actions because of human error or other issues.

But if we consider the automated monitoring processes, it became easier to find those variations, and we can fix them as soon as possible.

- **New or frequently altering applications**

As we know that Agile methodology is the new era of testing, where we will have a new release to production at the end of every sprint, which implies it will take every 2-3 weeks to complete one round of testing.

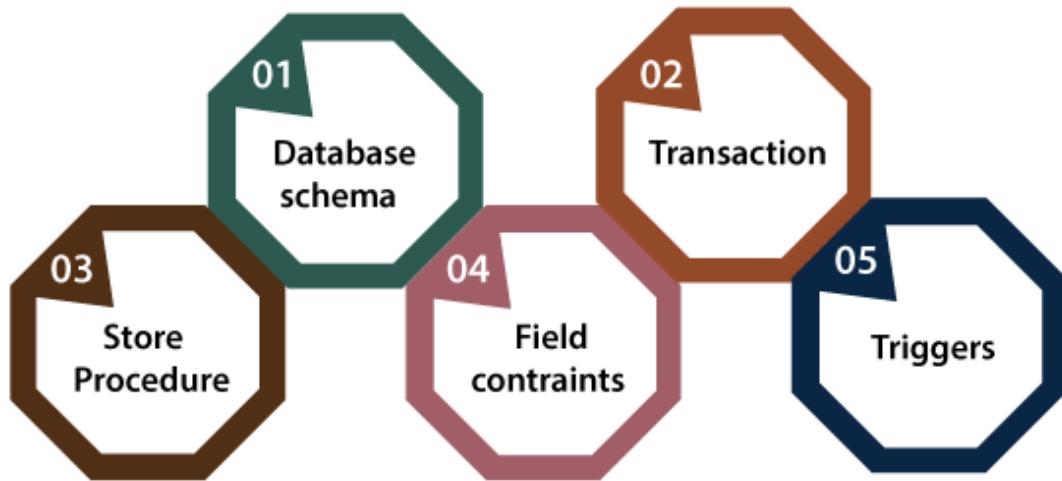
But with the help of automation features, which is completely constant and unchanged in the recent sprint, we can focus on new modified requirements.

Database testing components

Following are the components of the database testing:

- **Database schema**
- **Transactions**
- **Store procedure**
- **Field constraints**
- **Triggers**

Database Testing Components



1. Database Schema

A database schema is used to describe the working of data in a Database and its organization. In other words, we can say that it is nothing but a proper classification of how the data will be planned inside the database.

For testing those conditions, we have two ways, which are explained below:

One of the Below approaches can be used as per the tool's significance:

- We can use the **SchemaCrawler** tool, which is a free database schema discovery and comprehension tool.
- **Regular expressions** are a good approach to verify the names of the specific fields and their values.
- To verify the schema, we can use the following SQL command:

```
DESC<table name>
```

Find the needs according to the Database operates

- The **Field names** begin or end with explicit characters.
- The Primary keys need to be generated before any other fields are designed.
- The specific values can or cannot be inserted in the fields that have a constraint.
- For easy recovery and search, the Foreign keys must be indexed completely.

2. Transactions

One of the most important database testing components is **transactions** because while we are performing the database testing, the ACID properties need to satisfy.

- The most commonly used statements are as below:

```
BEGIN TRANSACTIONTRANSACTION#
END TRANSACTIONTRANSACTION#
```

- To make sure the database remains in a consistent state, we can use the below **ROLLBACK** commands:

```
ROLLBACK TRANSACTION#
```

- To ensure the modification have been reproduced, we can use a **SELECT** command after the implementation of the above command:

```
SELECT * FROM TABLENAME < Transactions Tables >
```

Note: In the above statement, the Transaction table is the table that includes the transaction.

3. Stored Procedure

The Stored Procedures are relatively parallel to user-defined functions. And the entire system works in loomng with the most consistent and correct result.

It can be used by **Execute or Call Procedure** commands, and generally, the output is in the format of result sets. The stored procedure system is used for multiple applications where data is kept in RDBMS.

We can test the stored procedure throughout the **white-box and black-box testing**.

- **White box testing:** In white box testing, the Stubs are used to invoke the stored procedures and then the output is verified in contradiction of the expected values.
- **Black box testing:** In this, we can operate the **application's front-end (UI)**. And also assess the implementation of the stored procedure and its outputs.

4. Field Constraints

The next database testing components are **Field Constraints**, where the entire system works on the **default value, exclusive value, and foreign key**.

In this, we can easily verify the outcomes retrieved from the SQL commands, and to ensure that the object condition in the database is implemented, we can perform the Front-end (user interface) operations.

5. Triggers

The **trigger components** are used to implement an entire table independently to record the output. In other words, we can say that a trigger (a piece of code) can be auto-instructed to be performed if a particular event takes place on a precise table.

Let us see one **sample example**, where we can understand the working of trigger components in the database testing:

- Suppose a new employee joined a company. And the employee is doing two tasks, which are **development and testing**. Then the employee is added to the **Employee table**.
- Once he/she is added to the **Employee table**, a **Trigger** could add the employee to the equivalent **task's**
- After that, we can follow the common process to test it, firstly the SQL command implanted in the Trigger independently and then it records the result.
- In the end, we can follow this process to perform the trigger as an entire system and then compare the outcomes.

These types of tests are completed in two ways, which are as follows

- **White-Box Testing**
- **Black-Box Testing**

Both the white-box and black-box testing have their procedure and sets of rules, which help us get the precise result.

Types of Database Testing

The database testing classified into three different types of testing, which are as follows:

- **Structural testing**

- **Functional testing**
- **Non-functional testing**

Types of Database Testing



Let understand each type one by one:

Structural Database Testing

- It is a most important database testing technique used to verify all the elements inside the data repository, which are primarily used for data storage and not allowed to be directly operated by end-users.
- If we want a successful conclusion of this testing, we must have a complete understanding of SQL commands.
- In structural database testing, we can test the database components that are not visible to users.
- The structural database testing is mostly vitally used to validate the database.

Functional Database Testing

- The most important database testing approach is functional database testing, which is used to authorize a database's functional requirements from the end user's interpretation.
- The functional database testing's primary purpose is to test whether the end-user's transactions and operations are connected to the database work as expected or not.

Non-functional Testing

In the topic of database testing, Non-functional testing can be divided into several types as vital by the business requirements.

Some of the important parts of non-functional testing are listed below:

- Load testing
- Stress Testing
- Security Testing
- Usability Testing
- Compatibility Testing

Note: The load testing and the stress testing come under Performance Testing, which helps two specific non-functional testing objectives.

What are the different challenges of Database Testing?

While performing database testing, we may encounter the following challenges.

In the below table, we listed some common challenges and their solutions:

Different challenges

Solutions

Testing huge data and production simulated the database	<ul style="list-style-type: none">○ A scaled-down or enhanced database would be the best solution because it is as close as possible to the production data set.○ Although it is a good approach to test in a production-like environment, it could sometimes become a very challenging and time-consuming process to test on enormous data.
Re-usability of data again for testing and Test data creation	<ul style="list-style-type: none">○ To resolve this specific issue, we need a better strategy to generate all the essential data for all the stretch repetitions. And then, we can use detailed data carefully.○ All the commands need to be separate from each other; for example, the input data and output of one command do not modify another command's output.
Separation of data and queries	<ul style="list-style-type: none">○ The software product quality depends on the cost.
Cost and time were taken to get the data from a massive database	<ul style="list-style-type: none">○ Therefore, it is significant to maintain a balance between the project timelines, expected quality and data load, and additional factors.
Frequently altering the database structure	<ul style="list-style-type: none">○ While performing DB testing, the Database test engineers most frequently faced this challenge.○ The best solution for the particular challenges is that the DB tester needs to create the test cases and the SQL command derived from the specific structure, which gets modified at the time of implementation or through any other retesting.○ To avoid the final delays, we need to intercept the modification and the impact as early as possible.
Unwanted data modification	<ul style="list-style-type: none">○ The best solution for unwanted data modification in DB testing is access control.○ We can provide access only to a limited number of people for the modification.○ And the access should be restricted for the EDIT and DELETE options.

Misconception or Myths related to Database Testing

When we are performing the database testing, we may undergo some misconceptions about database testing.

Let us see them one by one and also understand the reality of the related myths:

Misconception or Myths Reality

The overall development process will slow down because of the database testing.	For this particular myth, the reality is that database testing helps us enhance the database application's overall quality.
Database Testing is a monotonous job, and it involves plenty of expertise.	In software testing, database testing is an efficient process, which gives long-term functional stability to the entire application.
Database testing is an expensive process.	Expenses on Database Testing is needed because any expenses on database testing is a long-term investment that leads to long-term robustness and constancy of the application.
Database testing increases additional work for bottlenecks.	In reverse, identifying the hidden defects with the help of database testing enhances more value to the overall work.

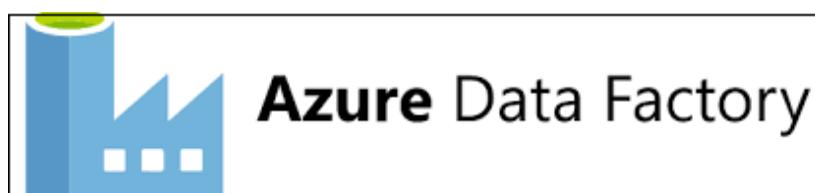
Automation tools used in database testing

We have several Database testing tools available in the market, but here we are discussing some of the most commonly used automation tools for database testing, which are as follows:

- Data factory
- SQL test
- Mockup data
- MS SQL Server
- DbUnit



Data Factory



- The data factory is one of the most popular tools used for database testing.
- Mostly it is used for commercial database testing tools, which means that the huge project can be tested by a data factory tool.

- It works as a data generator and data manager in the context of database testing.
- For handling the complex command with a large amount of data, it is the most efficient tool.
- This tool gives us a platform to easily perform the stress or load testing on the database.

SQL Test



- The SQL test is the most frequently used Database testing tool available in the market.
- It is an open-source tool **tSQLt** framework, which means that it can be used by all the Database test engineer at least once.
- It allows us to execute the unit tests for SQL Server databases.
- With the help of this tool, we can easily execute extensive SQL tests.
- The major drawback of this tool is that it is slow compared to the other Database testing tools in the market.

Mockup data



- The Mockup Data testing tool also comes under the Test Data Generator category, and it is commercial testing tool.
- In this tool, we need to add columns in our table to validate the outputs.
- It helps us to create a huge Amount of Data, CSV files and databases with accurate data.
- It rapidly creates a large amount of data and tests several tables for a relationship with foreign keys.

MS SQL Server



- The Microsoft SQL server tool is extensively used to execute the unit testing.

- It is a commercial tool where we can generate in VB or C# projects, and the test engineer is expected to understand the project schema before starting the test.
- Even though we are creating the tests from a database project, we can use SQL Server Object Explorer.
- The main disadvantage of this tool is that it does not have any good user interface.

DbUnit



- It is an open-source tool, which is also known as **the JUnit extension**.
- It helps us export and import data into a database to and from XML datasets and work on large databases.
- It performs the **CLEAN-INSERT** operation initially; that's why it does not perform any further clean-up.
- With **the DBUnit** tool's help, we can explore the data and connect relational and multidimensional databases.

Conclusion

In the **Database testing** section, we have learned the following topics:

- Database Testing is testing, which is used to analyze the schema, tables, triggers, etc., of the database under test.
- To understand the database testing process's key concepts, the database test engineer must be aware of database testing's various features, types, manual and automation processes, and database testing tools.
- With this tutorial's help, we get to know about the misconception or their solution of database testing.

Mainframe Testing

In this section, we are going to understand the **Mainframe testing**, which is used to test the software or the applications and services developed on Mainframe Systems.

And we also learn about the mainframe attributes, types of mainframe testing, how to perform it, the different challenges and Troubleshooting while performing the mainframe testing, various commands used in Mainframe Testing, some common issues faced during mainframe testing, and Mainframe Automation Testing Tools.

Before understanding the concept of Mainframe testing, we are going to learn about the Mainframe.

What is Mainframe?

The mainframe is a multi-user, high performance and high-speed computer system. The mainframe is the most reliable, scalable, and secured machine systems.

In other words, we can say that these systems are used for the intend of larger-scale computing, which involves great availability and safety. The mainframe systems are mostly used in various subdivisions such as Retail, insurance, finance, and other critical areas where massive data are processed several times.

In this, we can perform some million Instructions per second [up to **569,632 MIPS**] with the help of the below aspects:

- **Maximum Input/output bandwidth:** The connections between drives and processors have a few choke points if we have the extreme input and output bandwidth.
- **Reliability:** Frequently, the mainframes agree to the **graceful degradation** and service though the system is running.
- **Reliable single-thread performance:** It is important for practical operations against a database.
- **Maximum Input/output connectivity:** The maximum input/output connectivity implies that the mainframes excel at providing huge disk farms.

After understanding the mainframe concept, now we have come to our main point of discussion on **mainframe testing**.

What is mainframe testing?

The main objective of mainframe testing is to make sure the dependability, performance, and excellence of the application or service by verification and validation approaches and check if it is ready to deploy or not.

The tester only needs to know about the CICS screens' navigations while performing the Mainframe testing as these are custom-built for specific applications.

And the tester does not have to worry about the emulator set up on the machine if any modification happens to the code in COBOL, JCL, and so on.

Where we performed the mainframe testing?

Generally, the mainframe testing is executed on the deployed code with multiple data combinations set into the input file. In other words, we can say that the **Mainframe Applications** must be tested completely before the production run.

The Mainframe application, or else known as the batch job, is tested in contradiction of the test cases developed with requirements.

The software or the application which runs on the mainframe can be retrieved through the terminal emulator.

After understanding the mainframe testing, we will look into the several characteristics of mainframe testing.

Mainframe Testing Methodology

In Mainframe testing, the Software or an application are retrieved by **end-users** in a way, which is diverse from Web applications.

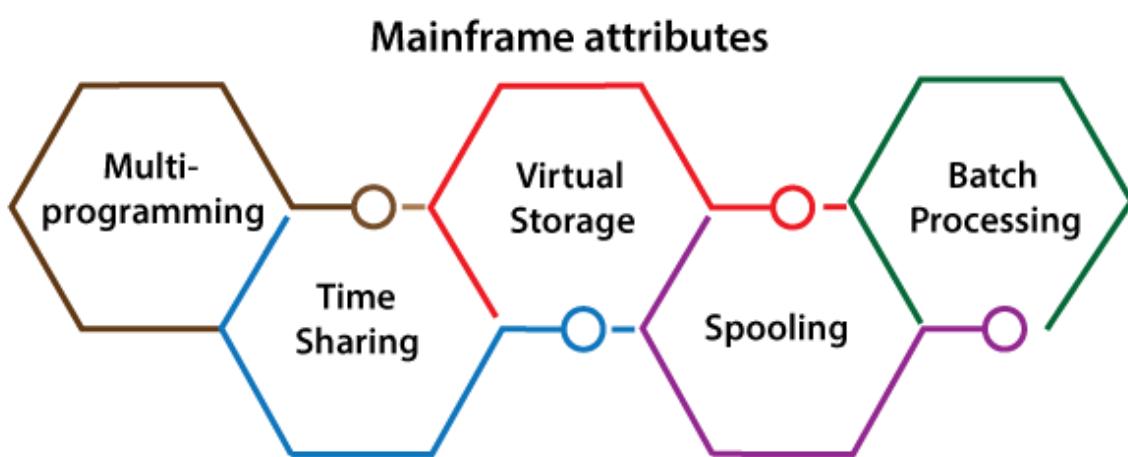
And the application tester should know these significant differences, which are shown below:

Web Applications	Mainframe Applications
The web applications are acquired via two-tier architecture [Client/Server] or three-tier architecture [Presentation/apps/DB storage layers] .	In the mainframe application, the end-user must log into the system directly.
The web applications retrieve across Browser or UI .	Terminal Emulator must retrieve the web applications.
In this, the testing can be performed straight on the application screen.	In this, the tester should have precise knowledge of mainframe operations.
In the web application, some of the processing is done on the Client-side, and the applications should install at the right place before the testing begin.	In mainframe testing, the Terminal Emulator is the only software, which has to be loaded in a client machine for testing where the applications exist in the main server.

Mainframe Attributes

In mainframe testing, we have the following mainframe features; let us see them one by one:

- **Multiprogramming**
- **Time sharing**
- **Virtual storage**
- **Spooling**
- **Batch processing**



Multiprogramming

- The multiprogramming attribute is a facility, which allows us to make efficient use of the CPU.
- The computer implements various programs at the same time.

Time-Sharing

- **Time-share** processing is also known as **Foreground Processing**, whereas **batch job processing** is known as **Background Processing**. Therefore, it is known as an **Interactive Processing** because it allows the user to relate with the computer directly.
- In a time-sharing system each user can access to the system over the terminal device.

Virtual Storage

- Virtual storage uses disk storage as an extension of real storage.
- It is a technique to use memory efficiently to store and perform many sized tasks.

Spooling

- The Spool means **Simultaneous Peripheral Operations Online**, which is used to accumulate the output of a program or an application.
- If it is required, then the spooled output is directed to output devices like a printer.

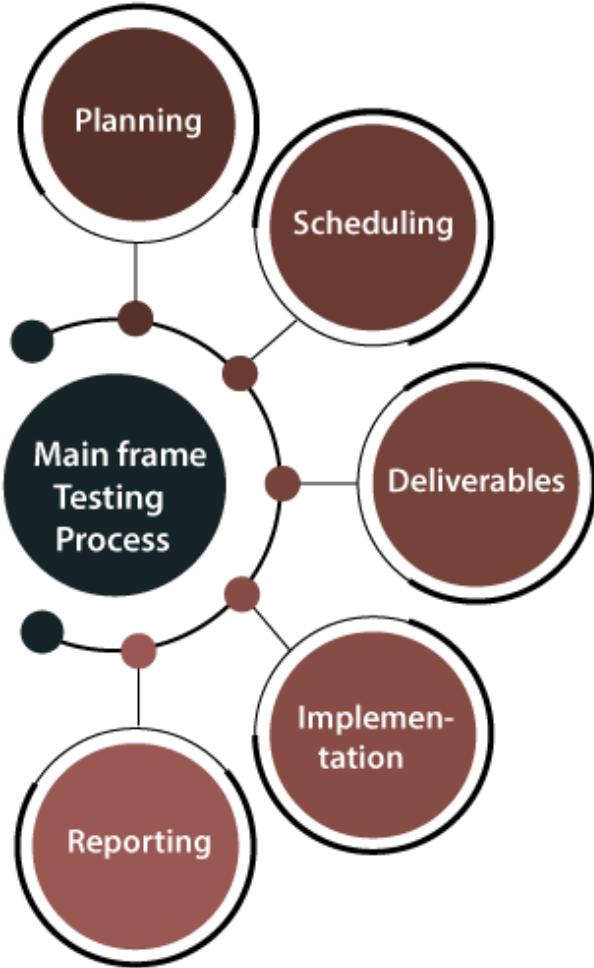
Batch Processing

- Batch processing is a technique where we can accomplish any task in units also known as **jobs**.
- We can perform one or more programs in an order based on jobs.
- The Job scheduler concludes about the sequence where the jobs are implemented.
- Jobs are planned as per their importance and class to maximize the average output.
- The batch processing gives us the necessary information with the help of **JOB CONTROL LANGUAGE(JCL)**.

How to do Mainframe Testing

The mainframe testing can be performed in two ways, either **manually** or using some **automation tools** such as **QTP, REXX, and IBM application performance analyzer etc.** But generally, we will execute the Mainframe testing **manually**.

To do mainframe testing, we need to follow the below steps:



Step1: Planning

Firstly, the **business team** or the **development team** constructs the **test plans** including the **Business requirement document, System requirement document**, other **project documents** and the **inputs**. And it controls how a specific item or process is going to be changed in the release's cycle.

Meanwhile, to prepare the test scenarios and test cases in advance, the testing team will coordinate the development and the Project management teams.

Step2: Scheduling

Once the requirement document is prepared successfully, it will hand over to the development team and the testing team. And the testing schedule should be writing with the project delivery plan, which should be accurate.

Step3: Deliverables

After receiving the document, they will check the deliverables. And the deliverables should well describ without having any uncertainty, and it should be fulfilling the scope of test objectives.

Step4: Implementation

After that, the implementation should be done as per plan and deliverables.

Generally, the 15-25% of the application will be affected directly by the modified requirement in a release. And the other 60-75% of the release will be depend on the out-box-features such as testing the applications and processes.

Therefore, we need to test the Mainframe application in two ways:

- **Testing Requirements**
- **Testing Integration**

Testing Requirements: We will test the application for the features or the modification disclose in the requirement document.

Testing Integration: **Regression Testing** is the main attention of this testing activity. And we will test the entire process or other applications that receive or send data to the precious application.

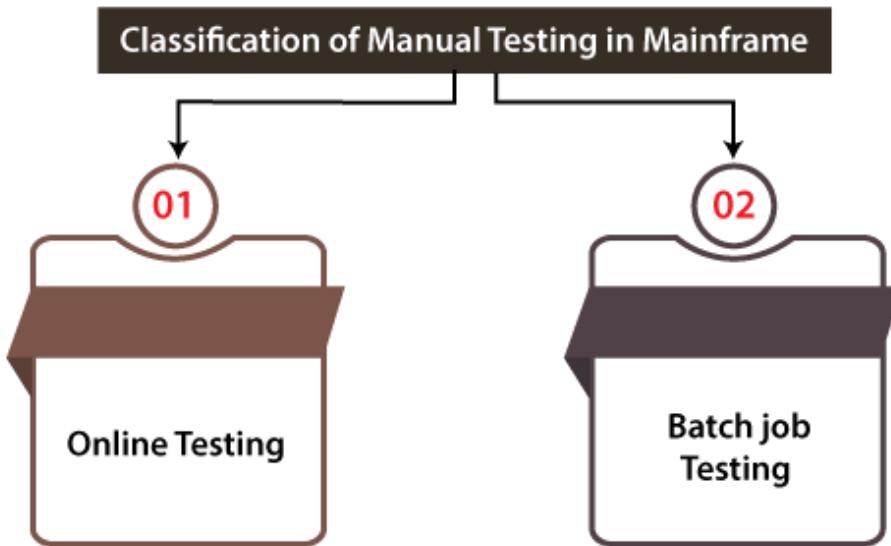
Step5: Reporting

After that, we will share the test results with the development team on a periodical basis. To maintain continuity, the testing team should be in touch with the development team for instant modification in crucial conditions.

Categorization of Manual Testing in Mainframe

Mainframe **Manual Testing** can be categorized into the following two types:

- **Online testing**
- **Batch job testing**



Online Testing

- Online Testing refers to the testing of **CICS screens** that is equivalent to **web page testing** where the functionality of the existing screens can alter, or new screens can be added.

- Various applications can have inquiry and modification screens where screens' functionality needs to be tested as part of the online testing.

Batch Job Testing

- Generally, in batch job testing, the testing process contains **batch jobs** for the functionality, which is executed in the existing release.
- The test result will extract from the output files, and the database should be tested and documented.

Mainframe Testing commands

In mainframe testing, we have used various commands, which are very helpful while testing an application or software.

Some of the most commonly used commands in mainframe testing are shown in the following table:

Commands	Description
COPY	The COPY command is used to copy a dataset .
SUBMIT	The SUBMIT command is used to submit a background job .
RENAME	The RENAME command is used to rename the dataset .
DELETE	The DELETE command is used to remove the dataset .
JOB SCAN	The JOB SCAN command is used to fix the JCL with the libraries, program, file , and so on without implementing it.
CANCEL	The CANCEL command is used to cancel the background job .
ALLOCATE	The ALLOCATE command is used to allocate a dataset .

Note: We have some other commands while performing mainframe testing, but they are not used regularly; therefore, we used those commands whenever it is needed.

Prerequisite statistics on Mainframe operations

A tester should have the following information on mainframe operation while performing the mainframe testing on an application:

- **ISPF [Interactive system productivity facility]** has to be used for menu operations.
- **TSO [Time-sharing option]** is a method that is used to access virtual storage and manage datasets with the help of commands.
- The **FTP [file transfer protocol]** or another transfer protocol must be used while testing software.
- If the mainframe is the backend, we should use the **cross-platform operations**.
- **SDSF [System display and search facility]** has to be used for controlling operations and system resources.
- Batch job management has to be followed.
- CICS transaction has to be used for testing the IBM mainframes.

Mainframe testing Challenges and Troubleshooting

If we are performing the mainframe testing on the application, we might have encountered some challenges, and for those challenges, we have some specified approach, as we can see in the below table:

Challenges	Explanation	Approach
Unclear or Incomplete Requirements	Sometimes a user may have access to the user manual or the training guide , but which are not similar to the documented requirements.	<ul style="list-style-type: none"> ◦ The Test engineer should be actively part of the Software Development Life Cycle (SDLC) from the requirements phase to overcome the unclear requirement issue. ◦ And if the requirements are testable, then it will help them to validate easily.
Identification or Data Setup	Sometimes, the tester might have a condition to reuse the current data as per the requirement. But occasionally, it is hard to find the mandatory details from the current data.	<ul style="list-style-type: none"> ◦ To overcome the data setup challenges, we can use the homegrown tools as per the requirement. ◦ And to get the existing data, queries should make in advance. ◦ In case of any trouble, a request place to the data management team to generate or replicating the vital data.
Job Setup	The job needs to be setup in the QA region when the jobs are saved into PDS. Therefore, the jobs are not submitted with a production qualifier.	<ul style="list-style-type: none"> ◦ For this, the Job set-up tools can be used to overcome human errors during setup.
Ad-hoc Request	Sometimes, we may encounter some situations where the end to end testing needs to be maintained due to a problem in upstream or downstream application. And the Ad-hoc requests enhance the time and effort in the execution cycle.	<ul style="list-style-type: none"> ◦ To overcome the particular challenges, we can take help with some regression scripts, automation scripts, and skeleton scripts, which decrease the time and effort overhead.
On-Time Releases for scope alteration	Sometimes, we have the condition where the code effect may entirely modify the look and feel of the system. And the modification could be in the test cases, scripts, and data.	<ul style="list-style-type: none"> ◦ For this, the impact analysis and the scope change management process should be in placed properly.

Steps to follow in Mainframe Testing

The below steps need to be followed while performing the mainframe testing:

Step1: Smoke Testing

In the first step, we will be performed **smoke testing**, where we check whether the code installed is in the correct test environment. And it also makes sure that there is no critical issues with the code, which saves the effort of testers' time in testing a faulty build.

Step2: Functionality Testing/ System testing

After performing the smoke testing, we will do one round of functionality or **system testing** to test various models' functionalities independently and concerning each other.

Following are the types of testing which has to done while implementing the System Testing.

- **Batch testing**
- **online testing**
- **Online-batch integration testing**
- **Database testing**
- **Batch Testing**

We will perform the batch testing to authenticate the test result on the output files and data modification completed by the batch job with the testing specification.

Note: Batch jobs are a set of events that got implemented without any user interaction with accessible computing resources.

- **Online Testing**

In online testing, we will test the **front-end** features of the mainframe applications. The online testing covers various aspects such as **user-friendliness, data input validations, look & feel, navigations within the screen etc.**

The application should be tested for exact entry fields such as interest on the plan, an insurance plan, etc.

- **Online-Batch Integration Testing**

The online-batch integration testing can be performed on the systems with the batch processes and online application. And here, we also tested the Integration features of the online process with the backend process.

Basically, in this testing, we validate the data flow's accuracy and the interactions between the screens and the backend system. And the batch job is used to check the data flow and communication between the online screens.

- **Database Testing**

The database testing is used to test the data stored by the transactions for conformance with the system specification. And the databases validated their layout and the data storage, which contains the data from the mainframe application such as **IMS, IDMS, DB2, VSAM/ISAM, Sequential datasets, GDGs** are

And in database testing, we can also authenticate the **data integrity** and other database parameters for their ideal performance.

Step3: System Integration Testing

The system integration testing is used to check the systems' functionality relating to the system under test. It is executed after unit-level tests because it is important to test the interface and serval types of messages such as **Job Successful, Job Failed, Database updated, etc.**

And for correctness, we will test the data flow across the modules and applications. The system integration testing is performed to make sure the readiness of the build for deployment.

In system integration testing, we can perform the below testing:

- **Batch Testing**
- **Online Testing**
- **Online -Batch Integration Testing**

Step4: Regression Testing

The most important phase of any testing is regression testing. The regression testing is making sure that the **batch jobs and the online screens** cannot directly relate to the system under test, which are not affected by the current project release.

The regression testing guarantees the modification done in a module and does not affect the parent application's and the overall function along with the integrated application.

A specific set of test cases should be accepted based on their complexity, and a **Test cases repository** should be created to get a successful regression testing. And the particular test should be modified whenever there is a new functionality moved into the release.

Step5: Performance Testing

In mainframe testing, the next step is performance testing. In performance testing, we will try to find the bottlenecks in important areas such **as front-end data, upgrading online databases, and project the application's scalability.**

In the Mainframe applications, we may encounter the below performance bugs:

- The online **response time** could be slow, which leads to user disappointment.
- Sometimes the **Batch jobs and backend process** takes extra time, which cuts into system accessibility to the online users.
- Scalability issues.

To overcome the above bugs, we should test the application appropriately with the help of the following:

- System integration parameters
- Application and Database design
- Coding
- System and database parameters
- Timing of back end jobs

Step6: Security Testing

The security testing includes evaluating the **threats, risks, vulnerabilities** and recommend remedial actions for applications and networks.

The security testing should cover use cases in **identity and access management, Risk & Compliance Management, Data protection & privacy policy adherence.**

In other words, we can say that security testing is performed to check how well the application is designed and developed to conflicting anti-security attacks.

Security testing should be completed on the two types of security systems: **Mainframe security and Network security.**

In security testing, we need to test the following aspects:

- Authorization
- Integrity
- Authentication

- Confidentiality
- Availability

Step7: Agile Methodologies

The Agile methodology is used to simplifies the gradual development of application and responds to modification quickly.

Note: In an agile development scenario, we can use the Incremental Testing approach.

Mainframe Automation Testing Tools

We have various types of mainframe automation testing tools available in the market. Some of the most commonly used mainframe automation testing tools are as follows:

- **QTP**
- **REXX**

QTP [Quick Test Professional]

QTP tool is used to test functional regression test cases of the web-based application. QTP stands for **Quick Test Professional**, and now it is known as **Micro Focus UFT [Unified Functional Testing]**.

It is very helpful for the new test engineer because they can understand this tool quickly. QTP is designed on the scripting language like VB script to automate the application.



Features of QTP

Following are the most common features of QTP:

- In this tool, we can perform **BPT [Business process testing]**.
- QTP uses the scripting language to deploy the objects, and for analysis purposes, it provides test reporting.
- Both technical and non-technical tester can use QTP.
- QTP supports multiple software development environments like Oracle, SAP, JAVA,
- We can test both desktop and web-based applications with the help of QTP.

REXX [Restructured Extended Executor]

It is an interpreted programming language, which is established at **IBM**. REXX is a high-level, structured programming language designed for reading and learning. REXX stands for **Restructured Extended Executor**.

It is used as a **scripting and macro** language. In other words, we can say that REXX is frequently used for **processing text, data, and generating reports**.

It supports various operating systems such as **MVS, OS/2, AmigaOS, VM**. And REXX can also be used as an **internal macro language** in some other software, for example, **KEDIT, ZOC terminal emulator, SPFPC, etc.**



Features of REXX

Following are the most common features of REXX:

- It supports **case-insensitive tokens**, which contain the variable names.
- By using REXX, we can access system services and commands easily.
- The REXX can support various procedures, functions, and commands which are related to a particular situation.
- It supports basic input/output advantages.
- It supports dynamic data typing with no declarations.

Advantage of performing the Mainframe testing

The advantages of performing the mainframe testing process will help us in the following aspects:

- It utilizes the resources optimally.
- It helps to evade the redundant rework.
- It enhances the user experience.
- It decreases the production downtime.
- It helps us to expands customer retention.
- And it also helps us to reduces the overall cost of IT operation.

Overview

In the mainframe testing section, we have learned the following topics:

- To test the application efficiently, the test engineer should participate in **design meetings scheduled** by the business and development teams.
- Mainframe testing is like any other testing process starting from **Requirement gathering, test design, test execution and result reporting**.
- We have understood the **mainframe attributes** such as **Multiprogramming, Time sharing, Virtual storage, Spooling, and Batch processing**.
- We have understood that mainframe testing classification is divided into two parts as **Online testing and batch testing**. The online and batch testing can implement effectively without missing any functionality mention in the Requirement specification document.
- We have understood the various **mainframe testing Challenges and approaches**.
- We have learned the most commonly used **command of mainframe testing**.
- We have understood the several **mainframe automation testing tools**, which help us enhance the process of mainframe testing.

Adhoc Testing

In this section, we will learn about Adhoc testing, types of Adhoc testing, the need for Adhoc testing, and advantage/ disadvantage of the adhoc testing.

What is Adhoc Testing?

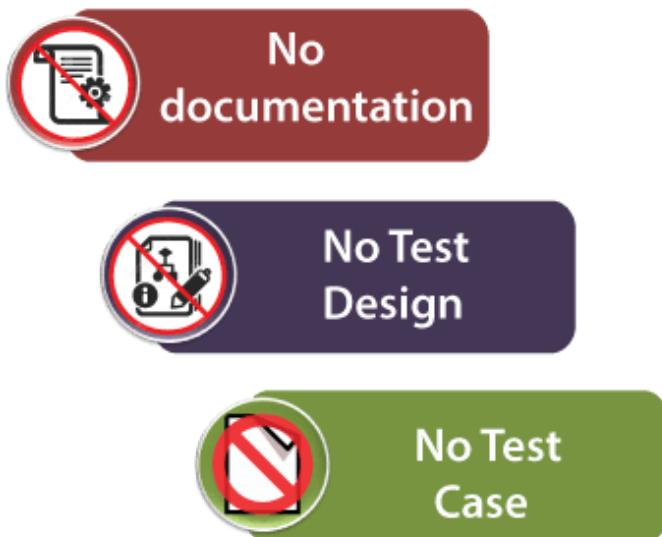
This testing we do when the build is in the checked sequence, then we go for Adhoc testing by checking the application randomly.

Adhoc testing is also known as **Monkey testing and Gorilla testing**.

It is negative testing because we will test the application against the client's requirements.

When the end-user using the application randomly, and he/she may see a bug, but the professional test engineer uses the software systematically, so he/she may not find the same bug.

Ad-Hoc Testing



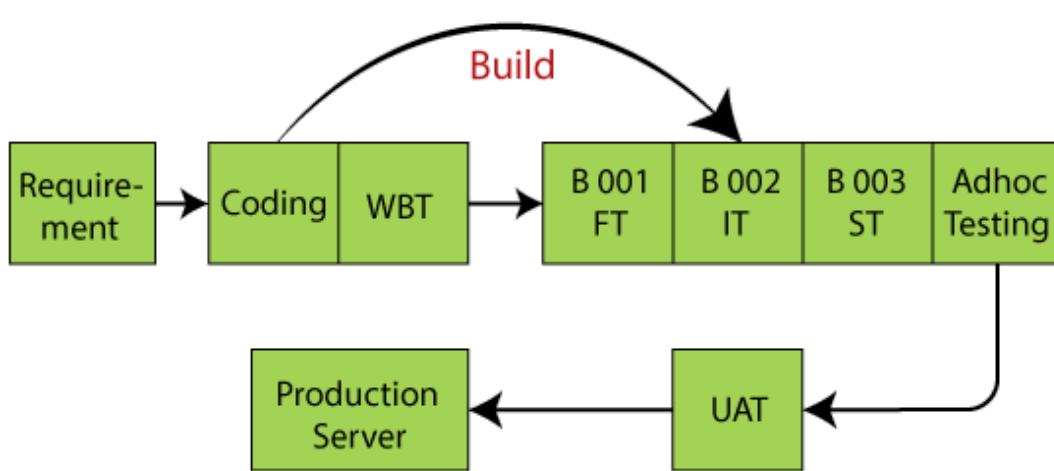
Example of Adhoc Testing

Scenario 1

Suppose we will do one round of functional testing, integration, and system testing on the software.

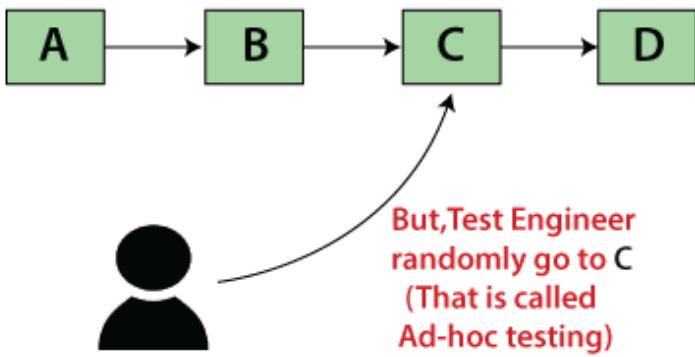
Then, we click on some feature instead of going to the login page, and it goes to the blank page, then it will be a bug.

To avoid these sort of scenarios, we do one round of adhoc testing as we can see in the below image:



Scenario 2

In Adhoc testing, we don't follow the requirement because we randomly check the software. Our need is A?B?C?D, but while performing Adhoc testing, the test engineer directly goes to the C and test the application as we can see in the below image:



Scenario 3

Suppose we are using two different browsers like Google Chrome and Mozilla Firefox and login to the Facebook application in both the browsers.

Then, we will change the password in the Google Chrome browser, and then in another browser (Firefox,) we will perform some action like sending a message.

It should navigate to the login page, and asking to fill the login credentials again because we change our credentials in another browser (Chrome), this process is called adhoc testing.

Why do we need to perform Adhoc Testing?

When the product release to the market, we go for Adhoc testing because the customer never uses the application in sequence/systematically for that sake; we check the application by going for Adhoc testing by checking randomly.

Checking the application randomly without following any sequence or procedure since the user doesn't know how to use the application, they may use it randomly and find some issues to cover this we do one round of Adhoc testing.

When we perform adhoc testing

We go for Adhoc testing when all types of testing are performed. If the time permits then we will check all the negative scenarios during adhoc testing.

Type of Adhoc Testing

Following are the types of adhoc testing:

- **Buddy testing**
- **Pair testing**

Buddy testing

Buddy testing is done with at least two members. And one member is from the testing team, and another one is from the development team.

When the unit testing is performed on the application, then only we can perform buddy testing. This type of testing helps the developer team and testing team to do their jobs.

Pair testing

In this type of testing, two testers will work together to test the software, where they can share their ideas and identify the bugs or defects in the application.

One of them will test the application, and the other one can review and analyze the application.

Advantages of Adhoc Testing

Following are some of the benefits of adhoc testing:

- Adhoc testing cannot follow any process; that's why it can be performed anytime in the software development life cycle.
- The test engineer can test the application in their new ways that helps us to find out many numbers of bugs as compared to the actual testing process.
- The developer can also execute adhoc testing while developing the module that helps them to code in a better way.
- When the in-depth testing is required in less time, adhoc testing can be performed and also deliver the quality product on time.
- Adhoc testing does not require any documentation; that's why tester can test the application with more concentration without worrying about the formal documentation.

Disadvantages of Adhoc Testing

Following are the disadvantages of adhoc testing:

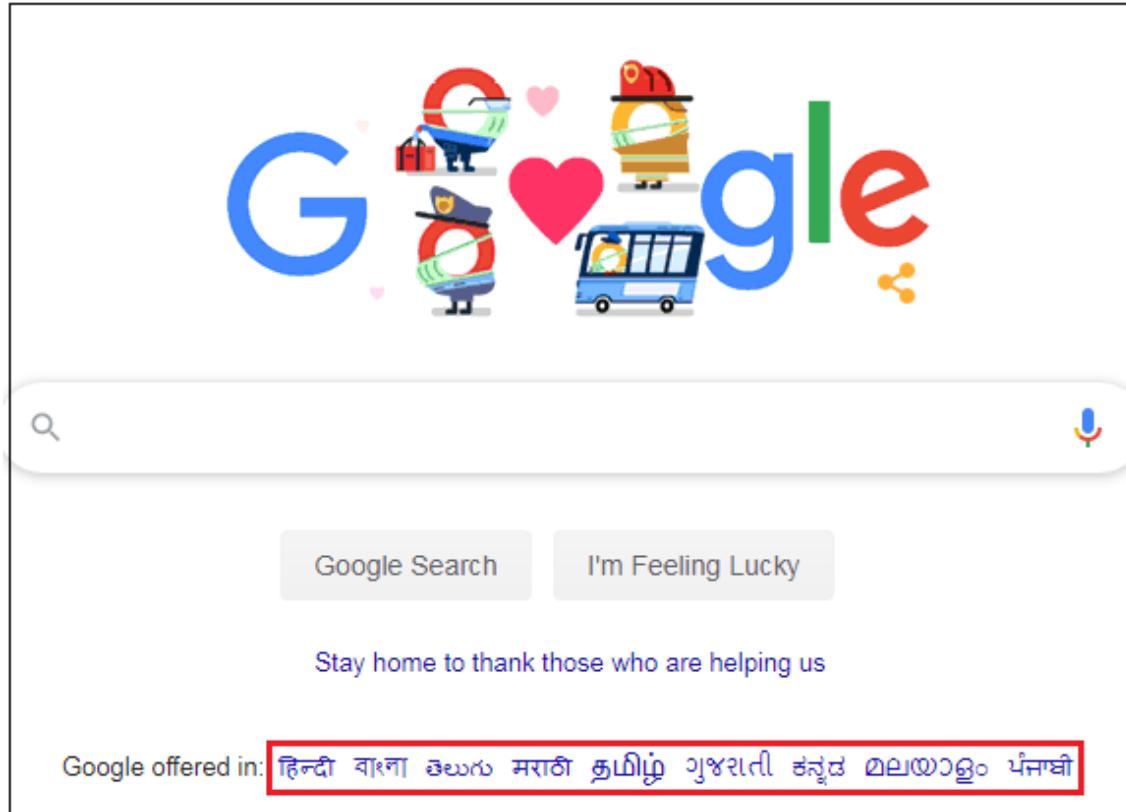
- Adhoc testing is dependent on the test engineer's product knowledge because he/she knows the flow of the application, so he/she knows where the application can collapse, and a new test engineer may not be that much familiar with the application.
- Sometimes reproducing the bug is difficult because, in this testing, we did not follow any planning.

Globalization Testing

Globalization testing is another type of software testing which is used to test the software that is developed for multiple languages, is called **globalization testing**, and improving the application or software for various languages is known as **globalization**.

This testing ensures that the application will support multiple languages and multiple features because, in current scenarios, we can see the enhancement in several technologies as the applications are planned in such a way that it is used globally.

For example, In India, the **Google.com** supports most of the languages, and it can also be retrieved by the large numbers of people of the various countries as it is a globalized application.



Purpose of Globalization testing

The primary purpose of globalization testing is as follows:

- It is used to make sure that the application is to support all the languages around the world.
- It is used for the identification of the various phases of the implementation.
- It is used to define the user interfaces of the software.
- This testing will focus on the world-wide experiences of the application.
- It is used to make sure that code can control all international support without breaking the functionality of the application.

Why we need to perform Globalization testing

We need to perform globalization testing to fulfill the following conditions:

Understanding the language vocabulary: The application should design in such a way that it can easily understand the multiple languages terms across the world.

For example

- In UK→ English
- in India→ Hindi and English

Zip Code: Suppose we lived in the UK where the Zipcode includes the alphanumeric characters as well as in India the zip code is in the 6 number digit formats. Thus if we select our country like India and entering the pin code of our state, it should accept only the 6-digit code.

So in this scenario, the software should receive the zip code based on the UK Zip code format. Therefore it is essential to make sure that the zip code functionality is working fine based on every location.

Address and telephone number format: The application should be tested in such a way that it will access the address and phone number format for multiple countries.

For example

- United kingdom→+44
- India→ +91

Types of Globalization Testing

Globalization testing is classified into two different parts, which are as follows:



Internationalization Testing

The internationalization testing is the procedure of developing and planning the software or the application (product) or file content, which allows us to localize our application for any given language, culture, and region without demanding any changes in the source code. This testing is also known as **I18N testing**, and here **18** is the number between **I** and **N** in the **Internationalization** word.

The main objective is to perform the internationalization testing, and this testing concentrates on the multiple testing such as functional testing, integration testing, usability testing, user interface testing, compatibility testing, installation testing, and validation testing.

In this testing, the application code is independent of language, and it is done at the design level.

Why we do the internationalization testing

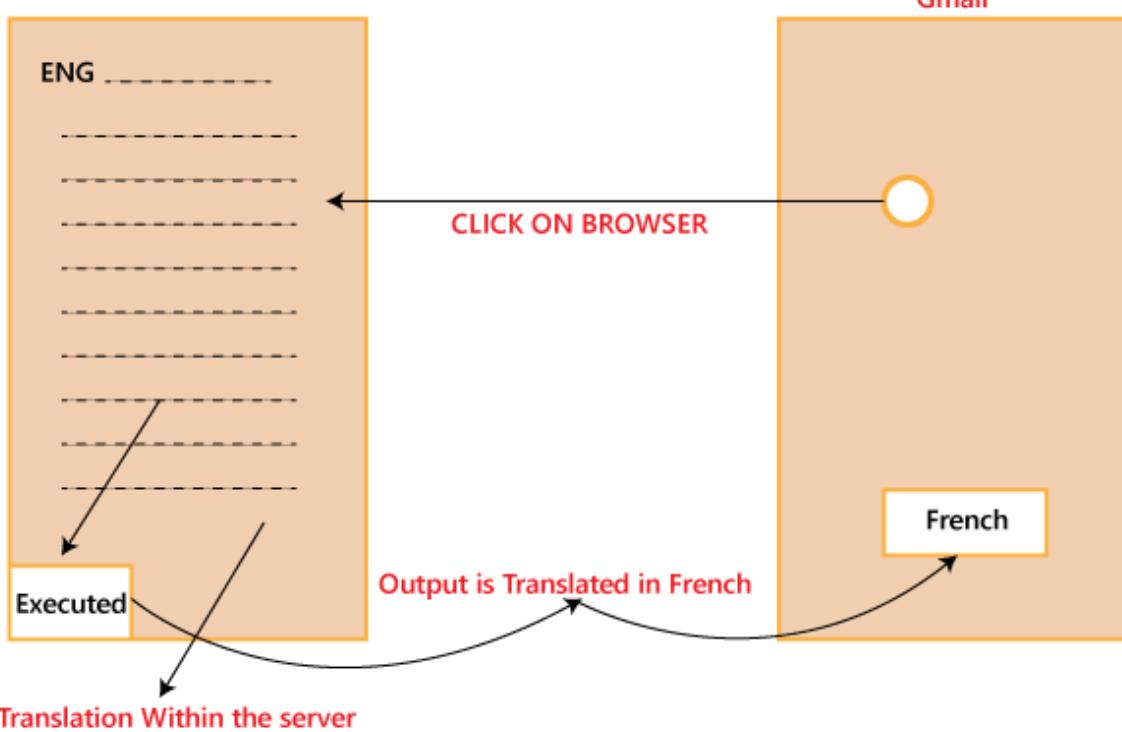
We will perform the I18N testing to check the following conditions:

- For verifying that the right content is in the correct place.
- To check an opposing effect on product quality.
- For verifying that the content is in the correct language.

Example of internationalization testing

Let see an example to understand internationalization testing.

Let say if we want our software in the French language, so we have to click on the browser. The browser will take us to the server where the code is in English, and from there, it is executed, and the output is translated into French and displayed in the French language.



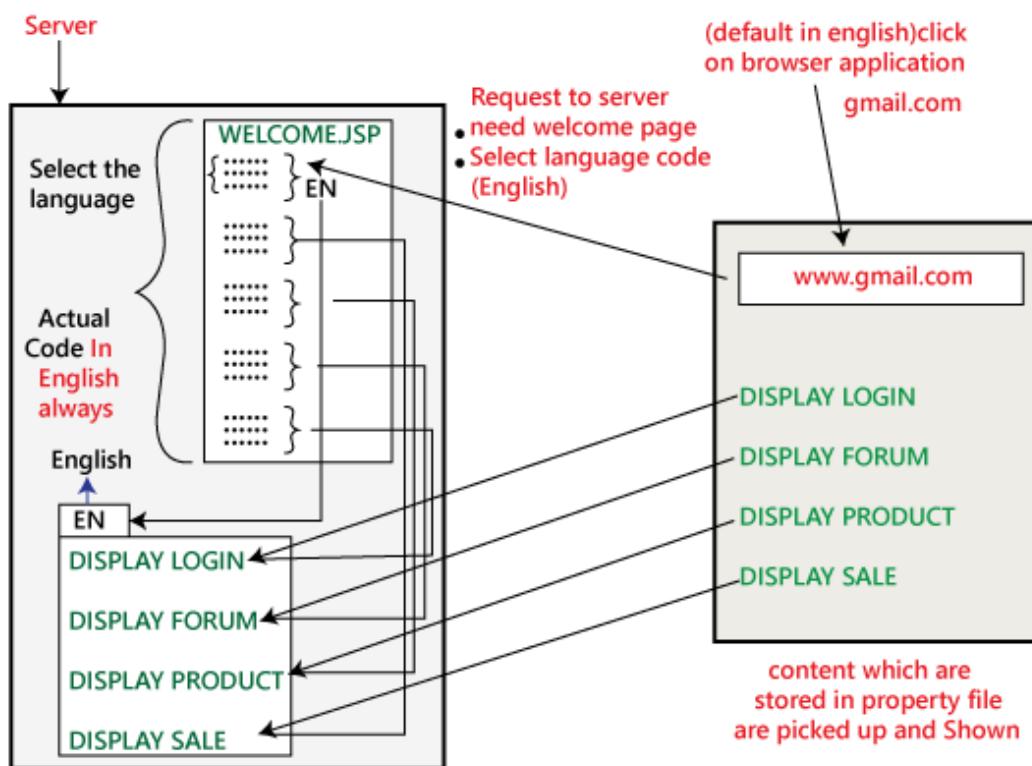
Now the **question arises how do we translate into another language?**

Can we do it with the help of the translator, NO there are some drawbacks of using the translator.

If we translate the code from English to any other language with the help of a translator, we face the below consequences:

- We will not be summoning the feeling, that's why the usage of the translators will fail.
- The meaning will change.

Let see how the internationalization testing works for the English language:



We will write one general code in English, and for all other languages, we have multiple property files. Assume that we have 6 various languages, and for that, we have 6 different property files. Let see how this will execute:

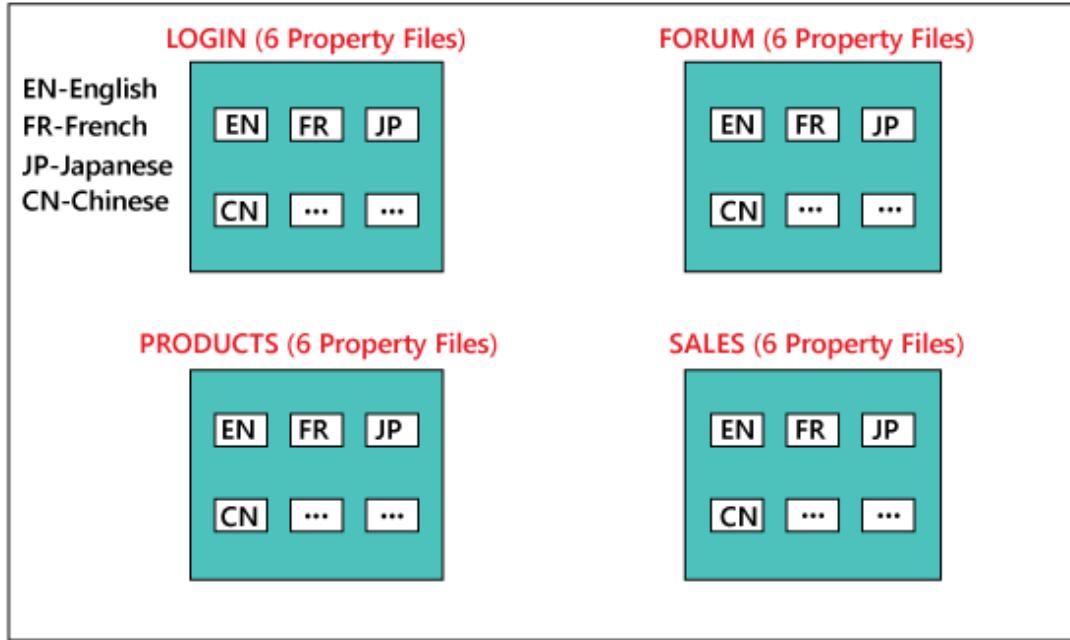
First, click on the browser, and it will take the request to the server with the following guidelines as we require a Welcome Page and start the language code EN. Then the code where the language is implemented has the property files of English. Now we select the English property file and connect to the particular link.

For example, if the next phase of coding is for the login module, then it relates to the Link1, then collects the data stored there and demonstrates it in the English language. The same process will be continuing for the other links, such as forums, sales, and products. The same process will be followed for the different languages also.

But here one has to consider that, **who wrote these properties files?**

Firstly, the developer writes the property files in English. Then the person of the experience takes the data and manually writes the property file and gives it back to the developers.

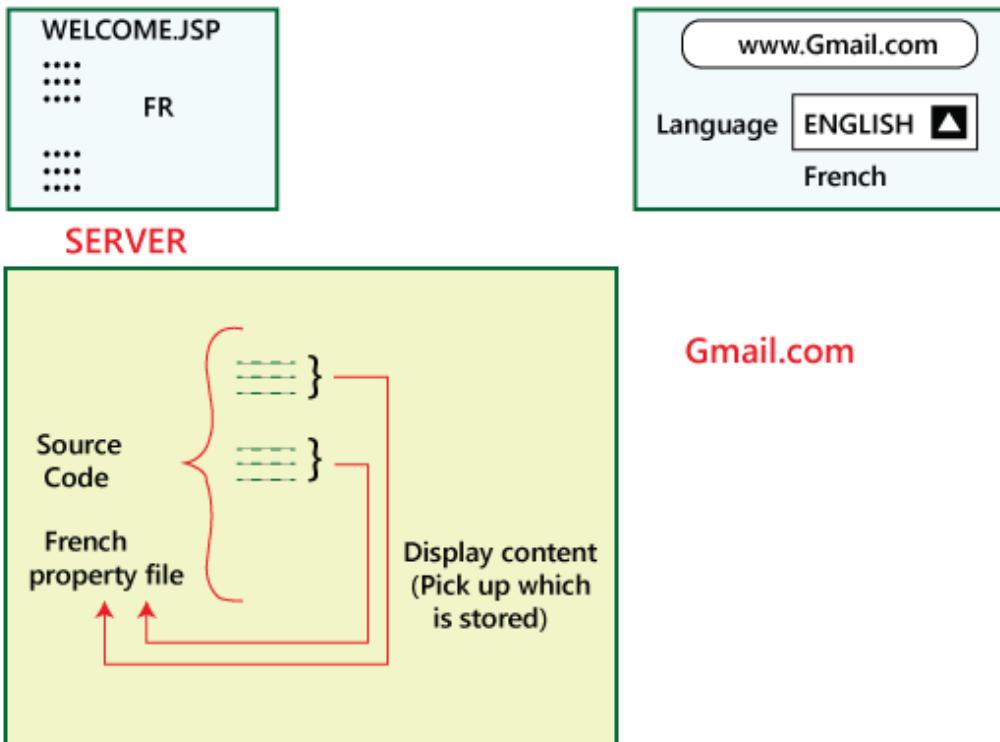
After that, the developer places this particular property in the server as we can see in the below image:



Now let us understand **how we will translate into the French language**, and we also test it.

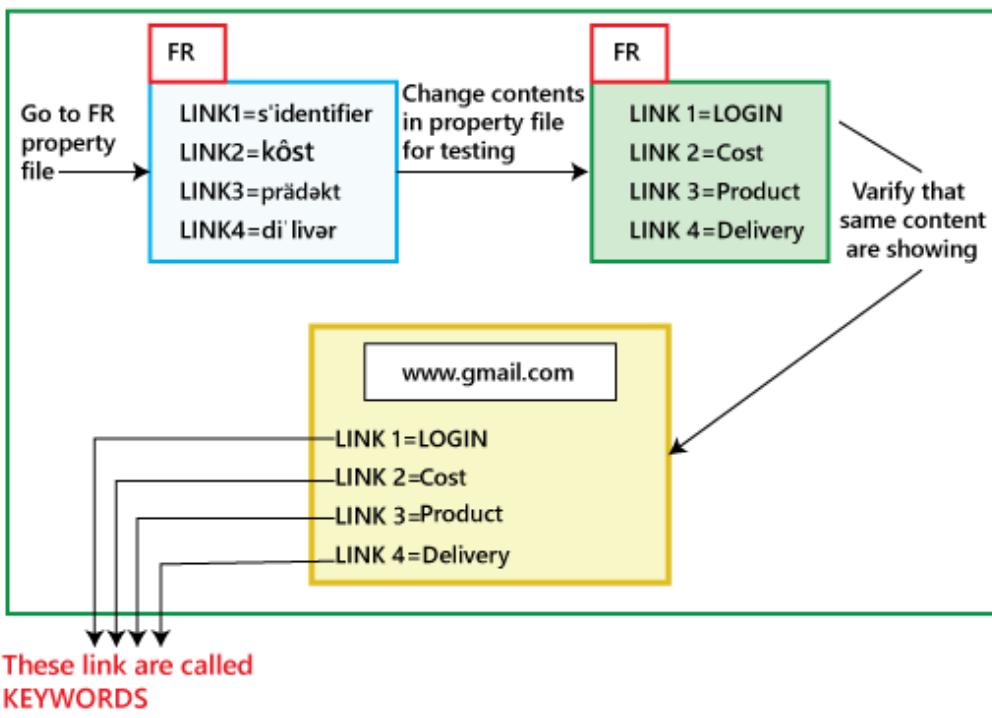
We will follow the same process as above:

- First, we click on the browser, select the language French before we further click.
- After we clicked, it leads us to the Welcome page with selected language code FR.
- Therefore, it will go to the FR property file and show the data (French), which is stored in the file based on the link it gets connected. **For example**, Link 1 to login shows the data of the login module. Likewise, for other modules such as a forum, sales, and product.



As if now, we have converted the language from English to French. So **how do we verify that it is in French or some other language?**

Move to the French property file and change the contents in that particular file as we can see in the below image:

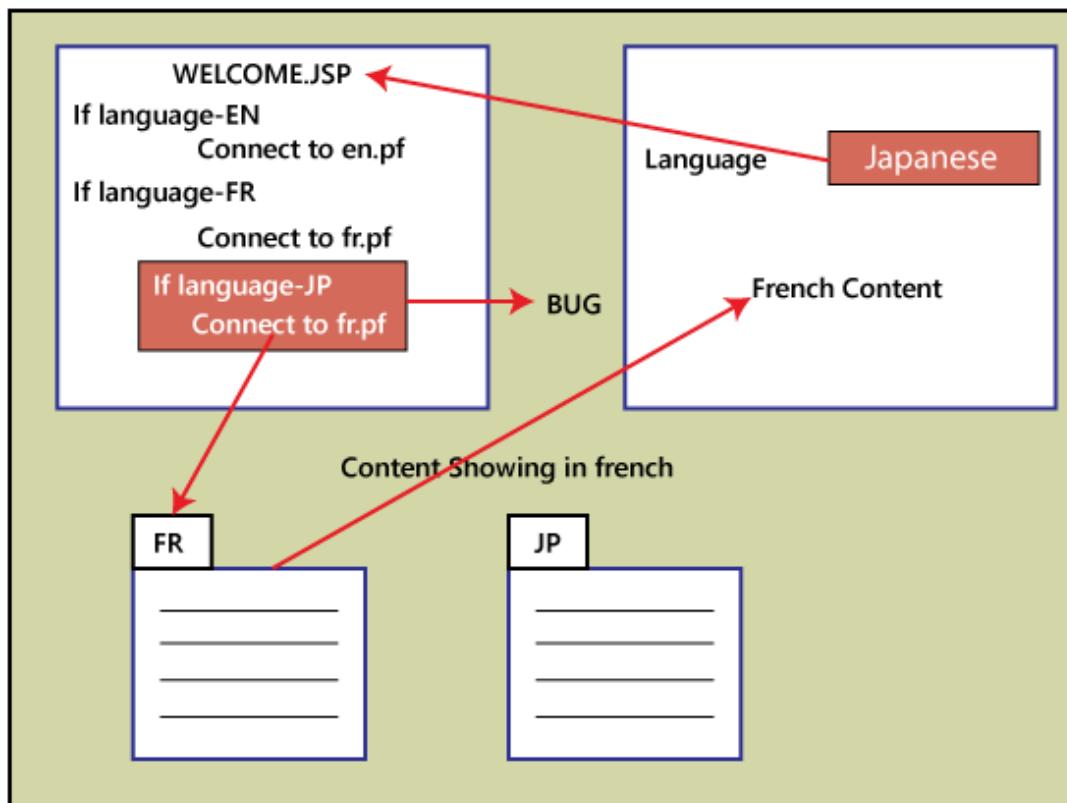


Once we have done modifications in the property file, and we select the French language again and see whether the changes we have done in the content is displaying the same or not.

If it shows the same, then the test engineer can do the testing, and we must do the modification in the property file, not in the actual code. Or if it is not showing the same, then we will find the defects which need to fix only in the code.

While doing I18N testing, we may find the below defects:

- If we select English→display in English If we choose French→ display in French If we select Japanese→display in French



Note:

Now, we will understand that if **we choose the Japanese language, and it will display the content in French?**

The above scenarios will occur if the developers will copy and paste the code and forget to modify the specific property file such as Japanese such as **jp.pf**; they continue with **Fr.pf** file; that's why the content is still present in the French language.

- The defect could be in **alignment**, so we will verify that the alignment specification for various languages is appropriately followed, like left alignment or the right alignment.

- We can check for opposite direction language, which means that how we can breakdown the words. There are two types of languages, such as **Bi-Directional and Uni-Directional**, where the **Bi-direction language** can start either **from right to left**, and the **Uni-directional language** will start either from **left to right**. Therefore, we have to check if the text is showing correctly as per the selected language or not.

Localization testing (L10N testing)

The localization testing is nothing but the format testing, where we test the format specification based on the country, region, etc. It is also known as **L10N** testing, and here **10** is the number between **L** and **N** in the **Localization** word.

The primary objective of localization is to provide a product the look and feel for a target market, no matter their culture, location, and the languages. This testing is not required to localize the product.

Let us understand the various formats testing, which we perform in the localization testing:

Date format testing

The software should be designed in such a way that it can follow the date based on its country format.

For example

- In the USA, the date format→MM-DD-YY
- In India, the date format→DD-MM-YYYY

Currency format testing

In this, we do not worry about the functionality of the format, such as \$ is converted to Rs. or not. Here we only test whether the \$ should be in the first or the last.

For example, 200\$, \$250, Rs.500 (the standard should be as per country standards)

Pin code format testing

In this, we have the countries that have the pin code with the characters such as PQ230. Checking the Pin code format is L10N testing, and checking whether PQ is translated to French is I18N testing.

The L10N testing contains the Date Format, currency format, and the Pin code format.

Image format testing

In this, we can only change the name of the image because the image cannot be changed. Therefore we must have multiple images depending on the country.

Advantages of Globalization Testing

Following are the benefits of globalization testing:

- This testing will provide the free edition of software applications or other content to multiple locations.
- This testing ensures that the application could be used in various languages without need of rewriting the entire software code.
- It will increase the code design and quality of the product.
- It will enhance the customer base around the world.
- This testing will help us to decrease the cost and time for localization testing.
- This testing will provide us more scalability and flexibility.

Disadvantages of Globalization Testing

The disadvantages of globalization testing are as follows:

- The test engineer might face the schedule challenge.
- We need the domain expert to perform globalization testing,
- We need to hire a local translator, which makes this procedure costlier.

Mutation Testing

What is mutation testing?

Mutation testing is a white box method in software testing where we insert errors purposely into a program (under test) to verify whether the existing test case can detect the error or not. In this testing, the mutant of the program is created by making some modifications to the original program.

The primary objective of mutation testing is to check whether each mutant created an output, which means that it is different from the output of the original program. We will make slight modifications in the mutant program because if we change it on a massive scale than it will affect the overall plan.

When we detected the number of errors, it implies that either the program is correct or the test case is inefficient to identify the fault.

Mutation testing purposes is to evaluate the quality of the case that should be able to fail the mutant code hence this method is also known as Fault-based testing as it used to produce an error in the program and that why we can say that the mutation testing is performed to check the efficiency of the test cases.

What is mutation?

The mutation is a small modification in a program; these minor modifications are planned to typical low-level errors which are happened at the time of coding process.

Generally, we deliberate the mutation operators in the form of rules which match the data and also generate some efficient environment to produce the mutant.

Types of mutation testing

Mutation testing can be classified into three parts, which are as follows:

- Decision mutations
- value mutations
- Statement mutations

Let us understand them one by one:



Decision mutations

In this type of mutation testing, we will check the design errors. And here, we will do the modification in arithmetic and logical operator to detect the errors in the program.

Like if we do the following changes in arithmetic operators:

- plus(+) → minus(-)

- asterisk(*) → double asterisk(**)
- plus(+) → incremental operator(i++)

Like if we do the following changes in logical operators

- Exchange P > → P<, OR P>=

Now, let see one example for our better understanding:

Original Code	Modified Code
if(p >q) r= 5; else r= 15;	if(p < q) r = 5; else r = 15;

Value mutations

In this, the values will modify to identify the errors in the program, and generally, we will change the following:

- Small value → higher value
- Higher value → Small value.

For Example:

Original Code	Modified Code
int add =9000008; int p = 65432; int q =12345; int r = (p+ q);	int mod = 9008; int p = 65432; int q =12345; int r= (p + q);

Statement Mutations

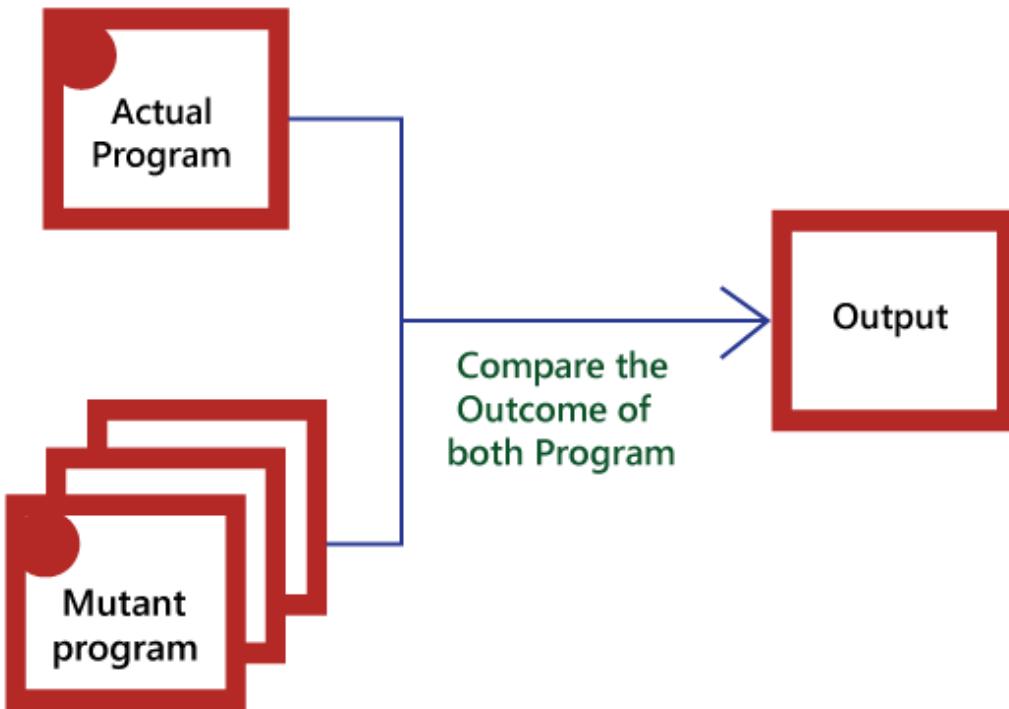
Statement mutations means that we can do the modifications into the statements by removing or replacing the line as we see in the below example:

Original Code	Modified Code
if(p * q) r = 15; else r = 25;	if(p* q) s = 15; else s = 25;

In the above case, we have replaced the statement r=15 by s=15, and r=25 by s=25.

How to perform mutation testing

To perform mutation testing, we will follow the below process:



- In this, firstly, we will add the errors into the source code of the program by producing various versions, which are known mutants. Here every mutant having the one error, which leads the mutant kinds unsuccessful and also validates the efficiency of the test cases.
- After that, we will take the help of the test cases in the mutant program and the actual application will find the errors in the code.
- Once we identify the faults, we will match the output of the actual code and mutant code.
- After comparing the output of both actual and mutant programs, if the results are not matched, then the mutant is executed by the test cases. Therefore the test case has to be sufficient for identifying the modification between the actual program and the mutant program.
- And if the actual program and the mutant program produced the exact result, then the mutant is saved. And those cases are more active test cases because it helps us to execute all the mutants.

Advantages and disadvantages of Mutation Testing

Advantages

The benefits of mutation testing are as follows:

- It is a right approach for error detection to the application programmer
- The mutation testing is an excellent method to achieve the extensive coverage of the source program.
- Mutation testing helps us to give the most established and dependable structure for the clients.
- This technique can identify all the errors in the program and also helps us to discover the doubts in the code.

Disadvantages

The drawbacks of mutant testing are as follows:

- This testing is a bit of time taking and costlier process because we have many mutant programs that need to be created.
- The mutation testing is not appropriate for Black-box testing as it includes the modification in the source code.
- Every mutation will have the same number of test cases as compare to the actual program. Therefore the significant number of the mutant program may need to be tested beside the real test suite.

- As it is a tedious process, so we can say that this testing requires the automation tools to test the application.

Security Testing

What is security testing?

Security testing is an integral part of software testing, which is used to discover the weaknesses, risks, or threats in the software application and also help us to stop the nasty attack from the outsiders and make sure the security of our software applications.

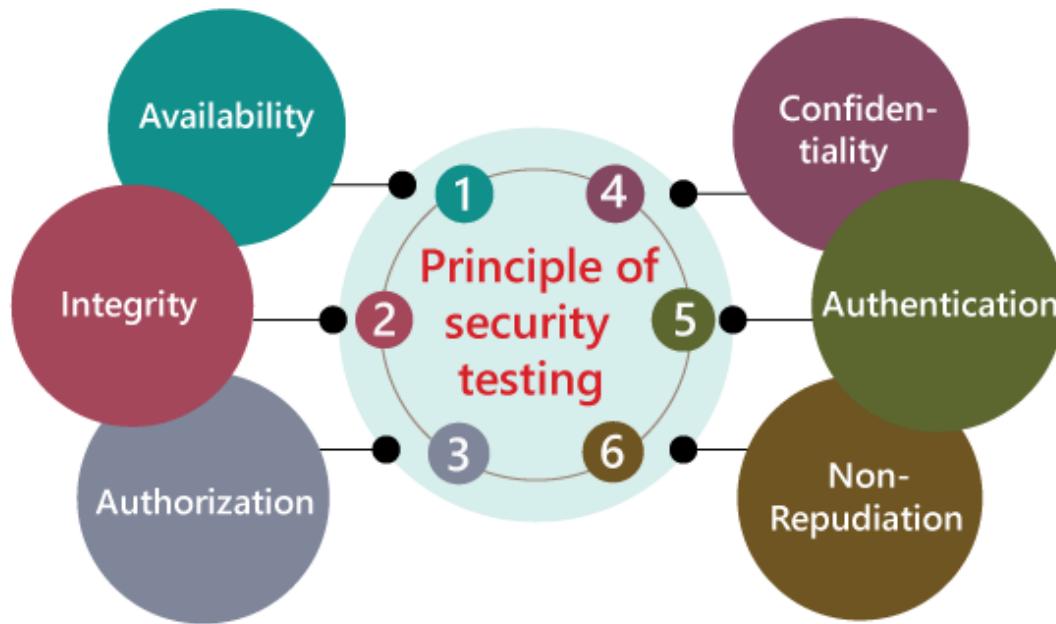
The primary objective of security testing is to find all the potential ambiguities and vulnerabilities of the application so that the software does not stop working. If we perform security testing, then it helps us to identify all the possible security threats and also help the programmer to fix those errors.

It is a testing procedure, which is used to define that the data will be safe and also continue the working process of the software.

Principle of Security testing

Here, we will discuss the following aspects of security testing:

- Availability
- Integrity
- Authorization
- Confidentiality
- Authentication
- Non-repudiation



Availability

In this, the data must be retained by an official person, and they also guarantee that the data and statement services will be ready to use whenever we need it.

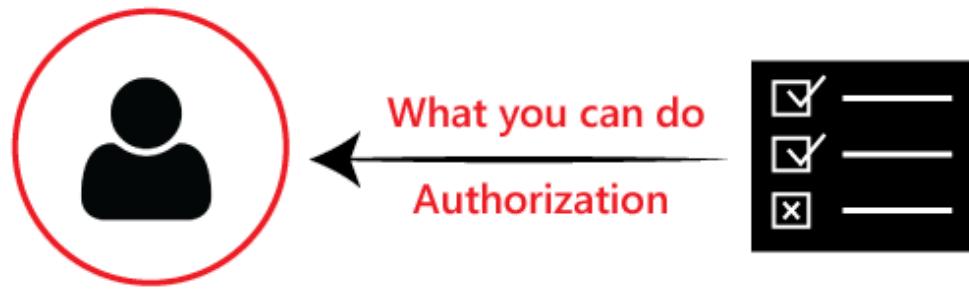
Integrity

In this, we will secure those data which have been changed by the unofficial person. The primary objective of integrity is to permit the receiver to control the data that is given by the system.

The integrity systems regularly use some of the similar fundamental approaches as confidentiality structures. Still, they generally include the data for the communication to create the source of an algorithmic check rather than encrypting all of the communication. And also verify that correct data is conveyed from one application to another.

Authorization

It is the process of defining that a client is permitted to perform an action and also receive the services. The example of authorization is Access control.



Confidentiality

It is a security process that prevents the leak of the data from the outsider's because it is the only way where we can make sure the security of our data.

Authentication

The authentication process comprises confirming the individuality of a person, tracing the source of a product that is necessary to allow access to the private information or the system.



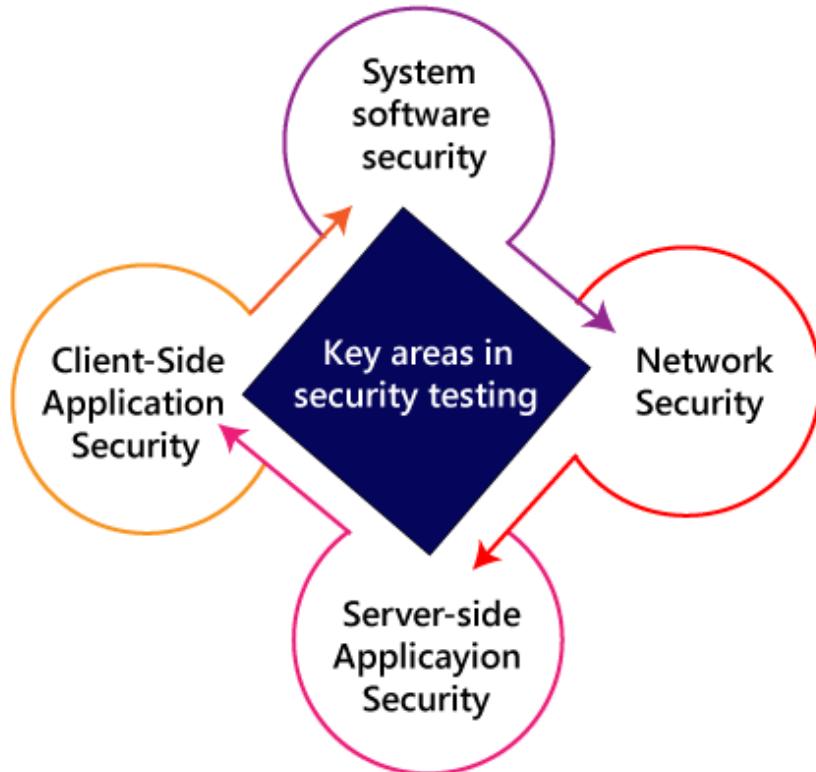
Non-repudiation

It is used as a reference to the digital security, and it is a way of assurance that the sender of a message cannot disagree with having sent the message and that the recipient cannot repudiate having received the message.

The non-repudiation is used to ensure that a conveyed message has been sent and received by the person who claims to have sent and received the message.

Key Areas in Security Testing

While performing the security testing on the web application, we need to concentrate on the following areas to test the application:



System software security

In this, we will evaluate the vulnerabilities of the application based on different software such as **Operating system**, **Database system**, etc.

Network security

In this, we will check the weakness of the network structure, such as **policies and resources**.

Server-side application security

We will do the server-side application security to ensure that the server encryption and its tools are sufficient to protect the software from any disturbance.

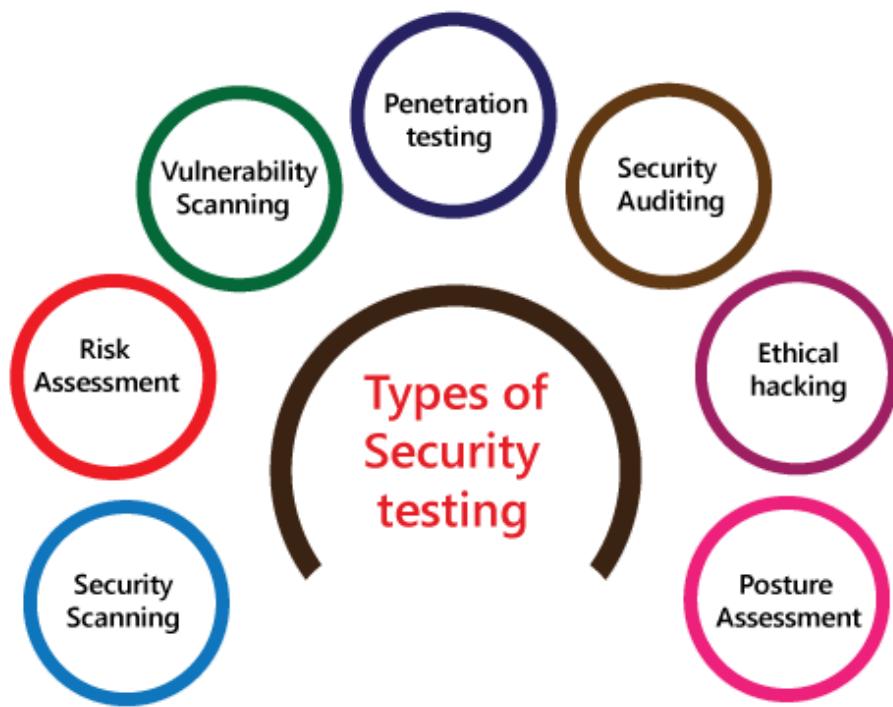
Client-side application security

In this, we will make sure that any intruders cannot operate on any browser or any tool which is used by customers.

Types of Security testing

As per Open Source Security Testing techniques, we have different types of security testing which as follows:

- **Security Scanning**
- **Risk Assessment**
- **Vulnerability Scanning**
- **Penetration testing**
- **Security Auditing**
- **Ethical hacking**
- **Posture Assessment**



Security Scanning

Security scanning can be done for both **automation testing** and **manual testing**. This scanning will be used to find the vulnerability or unwanted file modification in a web-based application, websites, network, or the file system. After that, it will deliver the results which help us to decrease those threats. Security scanning is needed for those systems, which depends on the structure they use.

Risk Assessment

To moderate the risk of an application, we will go for risk assessment. In this, we will explore the security risk, which can be detected in the association. The risk can be further divided into three parts, and those are **high, medium, and low**. The primary purpose of the risk assessment process is to assess the vulnerabilities and control the significant threat.

Vulnerability Scanning

It is an application that is used to determine and generates a list of all the systems which contain the desktops, servers, laptops, virtual machines, printers, switches, and firewalls related to a network. The vulnerability scanning can be performed over the automated application and also identifies those software and systems which have acknowledged the security vulnerabilities.

Penetration testing

Penetration testing is a security implementation where a **cyber-security** professional tries to identify and exploit the weakness in the computer system. The primary objective of this testing is to simulate outbreaks and also finds the loophole in the system and similarly save from the intruders who can take the benefits.

Security Auditing

Security auditing is a structured method for evaluating the security measures of the organization. In this, we will do the inside review of the application and the **control system** for the security faults.

Ethical hacking

Ethical hacking is used to discover the weakness in the system and also helps the organization to fix those security loopholes before the nasty hacker exposes them. The ethical hacking will help us to increase the security position of the association because sometimes the ethical hackers use the same tricks, tools, and techniques that nasty hackers will use, but with the approval of the official person.

The objective of ethical hacking is to enhance security and to protect the systems from malicious users' attacks.

Posture Assessment

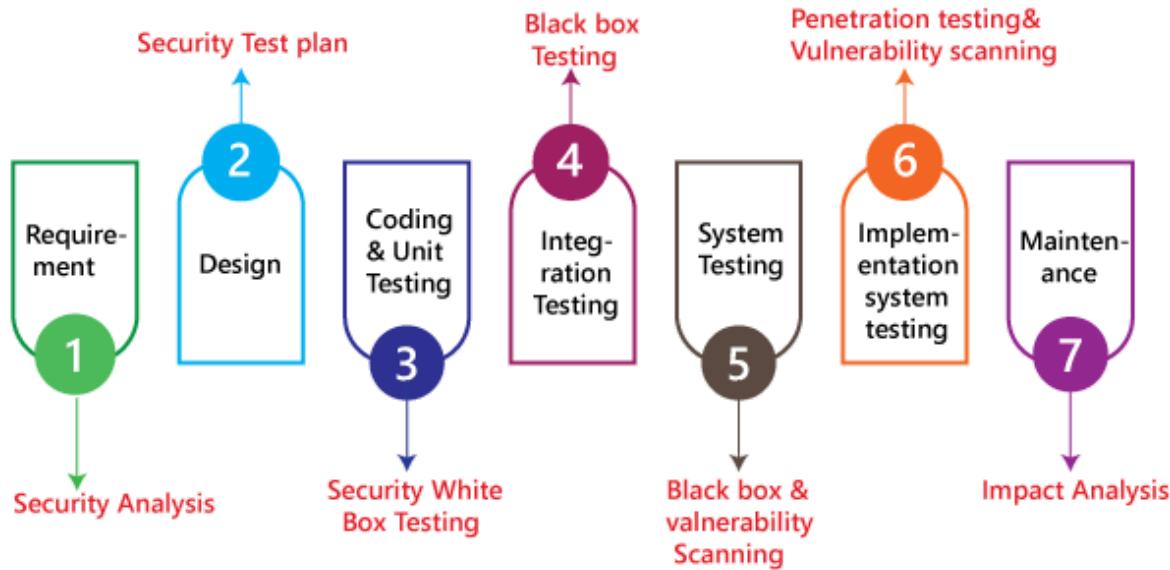
It is a combination of **ethical hacking, risk assessments, and security scanning**, which helps us to display the complete security posture of an organization.

How we perform security testing

The security testing is needed to be done in the initial stages of the **software development life cycle** because if we perform security testing after the software execution stage and the deployment stage of the SDLC, it will cost us more.

Now let us understand how we perform security testing parallel in each stage of the software development life cycle(SDLC).

Security Testing along with SDLC



Step1

SDLC: Requirement stage

Security Procedures: In the requirement phase of SDLC, we will do the security analysis of the business needs and also verify that which cases are manipulative and waste.

Step2

SDLC: Design stage

Security Procedures: In the design phase of SDLC, we will do the **security testing for risk** exploration of the design and also embraces the security tests at the development of the test plan.

Step3

SDLC: Development or coding stage

Security Procedures: In the coding phase of SDLC, we will perform the white box testing along with static and dynamic testing.

Step4

SDLC: Testing (functional testing, integration testing, system testing) stage

Security Procedures: In the testing phase of SDLC, we will do one round of **vulnerability scanning** along with black-box testing.

Step 5

SDLC: Implementation stage

Security Procedures: In the implementation phase of SDLC, we will perform **vulnerability scanning** again and also perform one round of **penetration testing**.

Step 6

SDLC: Maintenance stage

Security Procedures: In the Maintenance phase of SDLC, we will do the **impact analysis** of impact areas.

And the **test plan** should contain the following:

- The test data should be linked to security testing.
- For security testing, we need the test tools.
- With the help of various security tools, we can analyze several test outputs.

- Write the test scenarios or test cases that rely on security purposes.

Example of security testing

Generally, the type of security testing includes the problematic steps based on overthinking, but sometimes the simple tests will help us to uncover the most significant security threats.

Let us see a sample example to understand how we do security testing on a web application:

- Firstly, log in to the web application.
- And then log out of the web application.
- Then click the BACK button of the browser to verify that it was asking us to log in again, or we are already logged-in the application.

Why security testing is essential for web applications

At present, web applications are growing day by day, and most of the web application is at risk. Here we are going to discuss some common weaknesses of the web application.

- Client-side attacks
- Authentication
- Authorization
- Command execution
- Logical attacks
- Information disclosure

Client-side attacks

The **client-side attack** means that some illegitimate implementation of the external code occurs in the web application. And the data spoofing actions have occupied the place where the user believes that the particular data acting on the web application is valid, and it does not come from an external source.

Note: Here, Spoofing is a trick to create duplicate websites or emails.

Authentication

In this, the authentication will cover the outbreaks which aim to the web application methods of authenticating the user identity where the user account individualities will be stolen. The incomplete authentication will allow the attacker to access the functionality or sensitive data without performing the correct authentication.

For example, the **brute force attack**, the primary purpose of brute force attack, is to gain access to a web application. Here, the invaders will attempt n-numbers of usernames and password repeatedly until it gets in because this is the most precise way to block brute-force attacks.

After all, once they try all defined number of an incorrect password, the account will be locked automatically.

Authorization

The authorization comes in the picture whenever some intruders are trying to retrieve the sensitive information from the web application illegally.

For example, a perfect example of authorization is **directory scanning**. Here the directory scanning is the kind of outbreaks that deeds the defects into the webserver to achieve the illegal access to the folders and files which are not mentioned in the public area.

And once the invaders succeed in getting access, they can download the delicate data and install the harmful software on the server.

Command execution

The command execution is used when malicious attackers will control the web application.

Logical attacks

The logical attacks are being used when the DoS (denial of service) outbreaks, avoid a web application from helping regular customer action and also restrict the application usage.

Information disclosure

The information disclosures are used to show the sensitive data to the invaders, which means that it will cover bounts that planned to obtain precise information about the web application. Here the information leakage happens when a web application discloses the delicate data, like the error message or developer comments that might help the attacker for misusing the system.

For example, the password is passing to the server, which means that the password should be encoded while being communicated over the network.

Note:

The web application needs more security regarding its access along with data security; that's why the web developer will make the application in such a way to protect the application from **Brute Force Attacks, SQL Injections, Session Management, failure to Restrict URL Access and Cross-site scripting (XSS)**. And also, if the web application simplifies the remote access points, then it must be protected too.

Here, **Session management:** It is used to check whether the cookies can be re-used in another computer system during the login stage.

SQL injection: It is a code injection approach where the destructive SQL Statements are implanted into some queries, and it is implemented by the server.

Cross-site scripting (XSS): This is the technique through which the user introduces client-side script or the HTML in the user-interface of a web application and those additions are visible to other users.

Myths and Facts of Security testing

Here, we will discuss the Myths and facts of security testing:

Myths	Facts
There is no profit on an asset in security testing.	Security Testing can highlight the parts of enhancement, which help us to increase productivity and decrease interruption and also allow the maximum output.
Nowadays, the Internet is not safe, and we will be buying the hardware or software to save the business and protect the system.	Here, the fact is rather purchasing any software or hardware, the company first understands security and then follows the security process.
If we have a small business and we do not require a security procedure.	In that case, the fact is every organization needs a security procedure.
The only technique to secure is to disconnect it.	The one and the most excellent way to protect an association is to identify the Perfect Security. Here Perfect security can be accomplished by performing the implementation, compared with business, permitted, and manufacturing validations.

Security testing tools

We have various security testing tools available in the market, which are as follows:

- SonarQube
- ZAP
- Netsparker
- Arachni
- IronWASP

For more information about these tools, refer to the below link:

<https://www.javatpoint.com/security-testing-tools>

Conclusion

For an application or the software, it is necessary to perform security testing to verify that the sensitive information is still private. In software testing, the security testing is essential because it helps us to save our necessary data ultimately. In this, the test engineer will act as an invader and test the system or detect the security defects.

Accessibility testing

In software testing, **accessibility testing** is widely used to check the application for **disabled persons** and make sure the developer will create the application which can be accessible by all types of users, like a regular user and physically challenged (color blindness, learning disabilities, and so on).

In this section, we will discuss **accessibility testing**, how we **perform accessibility testing**, the **objective of using this testing**, and **tools of accessibility testing**.

What is accessibility testing?

Accessibility testing is another type of **software testing** used to test the application from the physically challenged person's point of view. Here the physical disability could be old age, hearing, color blindness, and other underprivileged groups. It is also known as **508 compliance** testing. In this, we will test a web application to ensure that every user can access the website.



For accessibility testing, we have some assured rules and regulations, which need to be followed as well.

The Law for Accessibility testing:

- **Web content accessibility guidelines:** These strategies are established to serve a purpose, which helps us to increase the user-friendliness of a website.
- **Rehabilitation Act, section 504, and section 508:**

Section 504: This section will help people with disabilities by providing workspace access, education, and other organizations.

Section 508: Section 508 will help those people by giving access to technology.

- **Americans with disabilities act (ADA):** The ADA rule says that all the domains, such as schools and organizations, public buildings should make the tools that everyone uses.

Individuals who are physically challenged will use assistive tools that help them to work on the software product. Let see some of the tools which are available in the market:

- **Special keyboard:** We have some special keyboards where the users can quickly type, and these keyboards are specially designed for them who have motor control problems.
- **Screen reader software:** This type of software is used to read out the text, which is shown on the screen.
- **Speech Recognition Software:** The speech recognition software will change the spoken word to text and works as an input to the computer system.

- **Screen Magnification Software:** This type of software is designed to help the vision-impaired persons because it will expand the screen and make the reading easy.

Example of accessibility testing

Let us assume that if a blind person uses the internet, and clicks on anything, the response connected into the voice, and the person can hear that and then uses it. The response should be read by the browsers and commented invoice.

Whatever the response is sent to the browser, it can be easily read, and the application or the software should be designed like that. The response should be immediately connected to voice. Therefore the blind person can easily access it.

The application should be designed in such a way that even the physically impaired person could be able to access the application without facing any difficulties.

The accessibility testing has many rules which could be followed while developing the software or the application. Some of the essential strategies are as follows:

- The red and green color objects should not be used or displayed.
- All the comments should have Alt tags.
- The application should be able to access all the components with the help of keywords.

Purpose of Accessibility testing

The primary purpose of Accessibility testing is to accommodate people who have disabilities like:



- **Hearing Deficiency:** In this, the person is not able to hear or hear clearly and has sensory issues such as hearing disabilities and deafness.
- **Learning Impairment:** The people who are facing reading difficulties.
- **Physical Disabilities:** In this type of disability, the people are not capable of using the Keyboard or the Mouse with one hand and facing the problem in hand activities, muscle detention, and reduced motor abilities.
- **Visual Impairments:** The visual or vision disabilities define that when a person has complete blindness, poor vision abilities, color blindness, and flashing effect problems and visual strobe.
- **Cognitive Deficiency:** In this, the person will have poor memory, not able to recognize more complex scenarios, and learning difficulties.

Myths and facts about Accessibility Testing

Myths	Facts
Accessibility testing is only for physically challenged people.	All types of users can use the accessibility testing as they enhance the credibility of software.

We are modifying the unapproachable application to the available use, which causes us lots of time and money?	We can work on the typical requirements that are essential for the challenged users because sometimes, it is not required to integrate all the modifications at one time.
Accessibility testing is costly.	This testing is not costly if we recognize the accessibility issues at the design phase besides the extensive testing, which can help us to decrease the cost and save lots of rework as well.
Accessibility testing is basic and tedious process to perform.	Here, we can prepare our application in such a way that all types of users can use it.

How to perform accessibility testing

We can perform accessibility testing both **manually** and with the help of **automation** as well. First, we see that how we perform accessibility testing **manually**:

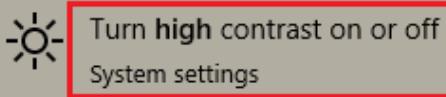
For accessibility testing, we have many tools available in the market, but while using it, we may face some problems **such as budget, less knowledge**, and so on. To overcome these issues, we will perform accessibility testing manually.

Let us see some scenarios, where we test the accessibility of the application manually:

- **Modifying the font size to large:** We can use the large font size and check the availability.
- **Testing for captions:** Here, we will test that a caption should be visible and also ensure that it is expressive. As we know that while we are accessing the Facebook application, sometimes the images and videos take lots of time to load, where the captions will help us to understand what is in the pictures and video.
- **By deactivating the style:** We can disable the method and test if the content of the table is accurately lined up or not.
- **We can use high contrast mode:** If we can use high-contrast mode, we can highlight the website's content. When we turn the high contrast mode in our windows, the content of the site gets highlighted automatically as it turns into white or yellow, and the background turns black.

To turn on **high contrast mode**, search the **high contrast mode** in the search box of the start menu on your system as we can see in the below image:

Best match



Settings

- 💡 Ease of Access **high** contrast settings >
- 📝 Highlight misspelled words >
- 💡 Change **high** contrast theme >
- ⟳ Sync your personalization settings >
- 💻 Windows HD Color settings >
- 🔊 Show the Narrator cursor on the screen >

Search the web

- 🔍 **high** - See web results >

high

Turn high contrast on or off

System settings

 Open

Related settings

Themes and related settings

Here, first we **turn-on the high-contrast**, and we can also select a **theme** from the given drop-down list as we choose the **high contrast** theme as we can see in the below image:

Settings

Home

Find a setting



Ease of Access

Vision

Display

Cursor & pointer

Magnifier

Color filters

High contrast

Narrator

Hearing

Audio

Closed captions

High contrast

Turn on high contrast



Press left Alt + left Shift + Print Screen to turn high contrast on and off.

Choose a theme

High Contrast Black



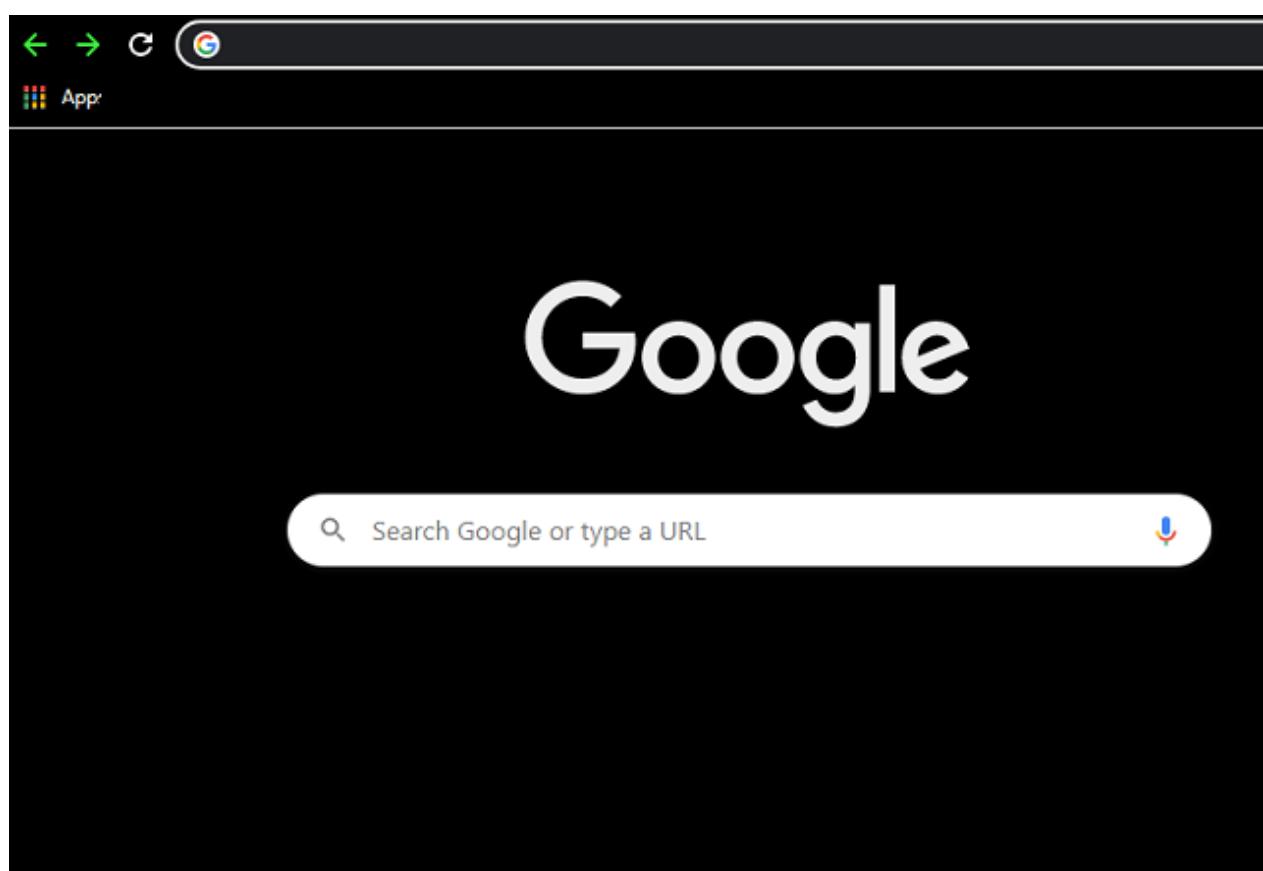
Select a colored rectangle to customize high contrast colors



Apply

Cancel

After modifying the settings, our browser will look like this:



- **Skip navigation:** We can also skip the navigation sometimes as it is helpful for people who have the motor incapacities. We can change our effort to the top of the page by clicking on the **Ctrl + Home**

- **Turn-off the CSS [cascading style sheet]:** Generally, the Cascading style sheet is used to define the appearance of the document. If we turn off this, we can test the text presentation, text style, and background color of the application.
- **Use the field label:** If we use the field label as it will help us in filing a form because this, we can see the template and fill out the required information while we ordering something online and login.
- **PDF document:** In we can try to save the PDF file in the form of text and test whether the order for the content is kept or not.
- **Content scaling:** We can check the image readability while zooming out it.

Automation method

Generally, the Automation technique is used for various testing methods. The **automation testing** process contains multiple tools to perform accessibility testing.

Some of the most commonly used tools are as follows:

- **Hera**
- **Wave**
- **Accessibility valet**
- **TAW**
- **aDesigner**
- **WebAnywhere**
- **Web accessibility toolbar**

Hera

The Hera tool is to test the accessibility of Web pages based on the WCAG requirement. It is used to do an initial set of tests on the page and also finds the automatically detectable issues. It will help us in manual modification by highlighting the parts of the page, providing guidelines on how to perform the tests, and also verify the style of the application which comes with a multilingual preference.

Wave



It is a web accessibility tool that is introduced by WEBAIM. It is an open-source tool that automatically tests the web page for several phases of accessibility. It is a suite of assessment tools which ensure the writers make their content more accessible to those who are physically challenged.

It is used to identify the WCAG (web content accessibility toolbar guidelines) issues but also simplifies the human assessment of web content. The WAVE tool will make sure that our accessibility reports are protected and hundred percent isolated.

For more information about WAVE, refers to the below link:

Accessibility Valet

The accessibility valet tool is used to test the web pages besides the Web Content Accessibility Guidelines [WCAG] agreement. This tool includes various features such as:

- It is a scripting tool.

- It will display the detailed reports to the developers.
- It will provide the automatic cleanup.
- It will help us to convert the Html to Xhtml.
- This tool will also provide the meta-data for the semantic web and WWW.

TAW

It is a tool that will help to explore the website in agreement with the W3c web accessibility strategies and also display the accessibilities problems. It is an online tool that defines the accessibility of our website. The web accessibility test problem is further classified as Priority 1, priority two, and priority 3. This tool will also provide the subsets of WCAG 1.0.

aDesigner

The aDesigner tool is established by **IBM** that helps us to understand the visually impaired persons. Thus the designer can recognize the necessities of Impairment people and create the applications.

WebAnywhere

It is an open-source tool, which is a web-based screen reader for the web. The screen reader allows blind people to access the network from any computer system. This tool will help the readers to read the web page as it is easily accessed on any device.

Web accessibility toolbar

It is an extension of Opera or Internet Explorer, which allows as designing web pages with the help of suitable features. The most important feature of this tool is **GreyScale** that helps to identify the small contrast spots in the design.

Conclusion

In the end, we can say that accessibility testing is testing where each user can use software or the application. The test engineer could perform the accessibility testing from each user's points of view because the test engineer's purpose of testing an application is to verify that all the strategies are fulfilled or not. All the users should easily access that application.

Structural Testing

In this section, we are going to understand **structural testing**, which is a significant part of **Software testing**.

And we also learn about its needs, **types of structural testing**, **tools compatible for Structural testing**, **advantage**, and **disadvantage**.

Introduction of Structural Testing

Another type of **software testing** technique is **Structural testing**, which is used to test the internal design of the software or structure of the coding for the particular software.

In this testing, the development team members are included in the testing team to execute the software's internal design. The working of structural testing is opposite to **Behavioral testing**.

In other words, we can say that structural testing tests the different features of an application based on its types.

Structural testing is also known as white-box testing, **glass box testing**, and **clear-box testing**. Developers mostly implement it to identify the issue and fix them quickly.

The structural testing process requires an in-depth knowledge of the programming language and is opposite to Functional Testing.

The knowledge of the code's internal executions and how the software is implemented is a necessity for the test engineer to implement the structural testing.

Throughout the structural testing, the test engineer intends on how the software performs, and it can be used at all levels of testing.

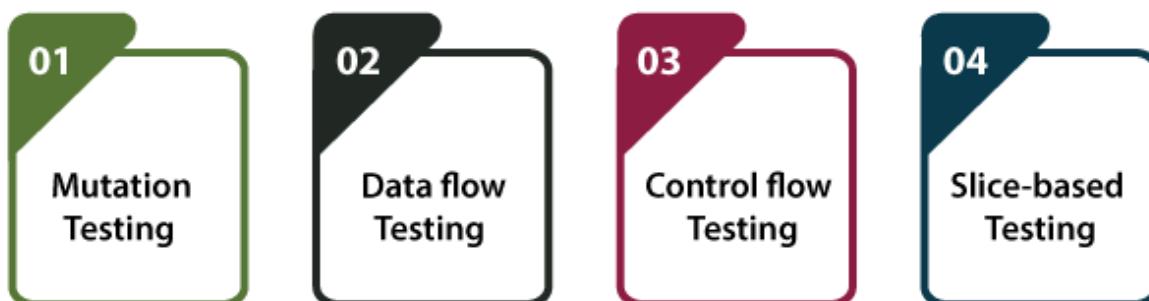
For example, the coverage of menu options or significant business transactions could be the system's structural element or acceptance testing.

Types of Structural Testing

Structural testing is divided into four different categories, which are as follows:

- **Mutation testing**
- **Data flow testing**
- **Control flow testing**
- **Slice-based testing**

Types of Structural testing



Mutation testing

- It is used to check the quality of the test case that should fail the mutant code.
- Mutation testing involves the development of new tests to be implemented on the software for its testing process.
- When we identify various errors, it implies that either the program is correct or the test case is inefficient in locating the fault.

- In the mutation testing, the developers make small modifications to the previously accessible software tests and generate a mutant of the old software test.
- It used to cause an error in the program, which implies that the mutation testing is performed to evaluate the test case's productivity.

Refers to the below link for detailed information about the Mutation testing: <https://www.javatpoint.com/mutation-testing>

Data flow testing

- It is a group of testing approaches used to observe the control flow of programs to discover the sequence of variables as per the series of events.
- It implements a control flow graph and analysis the points where the codes can change the data.
- If we execute the data flow testing technique, the information is kept safe and unchanged during the code's implementation.

Refers to the below link for the detailed information related to the data flow testing: <https://www.javatpoint.com/data-flow-testing-in-white-box-testing>

Control flow testing

- The **control flow testing** is the basic model of **Structural testing**.
- It is to check the implementation order of commands or statements of the code over a control structure.
- In the control flow testing, a specific part of an extensive program is selected by the test engineer to set the testing path.
- Generally, the control flow testing technique is used in unit testing.
- In this testing, the entire test is based on how the control is executed during the code.
- The complete information of all the software's features and logic is necessary to execute the control flow testing.

For detailed information about the Control flow testing is, refer to the below link: <https://www.javatpoint.com/control-flow-testing-in-white-box-testing>

Slice-based testing

- It was initially created and established to keep the software.
- The basic idea is to sort the complete code into small chunks and then evaluate each portion carefully.
- The slice-based testing is very beneficial for the maintenance of the software along with fixing the application too.

Note: The developers can use these four types of structural testing as per their requirements.

Structural Testing Tools

Like other testing has their tools, structural testing also contains some open-source and commercial tools that have their functionality.

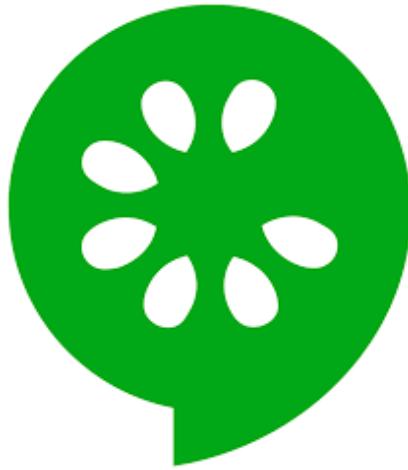
Some of the most commonly used tools for structural testing are as follows:

- **Cucumber**
- **JBehave**

- **Cfix**
- **JUnit**

Let understand them one by one in details:

Cucumber



It is a widely used tool for behavior-driven Development as it delivers an easily understandable testing script for **system acceptance** and automation testing.

It is a software tool used by the test engineer to develop test cases for the testing of the software's behavior.

For more information about the cucumber tool, we can refer to the following link:
<https://www.javatpoint.com/cucumber-testing-introduction>

JBehave

It is a framework for BDD (**Behaviour-Driven Development**). The BDD is a development of **TDD (test-driven Development)** and acceptance-test-driven design. It is planned to create these practices more accessible and spontaneous to beginners and professionals alike.



Features of JBehave

Some of the commonly used features of JBehave are as follows:

- It is purely executing in Java programming language, which plays well with Java-based enterprises.
- In this, we can write the user stories in **JBehave syntax/ Gherkin syntax**.
- The user stories can be implemented as **JUnit**
- It provides the Annotation-based configuration and steps class specifications.
- It allows stories to be executed through Ant task.

Cfix

Another structural testing tool is **Cfix**, an XUnit testing framework supported by the **C/C++ programming language**.

In this tool, the unit tests are compiled and linked into a DLL. It has been designed to work well in combination with **Windows Debuggers** such as **Visual Studio** and **WinDBG**.

Characteristic of Cfix

Following are the commonly used features of Cfix:

- It supports development of both user and kernel mode unit tests.
- The failing test case can be highly customized in case of implementation.

JUnit



JUnit is one of the essential tools of structural testing. It is an open-source unit testing framework, which was written in Java language.

It will help us to enhance the developer's efficiency, which provides the consistency of the development code and reduces the time of the debugging.

For more details about the **JUnit**, refers to the below link: <https://www.javatpoint.com/junit-tutorial>

Advantages and Disadvantages of Structural Testing

Below are the benefits and drawback of structural testing:

Advantages of Structural Testing

The benefits of Structural testing are as follows:

- Structural testing does not require a lot of manual work as it is an automated process.
- Structural testing is not a time-consuming process.
- All the early defects can easily be identified
- It removes the dead code (extra code) or statements easily.
- It provides easy coding and implementation.
- It delivers detailed testing of the software.

Disadvantages of Structural Testing

The **drawback** of the structural testing are as follows:

- To perform structural testing, in-depth knowledge of programming languages is required.
- Even though structural testing is automatic, it might turn out very difficult because it involves training in the tool used for testing.
- It is expensive in respect of money because sometimes resources are necessary to efficiently perform structural testing.
- There is also a small chance that some commands, statements or branches could be missed unintentionally.

Overview

In this tutorial, we have understood structural testing, the type of structural testing, advantages and disadvantages.

After learning all the specific topics, we can easily conclude that Structural testing, which is also called **White-box testing, glass box testing, and clear box testing** is used to verify the structure of code.

And apart from that, we can say that while executing the different types of software testing, there is no guarantee of 100% efficiency of the product. Therefore, it is always helpful if we associate different categories of testing and methods.

The various structural testing types, which we have looked at, such as data flow testing, mutation testing, slice-based testing, and control flow testing, could be copied back to errors such as:

- Mutation testing (using the wrong operator).
- Data flow testing (referencing a variable before using it).

In case someone is looking to use the structural testing methods, they need to consider both the benefits and drawbacks of the structural testing.

Furthermore, they need to take care of the fact that structural testing is implemented successfully.

Volume Testing

In this section, we are going to understand **Volume testing**, which helps us to check the behavior of an application by inserting a massive volume of the load in terms of data.

And we also learn about its process, why we need to perform the volume testing, the objective of volume testing, multiple examples, various attributes of volume Testing, advantages, and disadvantages.

Introduction of Volume Testing

Volume testing comes under **software testing**. It helps us to check the behavior of an application by inserting a massive volume of the load in terms of data is known as **volume testing**.

In volume testing, we will concentrate on the number of data rates than the number of users. It is also called **Flood testing**.

It is executed to analyze the effect on the system's response time and behavior when the volume of data is enhancing in the database. In this testing, a massive volume of information is acted upon by the software.

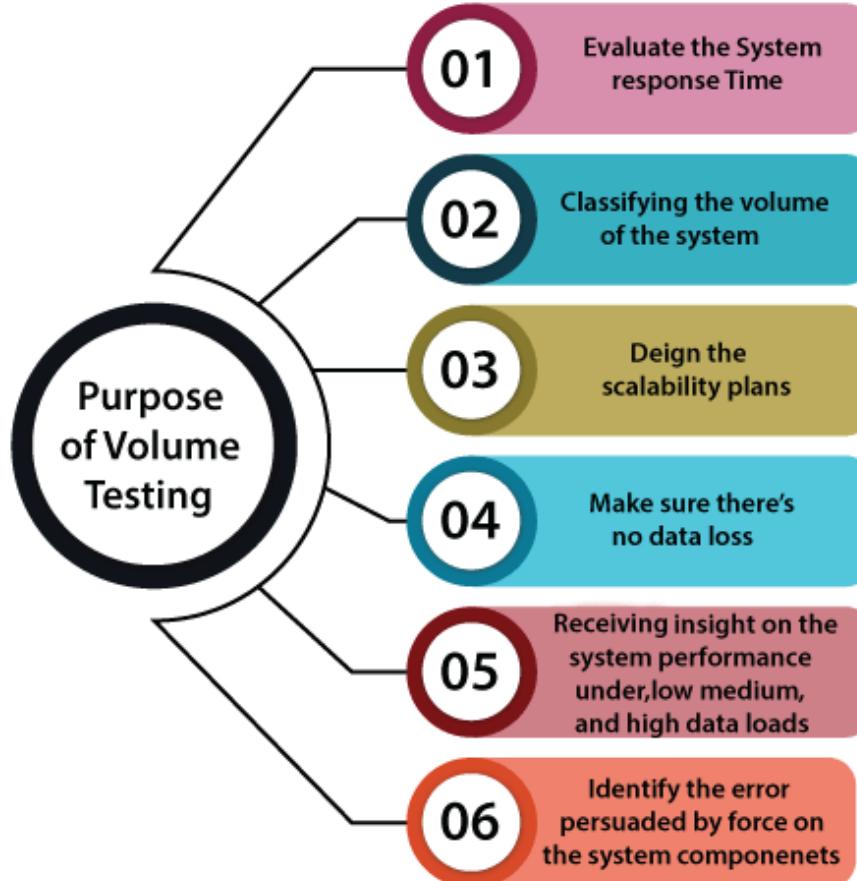
It is required to certify that the system is not at risk of overflow or data security issues.

Note: Volume is a capacity while Load is a quantity, i.e., load testing means no. of users, and volume testing implies the amount of data.

Purpose of Volume Testing

The initial objective of implementing the **volume testing** is to discover system performance with collective volumes of data in the database.

Let see some other objectives of performing volume testing:



- **Evaluate the system response time**

We can evaluate the system response time as we know that the Volume testing is productive, and it helps the **organization** executives to confirm that software performance is not interrupted.

- **Classifying the volume of the system**

If we identify the system's volume, it helps us to simplifies the scalability development which is very useful while preparing the contingency plans.

And the Volume testing gives understandings to developers, which helps them forecast how much data the system can proceed with without any failure.

Design the scalability plans

We can easily design the scalability plans by performing volume testing. As we learn that the volume tests help business executives understand if we enhance the size or accumulating more components to support the system.

- **Make sure there's no data loss**

In software testing, only through volume testing the project team can guarantee that there is no data loss as the burden on the system grows and the database's size.

- **Receiving insights on the system performance under low, medium, and high data loads**

The next objective of implementing the volume testing is to measures the system's performance under **low and medium** loads to ensure it works. And the risk of data loss is expanded under high data loads.

- **Identifying the errors persuaded by force on the system components**

The purpose of doing the volume testing is to find issues that only display when the **data load upsurges, higher response time, system failure, or security activities**.

Features of Volume Testing

It contains various features that are essential while executing the volume testing:

- In the development stage, only a small amount of data is tested.
- The software's execution is weakening with the passing time as there is a vast amount of data over time.
- While performing the volume testing, we make sure that there is no data loss because if we lose any data, we might miss some vital information.
- Mostly, a test data generator creates the test data, which needs to be logically accurate.
- To analysis the performance of the system, we can use the test data.
- The application's response time is tested during the volume testing and whether the system responds within a fixed time or not.
- In volume testing, we also test that the data is stored appropriately or not because if the information is not stored correctly, then it is restored accordingly in the proper place.

Why do we need the volume testing?

Before we understand the need for volume testing, we will see one example of volume testing.

Suppose there is an **e-commerce web application**, which is generally used by 1000 end-users. And during the sale or festive season, on the web application, around 40-50k users try to access the web application.

And the application will crash, or the data could not load successfully, which cause problems for the customers.

To overcome such real-time conditions, we need to perform one round of volume testing. The below aspects will help us to understand the importance of Volume testing:

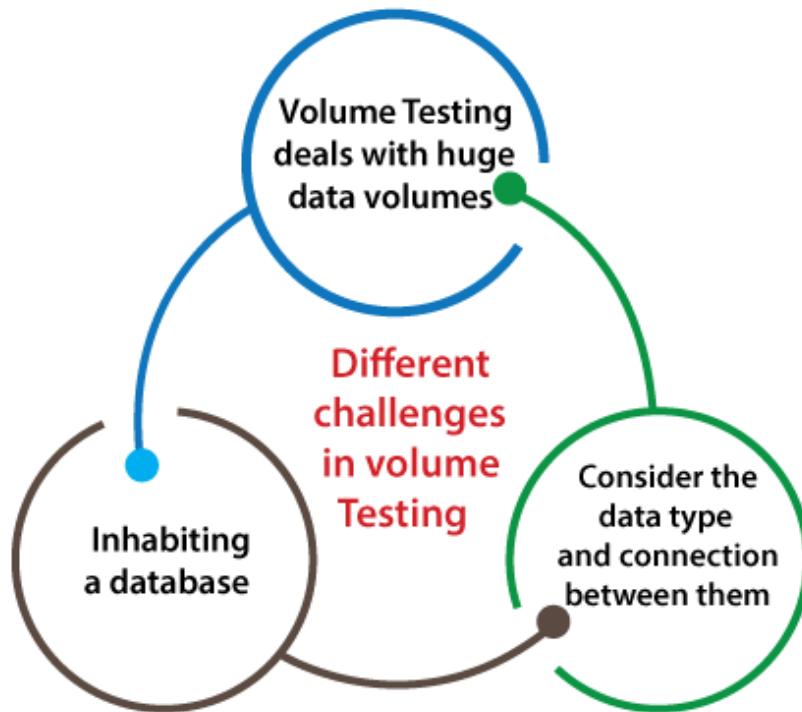
- The volume testing will help us detect any data loss when the volume of the database upsurges to a specific limit.
- The execution of volume testing is essential to identify the problems in the early stages of testing, which could increase the volume of data in the database.

- The volume testing can quickly identify many system performance issues before managing the customer.

What are the different challenges in Volume Testing?

There are few things we always keep remember while executing the volume testing

because all the initial thoughts in volume testing need to deal with data.



- **Volume testing deals with huge data volumes.**

As compared to other categories of **performance testing**, volume testing deals with huge data volumes.

To handle a considerable amount of data it requires extensive data sets with a comprehensive team of test engineers. The programmer also has to deal with managing the data that bumped as an outcome of regular testing meetings.

- **Inhabiting a database**

It is the most commonly faced challenge during the volume testing because we are dealing with the relational databases, which contain a robust structure and lots of adjacent tables.

And the tester needs to gather various fields, both the required elective ones that contain large binary files, to make sure the high quality of test data.

- **Review the data types and connections between them.**

For the testing beginner, it is tough to learn the connections between them, the differences between the types, and how the software reacts.

The test professionals have to deal with an extensive range of data valid, invalid, absent, boundary, or wrong during the volume testing.

Difference between Volume Testing and Load Testing

In the below table, we have listed some of the vital distinction between volume testing and load testing:

Volume Testing

VS

Load Testing

S.NO.	Volume testing	Load testing
1.	It helps us to check the behavior of an application by inserting a massive volume of the load in terms of data is known as volume testing.	Load testing is testing where we check an application's performance by applying some load, which is either less than or equal to the desired load.
2.	Volume testing tests the system both under regular and irregular circumstances.	Load testing tests the system under normal circumstances.
3.	Its analysis the system's response time.	Its analysis the system's performance.
4.	It mainly emphasizes verifying the system's capacity.	The load testing emphasizes on verifying the stability of the software.
5.	It makes the system capable of real-world use.	It makes the system capable according to end-users.
6.	It helps us to save the maintenance cost of the system.	It doesn't save maintenance cost of the system.

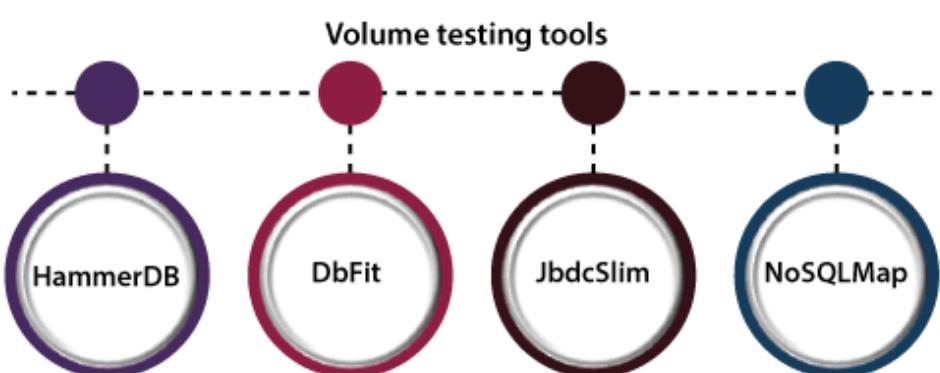
Volume testing tools

As we know that the automation testing is a time-saving process and also provides the precise outcome as compared to manual testing.

The primary advantage of using the volume testing tools is that we can execute the tests at night, and because of that, the team members or the other teams won't be affected by the data volume of the Database.

Let see a few open-source tools compatible for volume testing:

- HammerDB
- DbFit
- JdbcSlim
- NoSQLMap



HammerDb

It is an **open-source** tool and a database benchmarking application to the global database industry. It is used to automate, multi-threaded, and also allows run-time scripting.



Features of HammerDB

Following are the commonly used features of HammerDB:

- It is a fully transparent enterprise rating software with no virtual user restrictions.
- It can be used by all of the top technology companies.
- It is fully supported for various databases such as Oracle, MYSQL, SQL Server, PostgreSQL, etc.
- It allows expert level support.
- It provides complete and comprehensive documentation.
- It is fully functional and supported for **Linux** and **Windows** platforms.

DbFit

The DbFit is an **open-source tool** that supports the test-driven development. The DbFit tests can be used as existing executable documentation of our system behavior.

Features of DbFit

Some of the commonly used features of DbFit are as follows:

- It enables Agile practices such as **Test-Driven Development, refactoring**.
- It supports several database types such as SQL Server, Oracle, etc.
- It helps us to enhance the quality, design, and maintainability of our product.
- It gives readable and understandable syntax, which improves communication with non-technical people.
- It provides online documentation with examples.

JdbcSlim

Another volume testing tool is **JdbcSlim**, where the database statements and queries are easily integrated into **Slim FitNesse** testing. It mainly emphasis keeping the configuration, test data, and SQL commands distinctly.

Features of JdbcSlim

Following are the commonly used features of JdbcSlim:

- The JdbcSlim framework can be used by Developers, Testers, and Business Users who know SQL language.
- It supports all databases for which a JDBC driver
- It is also making sure that the requirements are written independent of the execution and easy to understand by the users.

NoSQLMap

The **NoSQLMap** is an open-source **Python** tool, which is designed to automatically insert outbreaks and disrupt the Database configurations to evaluate the threat.

Advantage of Volume testing

Some benefits of volume testing are as follows:

- It helps save maintenance costs that are spent on application improvement.
- It works only for the MongoDB database.
- It helps in the early detection of bottlenecks.
- It supports a quicker start for scalability plans.
- The volume testing makes sure that the system is capable of real-world usage.

Disadvantages of Volume Testing

Following are the disadvantages of volume testing:

- By using the volume testing, it is not possible to make the precise division of memory used in the real world.
- A skilled database performance testing team is essential to obtain the Volume testing that would become an extra expense of the project.
- The individual copy of the real environment is difficult and complicated.
- It takes a lot of time to execute the thorough volume testing, which covers all the test scenarios, creating scripts, and executing those scripts, which could delay the application's release time.

Overview

In this tutorial, we will learn that volume testing is used to analyze the system performance for high data loads. It is a type of non-functional testing.

The volume testing is executed manually as well as with the help of some automation tools to check the system performance.

The volume testing is very important, which is relatively complicated with its challenges. Before implementing it, we need a complete knowledge of a particular concept along with the database languages.

If any test beginner tries to perform the volume testing, they need to use some tool and perform some test case, which will help us understand the concept of volume testing before using it in real-time.

Scalability Testing

In this section, we are going to understand **scalability testing**, which checks the performance of an application by increasing or decreasing the load in particular scales like number of a user.

And we also learn about **its needs, the purpose of scalability testing, type of scalability testing, prerequisite for scalability testing, various features of scalability testing, how to perform it, advantages and disadvantages**.

Introduction of Scalability Testing

Another type of **performance testing** is **scalability testing**, which comes under the **non-functional testing** of **software testing**.

It is used to check an application's performance by increasing or decreasing the load in particular scales known as **scalability testing**. It is executed at a **hardware, software, or database level**.

It is specified as the capacity of a **network, system, application, product, or process** to make the function correctly when modifications are made in the system's size or volume to meet an increasing need.

In this testing, the **Test Cases** are designed and implemented in a well-organized manner. It also analysis the **system, processes, or database's ability** to meet an upward need.

For example, a **web page** scalability testing depends on the number of users, CPU usage, network usage. In contrast, scalability testing of a **web server depends on the number of requests processed**.

The Objective of Scalability Testing

Following are the critical purposes of scalability testing:

- The main objective of performing the scalability testing is to control how the application balances with an increasing workload and at what point the software product or the system stops scaling and identify the reason behind it.
- The scalability testing is needed to signify the user limit for the software product.
- To find out the client-side degradation, end-user involvement under load, and server-side stability.
- The usages of scalability testing will make sure the ability of software to scale up with the increasing workload and data traffic.

Why do we need to perform the Scalability Testing?

The execution of scalability testing is required because it ensures the even functioning of the software or the application while completing end-user's requirements without any issues.

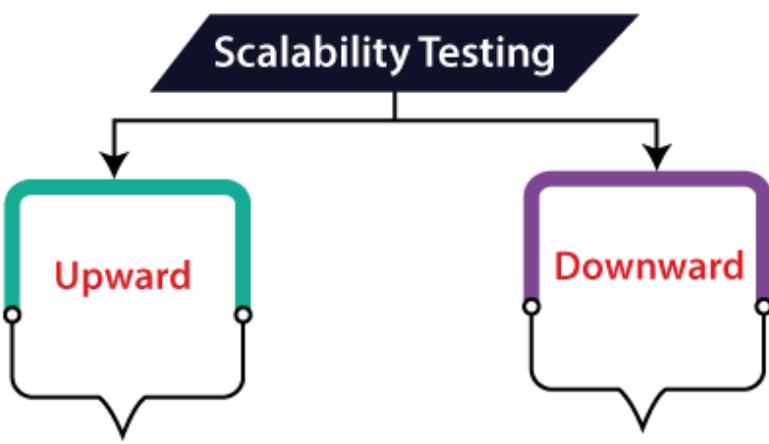
The scalability testing is needed if we encounter the following issues:

- In case of any modification in the software lead us to its failure?
- After the enhancement, the software is working correctly and efficiently in meeting the user requirements and expectations.
- Whether the software can produce and improve as per extended needs?

Types of Scalability testing

Scalability testing is classified into two parts, which are as follows:

- **Upward scalability testing**
- **Downward scalability testing**



Upward scalability testing

The upward scalability testing is used to expand the number of users on a specific scale until we got a crash point. It is mainly used to **identify the maximum capacity of an application**.

Downward scalability testing

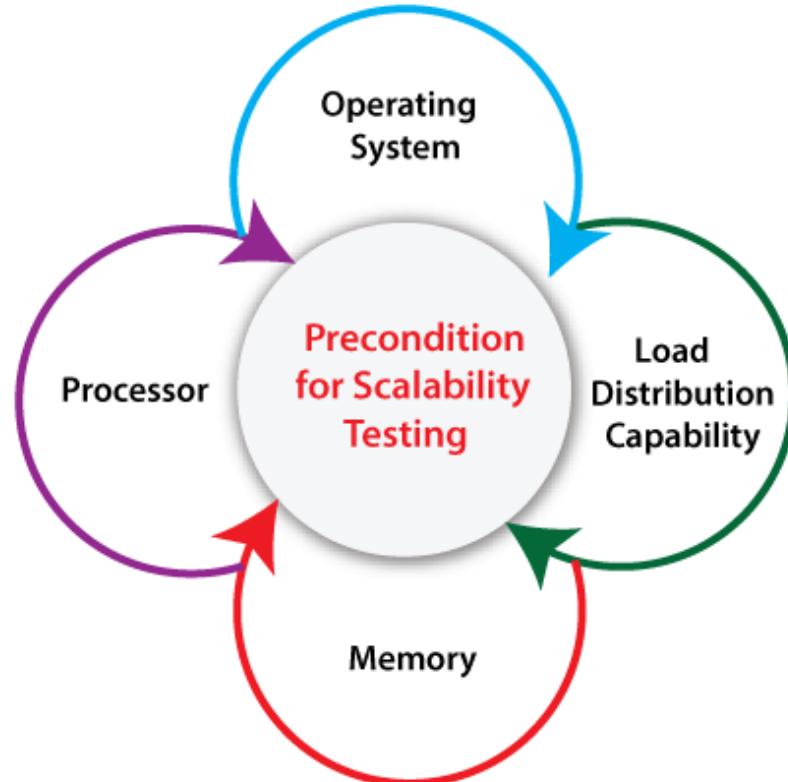
Another type of scalability testing is **downward scalability testing**. When the load testing is not passed, we will use the downward scalability testing and then start **decreasing the number of users in a particular interval** until the goal is achieved.

Therefore, we can quickly identify the bottleneck (bug) by performing downward scalability testing.

The Precondition for Scalability Testing

For the scalability testing, the **test strategy** can be changed based on the type of application being tested.

For instance, if a database is related to an application, then the testing constraints will be the database and the number of users. We have some default precondition available for scalability testing, which is as follows:



- **Operating System**

If we want to perform the scalability testing, we need to verify what operating systems are prepared by the load generation managers and load test master.

- **Load Distribution Capability**

It is used to analysis if the load testing tool allows the load to be created from several devices and measured from an essential point.

- **Memory**

Before performing the scalability testing, we must analyze how much memory would be sufficient for the virtual user and load test master.

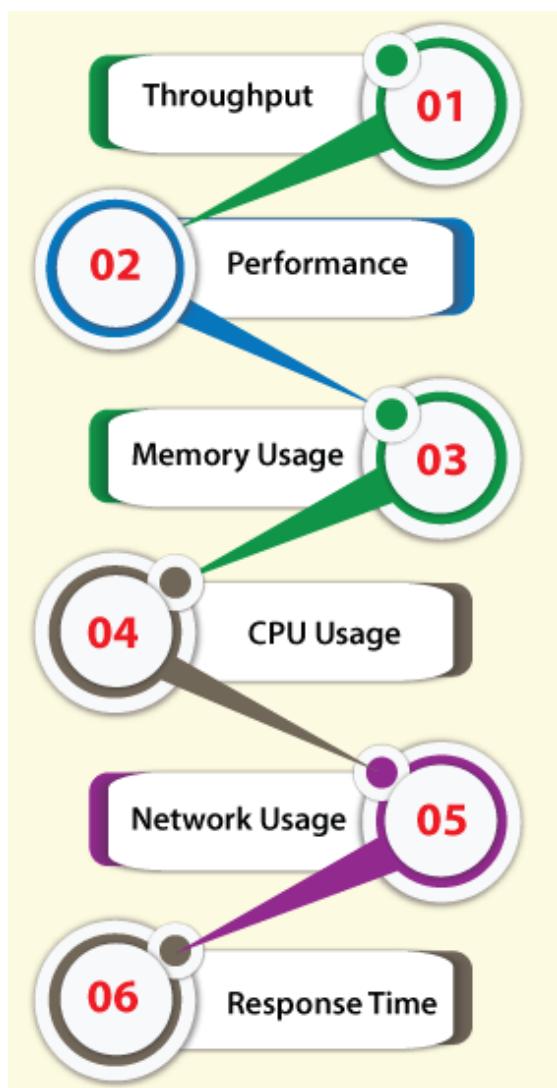
- **Processor**

We need to analyze what type of CPU is required for the load test master and virtual users before executing the scalability testing.

Feature of Scalability testing

Some of the vital components of scalability testing are listed below; let see them one by one:

- **Throughput**
- **Performance measurement with many users**
- **Memory Usage**
- **CPU Usage**
- **Network Usage**
- **Response Time**



Throughput

- The throughput feature is used to specify the amount of work implemented by the application in the given time.
- The throughput can change from one application to another.
- **For example**, in a database application, it is sustained in several commands managed in a unit of time. In contrast, it is uniform in the number of user requests handled in a unit time in a web application.

Performance

- The next feature of scalability testing is **performance**, which is used to check the user's repetitive collective load and request under the **webserver** and **repose of the system**.
- In other words, we can say that the performance of the application depends on its type as it is tested continuously for several users, which can support without its failure or backup condition.

Memory Usage

- In scalability testing, **Memory usage** is one of the **resource utilizations** used to sustain the memory expended for performing a task by an application.
- Typically, the memory usage is calculated in respect of the **unit bytes**.

CPU Usage

- Another **resource utilization** under scalability testing is **CPU usages**, which is used to calculate the CPU utilization while performing application code commands.
- Normally, the CPU usage is calculated in respect of the **unit Megahertz**.

Network Usage

- It is used to carry the bandwidth consumed by an application under test.
- The network usage is calculated in terms of **frames received per second, and bytes received per second, segments received and sent per second, etc.**

Response Time

- It is the time used up between the application's response and the user's request.
- In other words, we can say that the response time checks how fast the system or the application response to user or other application requests.
- It may enhance or reduce the time based on different user loads on the application.
- Usually, the response time of an application reduces as the user load is enhanced.

How to perform Scalability Testing

To perform the scalability testing, we need to follow the below steps:



Step1: Assessment

In the first step of scalability testing, we will assess the existing and predictable future software's possible increasing potential. They also check the criteria for the scalability test and decide the software tools needed to perform the test.

Step2: Test Execution

After the assessment, we will prepare the test plans, test scenarios, and test cases, which cover and analyze the software's working for each incremental development.

Note: While developing scalability tests, it is recommended to enhance the load for following stages to test the system at a fundamental level to an advanced stage.

Additionally, the test environment requests to be persistent and fixed for every diverse enhanced load.

Step3: Test Development

After that, we will set the testing environment, configure the hardware required to implement a scalability test, generate, validate the visual script and the load test scenarios.

Then, perform these test cases in an organized way and analysis the results.

Step4: Logging and reporting

Once the test development has been performed, we will analyze the recorded results and build the necessary report.

Advantages and Disadvantages of Scalability Testing

Below are the benefits and drawback of scalability testing.

Benefits of Scalability Testing

The advantages of Scalability testing are as follows:

- It helps in operative tool utilization tracking.
- The most vital advantage of using the scalability testing is that we can find out the web application restrictions below test in respect of **network usage, Response time, Network Usage, CPU Usage**, and so on.
- If we execute the scalability testing, we can control the end-user experience under the specific load. Hence, the helpful procedures can be taken earlier to fix the issues and make the application more accessible.
- The early detection of issues saves both time and money.
- It enhances the assurance in a software product to see the future development difficulties.
- The scalability testing will help us to save a lot of money and time if we identify the cause of multiple performance issues in the specific application during the testing phase.
- Usually, it includes a set of load tests with different hardware and software settings, and keeping the **test environment** unchanged.

The Drawback of Scalability Testing

Following are the disadvantages of Scalability testing:

- The functional faults can be missed during the Scalability testing.
- Sometimes, the test environment is not always precisely similar to a production environment.

- Occasionally, tests are working fine but fail in the testing phase because of the incorrect test scripts and test scenarios, which waste a lot of time doing pointless modifications.
- If we use the advanced tools for Scalability testing and an exclusive testing team for performance testing, it will cause an over budget for the projects.
- To perform scalability testing, requires a high-level of testing knowledge and skills.

Overview

In this tutorial, we have learned that scalability testing is a part of **non-functional testing** that tests the system's capability, a network, or a process when the size or volume of the system is modified to meet an increasing requirement.

For **Scalability Testing**, the Test Strategy is different regarding the type of application being tested. It is used to evaluate the application to stop scaling and identify the reason behind it.

It works as an operative tool to support the software applications to go along with the users' increasing requirements.

Stability Testing

In this section, we are going to understand **stability testing**, which is an essential part of **Performance testing**.

And we also learn about **its needs, the purpose of stability testing, why we need to perform the stability testing, example, advantage, and disadvantage.**

Introduction of Stability Testing

Stability testing is a **software testing** procedure where we analyze the application's performance by applying the load for a particular duration of time.

For the stability factor, we can say that when N-number of users using the application simultaneously for a particular time.

It comes under the **Non-Functional Testing** directed as part of **performance testing**.

Stability testing will provide the advanced substance in **software dependability, error handling, robustness, and scalability of an application** under massive load instead of analyzing the system behavior under usual conditions.

Generally, it evaluates the stability problems of the application and the productivity of a developed product. And the primary intended of executing the stability testing is to emphasize the software component to the extreme.

Why do we need to perform Stability Testing?

We need to perform the stability testing to achieve the following aspects:

- It helps us to detect the bugs when the system is pushed to strict circumstances and fix those bugs or defects that can increase the software's constancy or the application.
- The life of the software is enhanced if we execute the stability testing.
- If we performed stability tests endlessly, we could identify the constancy of the product.
- If we prepare the system in advance, we can endure any stress/

Purpose of the Stability Testing

Following are the significant objective of stability testing:

- The primary goal of Stability testing is to analyze if the software application fails over regular use at any point of time by exercising its full range of use.
- It helps us find the system's robustness and ensure that the system can handle a vast program.
- The stability testing will allow us to identify the application's constancy, which also enhances the developer's confidence.
- If we perform the stability testing, we can quickly identify the bug in the system in a stressful situation.
- The execution of stability testing will improve the complete evaluation and efficiency of the product.
- It helps us to analyze the database connectivity and test the response time of an application.

How to do Stability Testing?

1. In software testing, the Stability Testing can be executed either manually or with the help of some automation tools.
2. The main reason of performing stability testing is to create stability for the application. A minimum of three batches can be analyzed to assess the strength of the software product.
3. It helps us to define the scope of testing and identify business issues to validate system performance and load data as per user.

4. Any modification in data affects the entire application, and after analyzing the load, if we identify any issues, it can be retested.
5. The primary step in stability testing is regression testing or smoke testing. Once this is passed, we can proceed with functional or non-functional testing.
6. In **functional testing**, the performance of all different features can be completed, and **non-functional testing** will contain the performance-related issues.
7. It governs the **Bug tracking and reporting** and their appropriate mapping with the requirements.

Stability Testing Tools

Each tool has its functionality to test the stability of an application. Still, every organization has specific measures, environmental aspects of an application, and also simulating while selecting any tool to test an application.

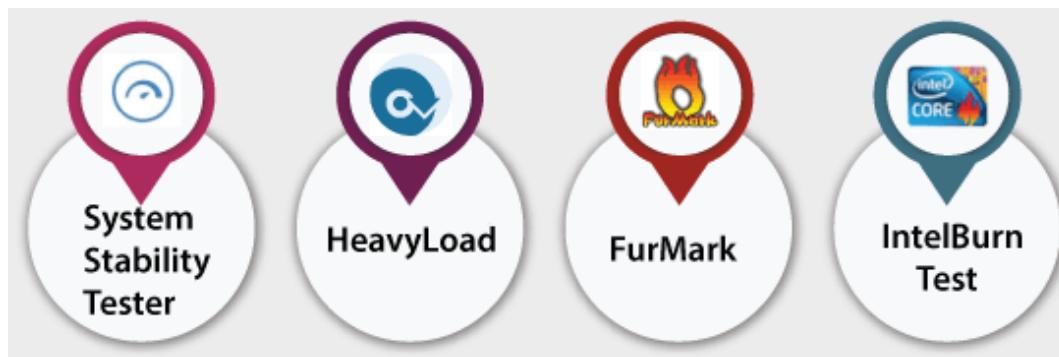
The test engineer needs a test tool that can have the load factor under several twisted scenarios.

The tools used for stability testing rely on the product the user wants to test because there is no such categorization to define which tool is best and worst available in the market for Stability testing.

However, we have several tools available in the market for stability testing, where some of them are **open-source** and some of them are **licensed**.

Following are the most commonly used tools of Stability testing:

- **System Stability Tester**
- **HeavyLoad**
- **FurMark**
- **IntelBurn Test**



System Stability Tester

The system stability tester tool is used for **CPU and RAM stressing, burning, and benchmarking** while consuming all our processor volume.

A specific stress testing process can be launched one by one or concurrently, and it can be enabled and disabled at any time throughout the test.



Feature of System stability Tester

Some of the crucial characteristics of the System stability tester tool are as follows:

- It is a multithreaded processor.
- The System stability tester tool can be performed on multiple platforms.

HeavyLoad

The **HeavyLoad** is used to analyze the stability of the system's primary components like **memory, processor, and hard drive**. It analysis how our system performs when we have decreasing disk space.

It can also execute these tests independently or altogether, which is when maximum stress will place the system hardware and the power supply.



Features of HeavyLoad tool

We list some of the essential features of the HeavyLoad tool, which are as follows:

- It is an open-source tool.
- The HeavyLoad has used tests the allocated memory.
- It does complex calculations to reproduce the load on our processor.
- It is used to replicate the hard drive disk access.
- HeavyLoad tool can work on multiple platforms such as **Windows XP to Windows 8 32-bit and 64-bit**.

FurMark

The **FurMark** is an open-source tool that can be used easily. It is a lightweight but very rigorous graphics card or **GPU stress test** on the **Windows platform**.

It is one of the most commonly used stress test tools for graphics cards because the algorithms used by this tool can be optimized to heat the **Graphics Processing Unit (GPU)**, test its stability, and resistance to extreme circumstances.



Features of HeavyLoad tool

Some of the crucial characteristics of the FurMark tool are as follows:

- It is compatible with various operating systems such as **Windows XP and all the above versions of it**.
- The FurMark tool is available in multiple languages such as **Spanish, English, and German**.

IntelBurn Test

Another open-source and **CPU stress benchmark and application** stability testing tool is **the IntelBurn test**, which helps us to push the CPU to its maximum limit.

The IntelBurn tool is easy to use and a portable device that can check real-time errors.



Features of IntelBurn test tool

The essential features of the **IntelBurn test** tool are as following:

- It is compatible with the modern **Windows-NT-based operating systems and all the above versions of it.**
- It provides real-time output to the screen.
- The IntelBurn tool has a better appearance as compared to the other stability testing tools.
- It simplifies the procedure of Linpack and shortens the process of selecting a test size to use.

Advantages and Disadvantages of Stability Testing

Below are the benefits and drawback of stability testing.

Benefits of Stability Testing

The advantages of Stability testing are as follows:

- It leads to an improved end-user experience and helps in getting a more stable system.
- It also monitors the efficiency of the System.
- It will analysis the stability and durability of the System under massive load.
- Stability testing provides the limit of the data that a system can handle virtually.
- It gives them confidence in the performance of the System.
- Stability testing make sure that even if there are many users, work can be constant without resuming the System.

Disadvantages of Stability Testing

Following are the **drawback** of not executing the stability testing on the system under test:

- If we do not perform the stability testing, the system might fail due to additional load on the CPU or processor that may cause the output's data loss.
- The System fails shortly without implementing the stability testing and slows down the speed with a massive amount of data.
- If we do not perform the System's stability testing, it will have a negative effect on the System's performance as well as the business.
- It works unusually when it goes to the changing environments.

Overview

In this tutorial, we understood that stability testing is an essential part of performance testing under the non-functional testing methodology.

The stability testing is executed to identify the scalability of the System in a specified environment and only concerned with the features of the application.

It helps in achieving performance when the System is under stress.

This testing helps us to analyze the stability of different components like **processors, CPUs, and memory**.

We also learned that stability testing is used to stabilize the system and deliver the good quality product.

Spike Testing

In this section, we are going to understand **Spike testing**, which is an essential part of **Software testing**.

And we also learn about **its needs, the purpose of Spike testing, why we need to perform the spike testing, the process of Spike testing, spike testing tools, and the advantage and disadvantages.**

Introduction of Spike Testing

As we learned earlier, the testing process is the essential part of **SDLC**, which evaluates the multiple software components like **speed, scalability, and dependability**.

It is a subcategory of **stress testing** that guarantees that the developed application works under enhanced and reduced load created by n-numbers. And the system's performance is detected.

It ensures that no issue won't occur in the speed of **software, constancy, and scalability after the product's delivery**.

In other words, we can say that the spike testing is performed to check how essentially the system responds with unpredicted increases and failure of users.

The objective of the Spike Testing

The significant objective of spike testing are as follows:

- The primary purpose of executing the spike testing is to govern whether the system will crash or remain in case of important alteration in load.
- To notice the performance or the behavior of an application under sudden altered load.
- To analysis the recovery time between two points or spikes as it impacts the performance.
- Spike testing is used to analyze the weakness of applications.

Spike testing Process

Spike testing is vital and is mostly done to test how a system reacts when there is an unpredicted change in user load.

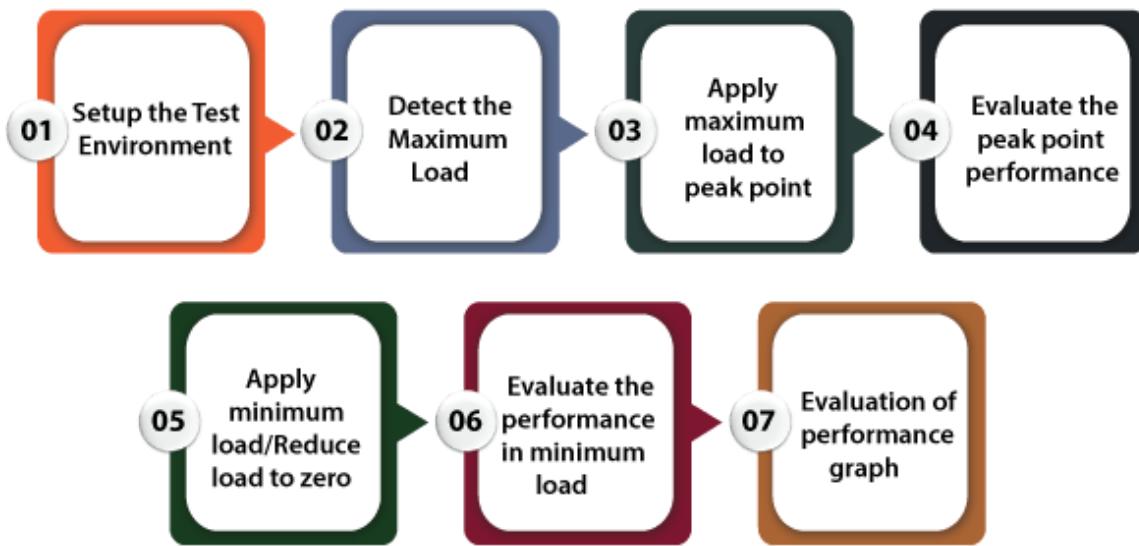
The spike testing procedure involves continuous monitoring. It evaluates the application's vulnerability and checks its affected performance when there are a rapid enhancement and reduction of its load.

Throughout the process of software development, the spike testing is implemented in a precise environment. Therefore, the spike testing process aims to extend an application's maximum capacity to make sure the ideal user involvement.

The spike testing process completed in the below steps, which are as follows:

- Setup the Test Environment
- Detect the Maximum Load
- Apply maximum load to the peak point
- Evaluate the peak point performance
- Apply minimum load/ Reduce the load to zero
- Evaluate the performance in minimum load
- Evaluation of Performance graph

Spike Testing Process



Step1: Set up the Test Environment

The first step of the spike testing process is **setting up the test environment**, which depends on the various parameters in the business needs.

The test environment setup is necessary to perform a successful test and retrieve a fair quality testing process. And we also ensure that no one is using the live environment to execute the **spike testing** in an application.

Step2: Detect the Maximum Load

After finish setting up the environment, the maximum load is initiated; a system can prevent and detect the full load size of the application or the software product, which has developed.

Here, the maximum load is the excessive number of users simultaneously using the system or the application.

Step3: Apply maximum load to the peak point

In the next step, we will enhance the load abruptly for a particular time, and after that, we will try to apply the maximum expected load to the peak point with the help of any **performance tools**.

Step4: Evaluate the peak point performance

Once the maximum load is applied to the peak point, we will evaluate the performance detected under the load on the peak point. It is used to check whether the system fails or survives under this unexpected increase.

Step5: Apply minimum load/ Reduce the load to zero

After that, we will decrease the load slowly to the lowest (zero to minimum load). The process is also executed rapidly, which means that the load is reduced from highest value to lowest possible value.

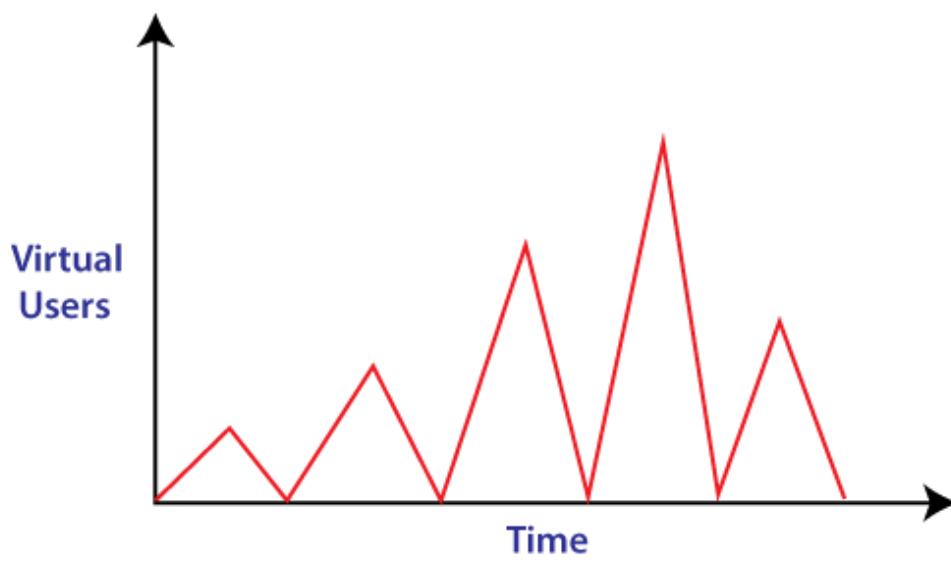
Step6: Evaluate the performance in minimum load

In this step, we will evaluate the performance under the minimum load to see whether the system survives or fails under this unexpected reduce load.

Step7: Evaluation of Performance graph

In the last step, the analysis of performance graph is done by seeing the metrics like **virtual users, failures, and time-taken**, etc.

The evaluation of the performance graph is essential for the test engineer in detecting when the application has undergone crashes, and reporting back to the developers to fix the issue depends on that.



Spike Testing Tools

We have multiple spike tools available in the market, where some of them are open- sources and some of them are licensed tools. And each tool has its functionality, and approaches.

Some of the most commonly used spike tools are as follows:

- **Apache JMeter**
- **LoadRunner**

Apache JMeter

Apache JMeter is the most famous open-source tool in automation testing. It is mainly used to test the performance of both static and dynamic resources and dynamic web applications.

It can reproduce the extensive load on an object, server, and network to discover the complete performance in multiple load types. It helps the developers or the users to use the source code for the development of other applications.



Characteristic of JMeter

Some of the significant features of **JMeter** tools are as follows:

- It is entirely designed on the **JAVA programming language** and platform-independent to load the functional test behavior and measure its performance.
- It keeps the various testing approaches like available, distributed, and load testing.
- It provides a user-friendly GUI, which is interactive and straightforward.
- It is incredibly extensible to load the performance test in multiple types of servers.

For more information about JMeter, refers to the below link: <https://www.javatpoint.com/jmeter-tutorial>.

LoadRunner

Another spike testing tool is **LoadRunner**, which supports spike testing for the extensive range of protocols, number of technologies, and application environments.

It quickly identifies the most common causes of performance issues. And precisely predict the application scalability and capacity.



Characteristic of LoadRunner

Below are the features of the LoadRunner tool:

- We can get detailed performance test reports while executing the LoadRunner tool.
- If we performed the LoadRunner tool, it reduces the cost of distributed load testing.
- The LoadRunner tool supports XML language; that's why we can easily view and handle the XML data within the test scripts.
- It provides the operational tool for deployment tracking.

Benefits and Drawback of Spike Testing

Following are the advantages and disadvantages of spike testing:

Advantage of Spike Testing

Some of the significant advantages of Spike testing are as follows:

- While performing the spike testing, we can easily sustain the system from crashing under the load's spikes.
- The spike testing will offer the test engineer the ability to test the system under an extremely high and low user load.
- It is a beneficial testing process as it saves the software application from failure or crashes.
- In the spike testing, the developers can avoid application failure since the issues are detected.
- The most crucial advantage of spike testings is that it decreases the possibilities of failure for the system or software application.
- Spike Testing is the most suitable testing to evaluate the software's performance even under such stressful circumstances.
- It regulates the software performance and also ensuring a good quality product.
- It is beneficial while we are retrieving the extreme case scenarios.

The Drawback of Spike Testing

Following are the **disadvantage** of not executing the stability testing on the system under test:

- Spike testing required an independent testing environment.
- To perform the spike testing, we need to set up special test conditions, which makes this process bit costlier.
- While performing a spike test, the application's performance may worsen, slow, or stop altogether.
- It is time taking process as compared to the other testing approaches.
- Only the experts can implement the Spike testing.

Overview

In this tutorial, we understood the significance of spike testing in SDLC (Software Development Life Cycle). We have also discussed the spike testing process, benefits, drawbacks and standard defects, etc.

We can say that the Spike testing helps us to find the extreme load and standard issues that can occur once the enhancement in load is applied.

Additionally, spike testing helps in confirming the software system's existing boundaries in the current effective environment.

The correct method to perform spike testing is to suddenly enhance users' numbers, followed by an instant reduction in the load.

Spike testing's primary focus is to detect the unexpected load and prepare the application to work on such conditions. The spike testing is mainly used to maintain the quality and the performance of the application or the software product.

Hence, if we implemented the spike testing in the initial stage of the Software testing process, we can verify its behavior and capabilities.

JMeter and LoadRunner are the most commonly used tools for spike testing.

Negative Testing

Software testing is all about checking the application whether it is working according to the given requirement or not. We may have to use various software testing types like **functional testing**, **Unit testing**, **Integration system**, **System testing**, **smoke testing**, **regression testing**, and **sanity testing** to complete the process.

Software development is not an easy task to complete because it is all about writing extensive and complex codes and then testing these composite codes to guarantee faultless and constant performance. As we know, **software testing** is an essential aspect of writing a successful code.

However, all of these come into the following two categories, such as:

- **Positive Testing**
- **Negative Testing**

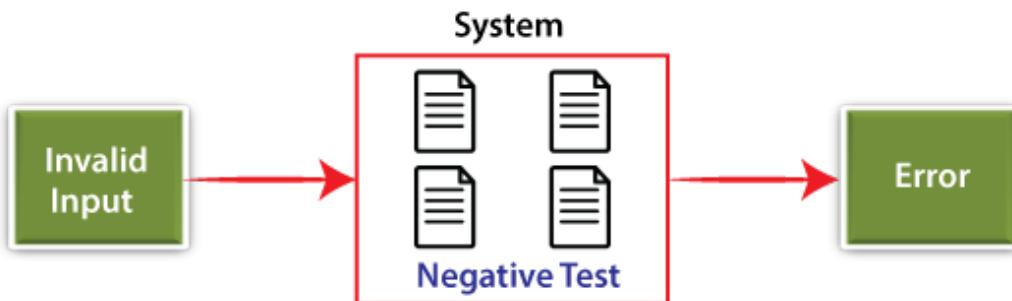
In this section, we are going to cover the following topics related to the particular testing technique known **as negative testing**:

- **Introduction to negative testing and its purpose.**
- **Attributes of negative testing.**
- **Why do we need to perform negative testing?**
- **Examples of negative testing.**
- **Advantages and disadvantages of negative testing.**

What is Negative Testing?

It is a unique type of software testing technique used to evaluate the system for unpredicted circumstances. It plays a very significant role in high-performance software development.

In this testing, the system is authorized by giving the invalid data as input. A negative test analyzed if an application performs as predictable with its negative inputs.



Mainly, negative testing is used to check whether such unpredicted situations will be the software's performance.

In other words, we can say that negative testing is implemented to guarantee that the software product under test does NOT fail when an unpredicted input is given. **It is also known as failure testing or error path testing.**

The Objective of Negative Testing

- The primary objective of performing the Negative testing is to interrupt the system and validate the application response throughout the unexpected inputs.
- The execution of negative testing ensures suitable and ideal software performance even when the user performs inconsistently by inserting the invalid and wrong data.
- To make sure the constancy of the application against the impacts of different variations of inappropriate validation data set, we will implement the negative testing.
- It helps us to identify n-number of bugs and enhance the quality of the software application under test. However, the negative testing is done after the implementation of the positive testing.

Attributes of Negative Testing

Here, we are discussing some of the essential characteristics of negative testing, which are as follows:

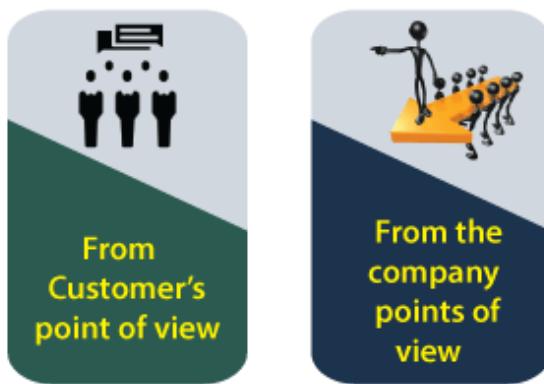
- We can use negative testing to evaluate the potential security breaks and special handling procedures.
- Negative testing is used to analysis the application against the failed conditions.
- It is executed to break the system and achieve failure in a software product's functioning by feeding harmful data.
- It is performed to identify the defects which can result in essential breakdowns.
- Negative testing is executed to display data corruption or security violations.
- Usually, the aim of achieving the negative testing is being performed by a test engineer.
- In order to uncover the software vulnerability and potential for exploitation, we will execute negative testing.
- Negative testing is implemented to guarantee the stability of an application or the software product after being come across with the input values, outside the scope or limit, or invalid input data.
- It is implemented to find the critical loopholes or bugs and weak areas resulting in its failure.

Why do we need to perform the Negative Testing?

Subsequently, performing any type of testing activity is a cost and time-taking process. So, we have to select sensibly whether we need to implement the negative testing in our system or not.

Here, we are discussing why we need to perform negative testing in the particular application by considering the following preservatives of clients, organizations:

Why do we need to perform the Negative Testing?



From Customer's point of view

- The Implementation of negation testing make sure to delivers a bug-free and zeroes vulnerability product in order to meet the Customer's expectation.
- Negative testing is needed when the application is crucial such as e-commerce, online stock, and so on.
- The cost is the only concern to the client while performing the negative testing. But when the effect is evaluated, it is up to the client to choose whether to perform negative testing or not.

From the company's point of view

- To deliver a good quality product to its customer is the responsibility of the organization. And to accomplish this, one has to perform negative testing.
- From a company's point of view, the negative testing needs to be performed because of validation against a failure.
- We should perform the negative testing because sometimes, we cannot guarantee to develop a 100% bug-free system, but we have to ensure that everything is done to avoid a failure.

- By performing the negative testing, we can also cover the vital cases of hacking since there are many hackers out there who are looking for a chance to destroy the system.

Example of Negative Testing

In negative testing, the software product's implementation is evaluated with the help of invalid data inputs. When unexpected parameters are entered, the response of the software is tested.

In such conditions, the application should display the following error message:

Invalid data input.

For example: Suppose we have one sample form to enter the values for the name, phone number, and Pincode fields. In such case, the negative input could be the following:

Name: 838383

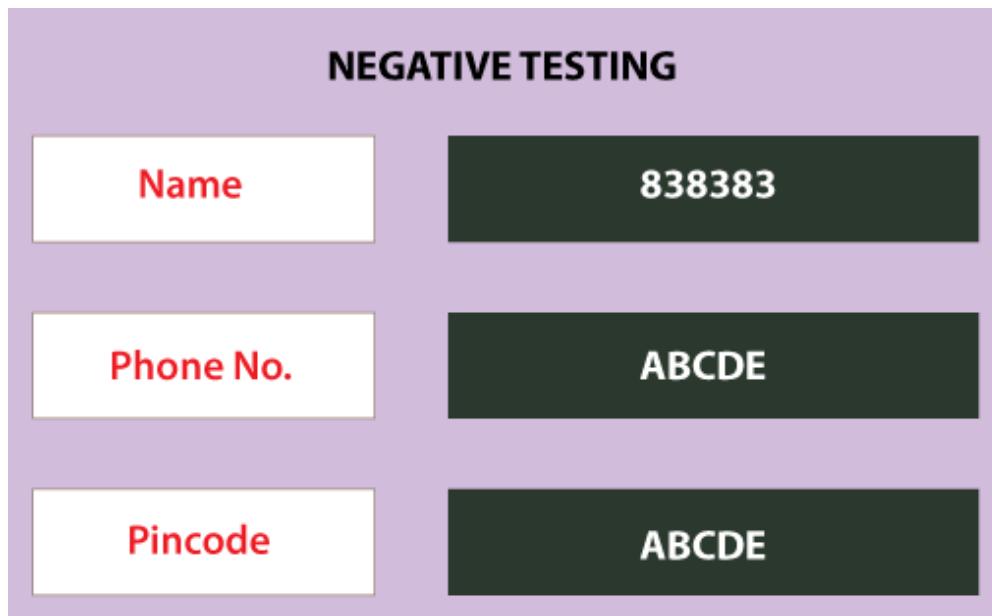
(As the name field only accepts the alphabetic character value)

Phone No: ABCDE

(As the Phone no field only accepts the he values of the number)

Pincode: ABCDE

(As the Pincode field only accepts the he values of the number)



How to perform Negative Testing

To implement the negative testing, the test engineer needs to look out for all the possible scenarios. Specifically, if it is achievable, we have to think about it in the Test Case no matter whether it is not the right way to use it.

For example1:

Suppose we see an image upload option, so we have considered all probable inputs, and we can put there to test it with all possible files.

For example2:

Similarly, we have an email field, and we have to think about all possible inputs, and we can put them there other than the correct email format.

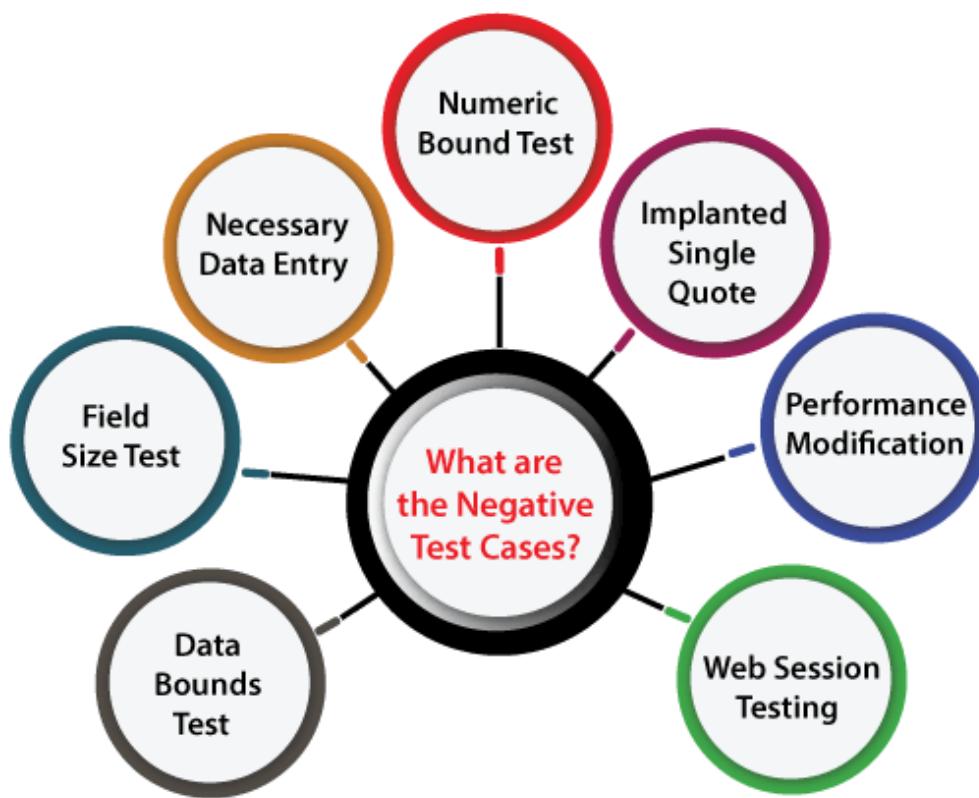
Note: In negative testing, we have to select the cases before implementation to save time and testing cost.

What is a Negative Test Case?

Another significant component of negative testing is **Negative test cases**. The team creates these negative test cases to test the application in a way where it is not meant to be used.

As we understood that the negative testing is performed to ensure the constancy and dependability of the application. And these should be measured by the team in the following testing efforts:

- **Data Bounds Test**
- **Field Size Test**
- **Necessary Data Entry**
- **Numeric Bound Test**
- **Implanted Single Quote**
- **Performance Modification**
- **Web Session Testing**



- **Data Bounds Test**

The testing team must test all the lower and upper bounds for data fields.

- **Field Size Test**

The functional specifications should specify the number of characters one can enter into a field. To ensure that this limit is not exceeded, we can write a test case.

It helps prevent users from incoming more characters before getting error messages after they have surpassed the limit.

- **Necessary Data Entry**

Each field in the software which involves data entry on the screen should be verified before the critical data is entered in the particular field.

- **Numeric Bound Test**

In order to make sure the negative test cases are accurate, the testing team needs to analyze both the lower and upper bounds.

- **Implanted Single Quote**

Some software systems come across an issue when the end-users try to store information containing a single quote.

Hence, for all the screens which receive alphanumeric data entry, the specific team should provide the text that involves one or more single quotes.

- **Performance Modification**

The test suite should contain test cases that compare previous and current release performance statics, which could help in categorizing potential performance problems.

- **Web Session Testing**

As we know that, various web applications vary on browser sessions to display several user information.

And the testing team prepares the **test cases** to release web pages within the application that don't involve users log in.

Negative Test Scenarios

The usage of negative testing enhances the test coverage of a software product or an application and finds the possible application failure in different conditions.

Following are some of the negative cases and test scenarios:

- Web Session Testing
- Inhabiting mandatory Fields
- Allowed number of special characters
- Consistency between Data Field Types
- Sufficient Data

Advantages of Negative Testing

Some of the significant benefits of negative testing are as discussed below:

- It will give more confidence to the client before going live.
- It enhances the possibility of covering all bases and each type of error, which can arise because of inconsistent human behavior.
- As we all understand, the implementation of negative testing is very significant to guarantee a product's quality because a good quality product or an application is considered as a zero-vulnerability product.
- It also ensures that all possible cases are covered during the execution of negative testing because, purposely or accidentally, there is a slight chance of negative test cases. Hence to ensure that all the test cases are covered, we have to perform one round of negative testing accompanied by positive testing.

Disadvantages of Negative Testing

However, the implementation of negative testing is helpful for an enhancement of an application, but there are still some drawbacks of negative testing, which are discussed below:

- The execution of negative testing needs unnecessary time, money, and effort.
- For the customer, it causes extreme delays in the software product or an application release.
- The implementation of negative testing needs a trained and experienced test engineer to develop negative test cases.
- In Software Testing, Negative testing in some cases becomes a time taking process. And in various circumstances, there is no need to execute excessive negative testing.

- Suppose, if the software is only developed for single-person use, then we don't have to think for a situation of 50-100 end-user using the application simultaneously. Therefore, in crucial cases, negative test cases are very significant. There will be times when we don't have to perform negative testing on a specific software product.

Overview

In this tutorial, we have understood the concept of negative testing. And after seeing all the relatable topics of **negative testing**, we can conclude that negative testing makes sure that the delivered software has no bugs and can be restrained in its usage by the customer.

To design the detailed and powerful negative test scenarios requires a creative, skilled, foresight, and intelligent test engineer.

As we know, each software development company desires to have capable and robust software that needs to undertake severe negative testing.

People often live under the delusion that **Negative Testing** is one more way of enhancing the expenditures without any possible benefits. This thought is significant as it can compromise the excellence of the final software product.

In the end, we can say that by implementing Negative Testing, we can enhance the quality of the software and make it stronger.

Positive Testing

Software testing is the procedure of validating and confirming a software application to test whether the application is working as per the given requirement or not.

The primary purpose of performing any **type of software testing** is to identify the bugs or help the end-users to correct his/her mistakes through inserting incorrect data, handling them to categorizes and rectify.

The **software testing** process helps us to enhance the product quality, robustness, and deliver a more user-friendly application.

To achieve all the above aspects, we will use the different testing types and test techniques. Here, we are discussing one of them, namely, **Positive Testing**.

In this article, we are going to discuss the following topics related to the Positive testing:

- **What is Positive testing and its objective**
- **Why do we need to perform Positive testing?**
- **How to perform Positive testing**
- **Example of Positive testing**
- **Benefits of Positive testing**

What is Positive Testing?

It is another essential testing technique used to show a software product or an application under the test.

It validates how the application performs for the positive set of data. In this type of testing, we will enter the valid data set as the input value.



We will implement positive testing to validate the exact working of different software modules with the lines of estimated performance in response to valid data input.

For example, suppose we have the Software applications like website and mobile application which needs objects like **text boxes and text forms**. So, in such cases, positive testing is used to tests the specific functionality of these objects.

Usually, positive testing is implemented to make sure that the particular application or the software product meets the client's specifications and prospects.

In other words, we can state that positive testing is mainly used to help the test engineer to check whether the software is working as expected by using positive inputs or not.

Concisely, positive testing is used to test the software or an application precisely what it's meant to perform.

Note: Positive testing evaluates the positive scenarios like happy paths with valid data.

Why do we need to perform Positive Testing?

Positive testing is required to perform if we may face the following situations:

- Whenever the application is ready for the testing process, a test engineer can implement all the other scenarios designed for that functionality like Database testing, Negative testing, and so on, only after positive testing is approved.
- The execution of positive testing is required each time the build is ready, further known as Smoke testing/**build verification testing**/sanity testing, because it is a first step of the test execution process.

Examples of Positive Testing

For our better understanding of the positive testing, we will see the below examples:

Example1:

In this example, suppose we have one **Textbox** that can only accept **alphabetic character values**.

A rectangular input form with a light orange background. At the top center, the text "Enter Alphabetic Characters" is displayed in bold black font. Below this, there is a single rectangular input field with a thin gray border.

Therefore, in this case, the positive test scenario is as follows:

Positive Test Scenario:

- Enter the **alphabetical character values** and verify that the application receives the input values under test or not.

The same input form as before, but now the text input field contains the text "ABCDE" in red, indicating the user has entered alphabetic characters.

Let's see one more example of positive testing:

Example2

In this example, suppose we have one web application where we need to fill the login form. And, in the specific login page, the following condition needs to be considered:

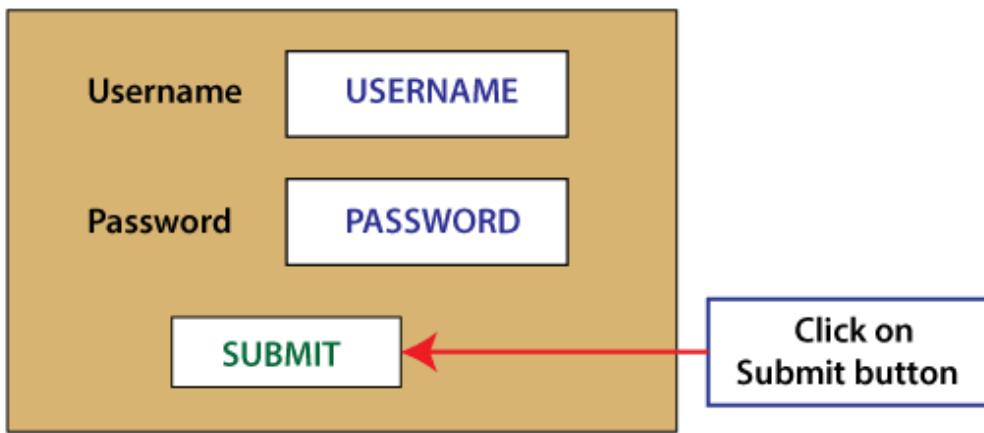
- The user should enter **Username, Password** and click on the **Submit button**.
- After inserting the exact the username and password and click on the Submit button, the user should be navigated to the next page.

A rectangular login form with a light beige background. It contains two text input fields: one labeled "Username" and another labeled "Password", both with thin gray borders. Below these fields is a green rectangular button with the word "SUBMIT" in white capital letters.

So, in the above case, the positive test scenario is as discussed below:

Positive Test Scenario:

- Insert the right **Username and Password** (like **USERNAME** and **PASSWORD**).
- Then click on the **Submit button** and check that the user is navigated to the next page as expected or not.

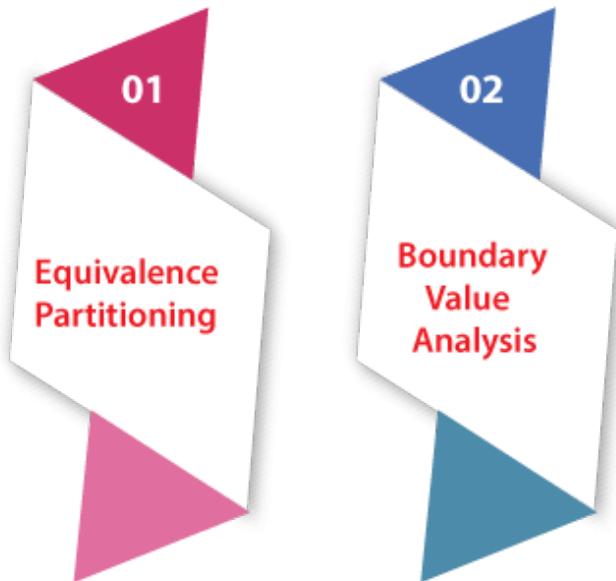


How to execute Positive Testing?

In order to execute the positive testing, we have the following two testing approaches, which are highly recommended and most commonly used during the execution of positive testing.

- **Equivalence Partitioning**
- **Boundary Value Analysis**

How to execute Positive Testing?



Both the testing techniques are mainly used to design a test case. These approaches will be relevant whenever a test engineer needs to test the numeric fields.

A test engineer should check the input data, testing activities, and output while executing the positive testing. Because all the time a test engineer needs to analysis if the test input is within the limits of given test data or not.

Let us see the details working of both **equivalence partitioning** and **boundary value analysis** testing techniques one by one.

Equivalence Partitioning

It is one of the most commonly used test case design techniques of software testing. In equivalence partitioning, the input data is separated into partitions of **valid** and **invalid** values. It is derived from the requirements and specifications of the software product.

In other words, we can say that a test engineer can divide the test input into equal partitions and use the values from each section as test data in the equivalence partitioning approach. A test engineer also needs to ensure that the test data include the values from all partitions.

Let see one sample **example** for our better understanding.

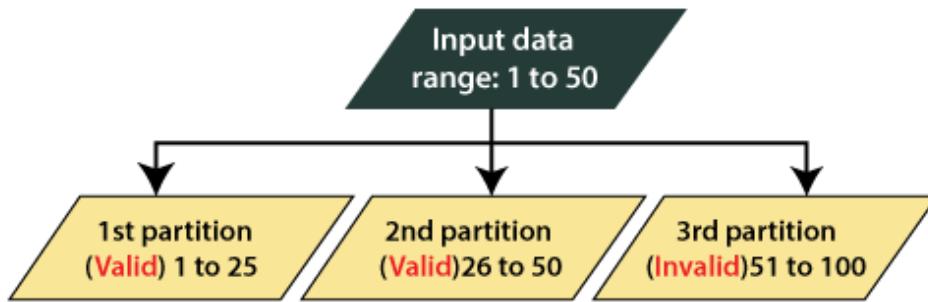
For example, suppose we have one field or system and input value (1-50), which is considered a range of values.

In the Equivalence Partitioning testing technique, the input data is divided into two or more than two partitions, as we can see below:

First partition(valid):1-25

Second partition(valid): 26-50

Third Partition(invalid): 51-100



Boundary Value Analysis

Another most essentially used test case design technique is **Boundary value analysis**. This technique is used to test boundary values as the input values because near the boundary have greater chances of error or mistake. Mainly, it is used to test the numeric fields.

A test engineer needs to develop test data within the boundaries or a data range in the boundary value analysis technique. Because the input data is selected outside the boundary value limits.

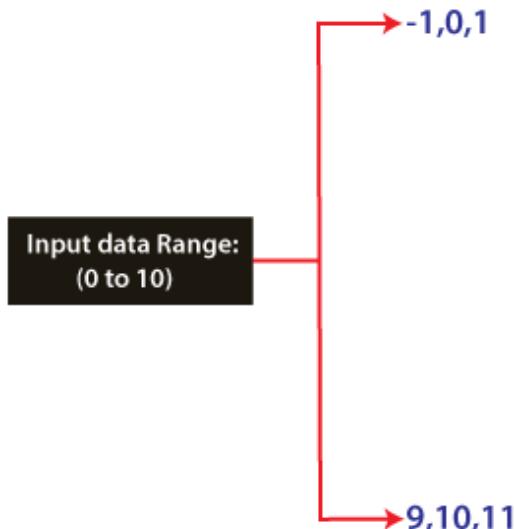
While performing the positive testing, if the input is a **range of values** between A to B, then we have to design the test for such input's values:

- **A, A+1, A-1**
- **B, B+1, B - 1.**

Let us take an example for better clarification:

For example, suppose we have a system that can accept the numbers from **0 to 10** numeric values. And all the rest other numbers are invalid values.

Under boundary value analysis technique, the boundary values will be tested in the below range: **-1,0,1 and 9,10,11**



Advantages of Positive Testing

Some of the significant benefits of Positive testing are as discussed below:

- It is used to saves a test engineer's efforts by finding the wrong build in the initial stages.
- To test the positive path of an application, we can execute the Positive testing.
- Positive testing does not promise the quality of an application as a test engineer cannot test an application's performance in unpredicted conditions, such as when a user enters incorrect data.

Overview

In this article, we have seen in-detail information about Positive testing, like needs of positive testing, execution process of positive testing, examples, and benefits of positive testing.

After seeing all the above mention topics, we can conclude that positive testing is necessary. Positive testing plays an essential role in verifying regular performance of basic objects identified in most websites.

For effective testing results, we need to perform both **Positive and Negative** testing, which provides enough certainty in the software's quality.

The primary aim behind implementing all type's software testing is to ensure that the software product is bug-free before the software is launched and help deliver a high-quality software product.

Endurance Testing

In this section, we are going to explore the following topics related to the particular testing technique known as **endurance testing** in detail.

- **What is Endurance Testing?**
- **The Objective of Endurance Testing**
- **Attributes of Endurance Testing**
- **Process of Endurance Testing**
- **Endurance testing examples**
- **Advantages and Disadvantages of Endurance Testing.**

What is Endurance Testing?

In Software testing, we have several different types of testing that are very effective while analysis any software or application.

Here, we are discussing another type of software testing, which is known as **Endurance testing**.

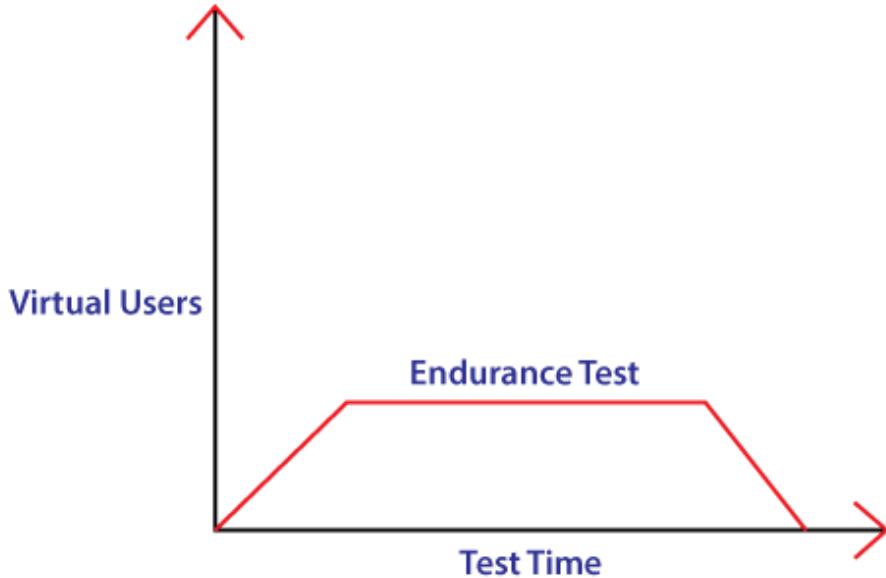
Endurance testing is testing where we test the system performance under certain load conditions over an extensive period.

Endurance testing is one of the significant parts of non-functional testing.

Generally, it is also known as **Soak testing** and **longevity testing**. Endurance testing is done at the last phase of the performance run cycle.

Note: Soak testing is used to test the behavior of an application in the environment, which is unsupportive for a long time. Usually, soak testing is a destructive type of testing; subsequently, we already know that the server or domain is not supportive.

In order to evaluate the response of a tested component under possible replicated situations for a specific load and time, we used to implement the endurance testing.



In simple words, we can say that the term **Endurance** is used to define **the capacity of something to last**, which is further call it as **durability, ability**, or **Endurance**.

The memory consumption is measured to fix potential failures and determine the performance quality throughout Endurance Testing.

For enhancing the consistent parameters of the software application, we can record the performances in the endurance testing.

To find out any memory leaks in the system, endurance testing is beneficial, and it also helps us test the response time of the system over a more extended period.

During the endurance testing, the testing is performed for a more extended period as per the given testing requirements, such as 15 hours, 90 hours, as compared to other type of testing such as **load or stress** testing.

And that makes endurance testing vary from **Load Testing**, which generally ends in a couple of hours.

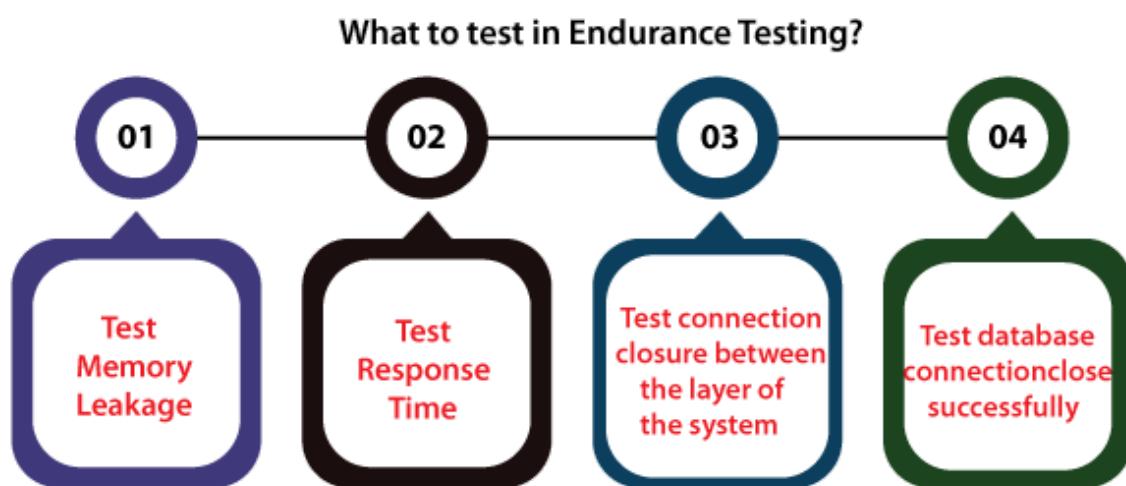
The Objective of Endurance testing

- The primary objective of performing the endurance testing is to confirm that the application is efficient enough to handle the increased load without response time failure.
- The purpose of executing the endurance testing is to test the memory leaks.
- It is generally performed by either overloading the system or decreasing the specific system resources, and assessing the results.
- The implementation of endurance testing ensures that the system response time will continue the same or enhanced after an extended period.
- For identifying that how the system performs under constant usage.
- To accomplish the future loads, we need to understand how many additional resources like disk capacity, processor capacity, memory usage, or network bandwidth are essential.
- It governs the robustness and stability of the software product for the maximum time limit.
- It focuses on the consumption of memory.
- Before release any product in the market, we have to make sure that it will help in growing confidence in the software or system stability.

What to test in Endurance Testing?

During the execution of endurance testing, we will test the following aspects:

- **Test Memory Leakage**
- **Test Response Time**
- **Test connection closure between the layer of the system**
- **Test database connection close successfully**



Test Memory Leakage

In the implementation of endurance testing, checks are done to verify any memory leakage in the particular software or an application, which can cause failure of the system or operating system.

Test Response Time

We can also examine the test response time. And the application or system is tested for the response time of the system as the application becomes less operative as an output of the system's using endlessly.

Test connection closure between the layer of the system

During the endurance testing implementation, if the connection between the system's layers is not closed effectively, it may stall some or each modules of the system.

Test database connection close successfully

If the database connection is not closed successfully during the execution of endurance testing, we may get output in a manner of system crash.

Example of Endurance testing

Let us see an example of Endurance testing for our better understanding of the particular topic.

Here, we will see the example of the memory leak concept, but we will first understand what memory leak is?

Memory Leak: It is the most commonly faced issue in computer science. It is a breakdown to release objects that are no longer used because of a vulnerable programming code written by a developer.

In endurance testing, the test engineer will execute the system for a fixed amount of time by providing a specific quantity of load to test any memory leak.

Handling memory leak issues are completely varied in the usage of programming language.

Hence, we will see one real-life example of any bank.

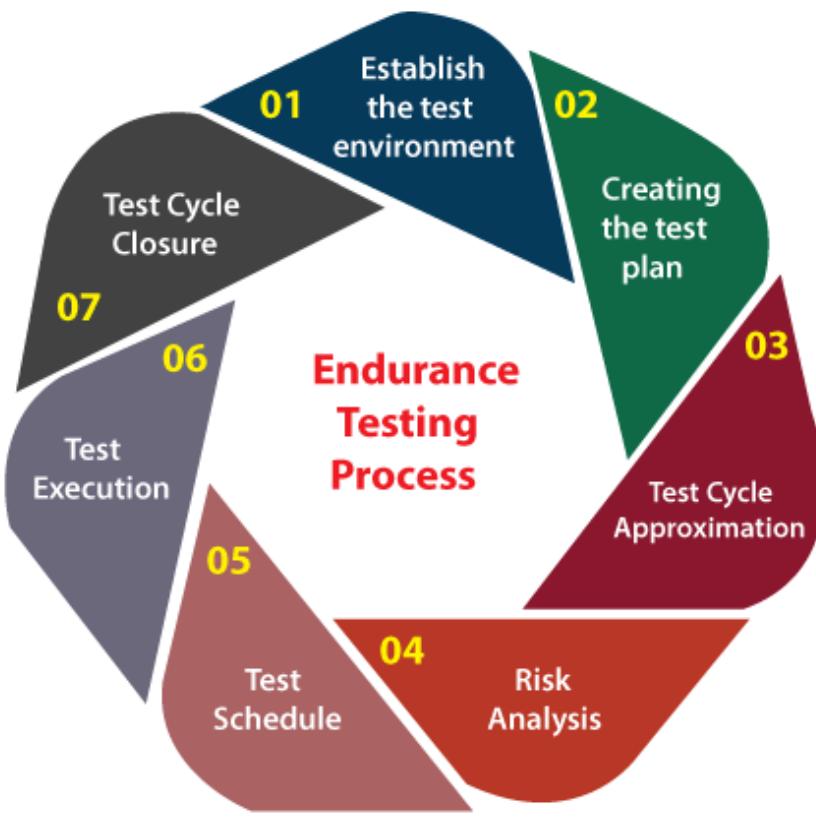
Example:

- Suppose lockdown has happened and banks will not work for straight 7-8 days.
- Throughout the lockdown period, online banking activities will upsurge as compared to other standard business days.
- A system needs to be ready to handle such a load if there is upsurge in the number of users and the number of operations.

Endurance Testing Process

The process of endurance testing is completed into the following steps:

- **Establish the test environment**
- **Creating the test plan**
- **Test Cycle Approximation**
- **Risk Analysis**
- **Test Schedule**
- **Test Execution**
- **Test cycle Closure**



Let's discuss them one by one for our better understanding:

Step1: Establish the test environment

The first step of the endurance testing process is **creating or establish the test environment**. This step will find the necessity of software, hardware, operating system, and database to execute the endurance test.

The test environment should be prepared before the test execution, and it should be separated appropriately from the actual live system.

The creation of team is included to execute the endurance testing and handover the roles and responsibilities within the group.

Step2: Creation of Test Plan and Test Scenarios

After completing the first step of the endurance testing process, we will move to the next step, creating a test plan and test scenarios.

In this, the test cases should be designed, reviewed, and execution should be prepared based on the necessity of manual and automation testing or a combination of both.

The application's breakpoint should be decided, and it should be confirmed how much load would be put on the application throughout an Endurance test.

Step3: Test Cycle Approximation

After the completing both step1 and step2, we will move to the next step of endurance testing, **Test Cycle Approximation**.

The time of every test phase and how many test cycles are required are evaluated in the test cycle approximation step.

It provides an approximation of how much time is required to complete the testing phase. It should be evaluated based on the number of test engineer included and the necessity of number of test cycles.

Step4: Risk Analysis

The risk analysis is the most critical step in the endurance testing process. Here, we will prioritize the test cases based on the risk factor.

And below are some of the most commonly faced issues and risks by the test engineers during the endurance test.

- Is there any external interfering which is not yet delivered?
- Are there any other minor issues, which are not yet identified?
- Is the performance test will to continue reliable with time?

Step5: Test Schedule

In the next step of the endurance process, we will conclude the budget, deliverables, and timeframes for the further process.

Step6: Test Execution

Once the test schedule has been set, we will move to the next step that is **test execution**. In this, test cases are implemented as soon as the environment is ready, and if there are any issues recognized throughout this stage, they are also noted down.

And the developers will solved these issues and retesting is performed after they are fixed.

Step7: Test Cycle Closure

The last step of endurance testing is **test cycle closure**. As soon as the test cycle meets the testing process's exit criteria, it will be considered closed, and a report of recognized issues and their final status will be arranged.

Advantages of Endurance Testing

Some of the significant benefits of endurance testing are as discussed below:

- It helps in identifying bugs related to a memory leak.
- The system might face performance-related problems once they are executed for a more extended period.
- Endurance testing also helps us discover these issues and checks the software's robustness.
- During the endurance testing, the bugs and faults can be easily identified, which allows the development team and the client to enhance the infrastructure.
- The endurance testing assures the development team and the client that the system will implement efficiently for a more extended period without any maintenance, specifically on long weekends.
- The essential benefit of performing the endurance testing is reducing maintenance costs that can be higher if particular performance-related issues are not identified throughout the testing cycle.
- Like another non-functional testing type, such as performance testing, endurance testing also results in customer satisfaction.

Drawbacks of Endurance Testing

Though the execution of endurance testing is helpful for an enhancement of an application, there are still some drawbacks of endurance testing, which are discussed below:

- It is a bit costly as it needs the system to run endlessly during the test execution, where we will also involve the suitable infrastructure.
- Endurance testing is a time-consuming process.
- In order to execute the endurance testing, it requires appropriate planning, especially when it comes to determining the number of hours the test runs need to be completed.
- It is essential to keep the endurance testing environment distinct from other test environments because it might cause application failures or data loss.
- It is not possible to execute the Manual endurance testing; therefore, we will use the automation tool to run the testing.
- And the testing team should be experts in order to use the various automation tools.

Overview

In the above article on endurance testing, we have discussed the in-details process of endurance testing, an example of endurance testing, some essential benefits and drawbacks of performing the endurance testing.

Finally, we can say that **Endurance Testing** is non-functional testing and a subcategory of **load testing** using performance testing.

Endurance testing is performed to check the performance of the system under constant use. In terms of detecting the issues such as memory leaks, the execution of endurance testing is essential. These issues can be the reason for system failure, causing the loss of crucial data.

Endurance testing is making the application more robust and prepares it to sustain under continuous heavy loads. We should prevent performing endurance tests manually, as it is a very time-consuming process. That's why it is mainly automated.

Some of the most frequent tools used for Endurance test are **Apache JMeter**, **LoadUI**, **LoadRunner**, **WebLoad**, **IBM Rational Performance Tester**, etc.

The time of the Endurance test depends upon the business, project, and client requirements. It may last for 10-15 hours or few days or a month or sometimes even a year.

The endurance test process begins with establishing an isolated test environment, creating test plans, estimating test cycle duration, analyzing the risk, preparing the test schedule, executing the endurance test, and finally closing the test cycle.

Monkey Testing

In **Software Development Life Cycle (SDLC)**, the testing phase plays a crucial role in order to help test engineers to establish the quality, performance, consistency, efficiency, and security of the product, together before and after its release.

As we know that the testing process is the best way to identify the **bugs and defects** in the particular software and control them immediately, they are detected by the team of test engineers.

In this tutorial, we are going to understand the following topic of the particular type of software testing, namely **Monkey testing**.

- **Introduction to Monkey Testing**
- **The attribute of Monkey Testing**
- **Usage of Monkey Testing**
- **Types of Monkey Testing**
- **Smart Monkey Testing vs Dumb Monkey Testing**
- **Monkey testing vs Adhoc testing**
- **Advantages of monkey testing**
- **Disadvantages of monkey testing**

Introduction to Monkey Testing

One of the unique types of **software testing** is **Monkey Testing**. It was firstly introduced in the book **The Art of Software Testing** which is written by **Glenford J. Myers** in **1979**.

It is a software testing technique where the user checks the application by giving random inputs; that's why it is also known as **Random testing**.

If we don't have enough time to write and perform the tests, we will implement the **monkey testing**.

It is also known as **stochastic testing**, and best suited for desktop, web, as well as mobile applications. It is a time and effort-saving process if we are using random testing or monkey testing inputs.

Monkey testing is usually executed as random, automated unit tests, and provides us the benefits of efficiently assessing software reliability from test results.

The monkey testing is primarily implemented automatically where the user inserts any random invalid input and tests its performance.

If we are performing load and stress testing, **Monkey testing** works very well, or we can say it is a very good approach.

The data created or developed from random testing can be additionally used to approximate product consistency. On the other hand, various testing approaches cannot be used in this way to approximate the software consistency.

It is trendy among test engineers as they used this testing to test applications by providing random inputs and checking their behavior.

To detect the bugs and errors in the software application by using tentative performances is the primary intent of executing the monkey testing.

Features of Monkey Testing

Subsequently, **Monkey Testing** includes testing the software or application by giving some random data and detecting whether the system fails or not.

The key intent of executing the monkey testing is to identify the defects and errors in the software and make sure that the system does not crash once the entire development on the software product is done.

Some of the significant features of monkey testing or random testing are as follows:

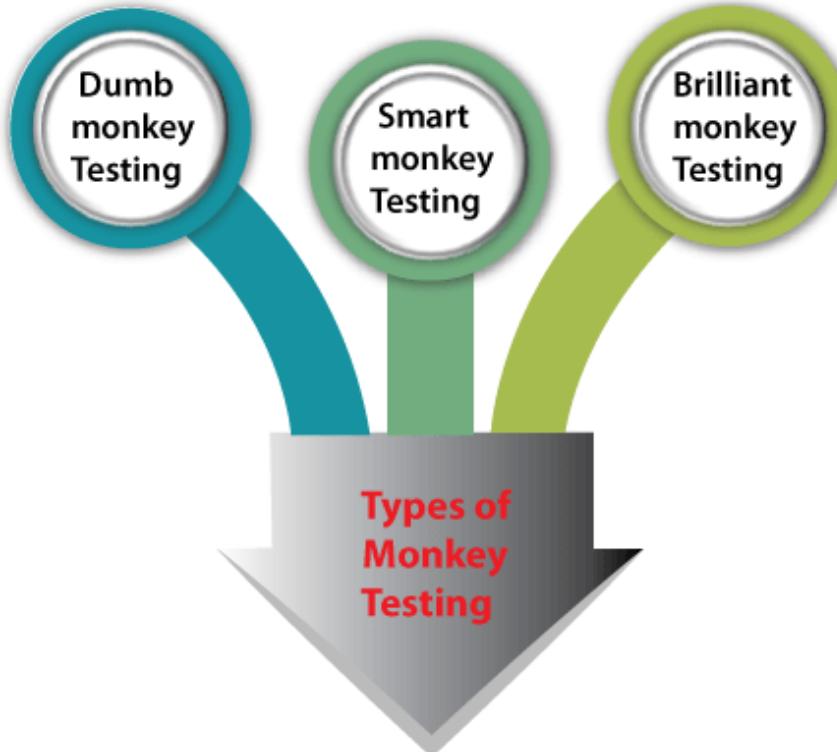
- While performing the monkey testing, there is no specification.
- It is an instinctive test that is performed without any precise test in mind.
- Monkey testing is implemented when the defects are not detected in regular intervals.
- The monkey testing is used to list the system's dependability and performance.
- The time and energy devoted in monkey testing are comparatively less than actual test efforts.
- The implementation of monkey testing makes sure the reliability and efficiency of the system.

Usage of Monkey Testing

- Monkey testing can be used for database testing by beginning a transaction and inserting some random data.
- It can also be tested with the help of hardware or, more preferably, software to imitate the activities of a monkeys who are inserting the random data.
- To test the application for OWASP issues, we can use the pre-compiled and random data.

Types of Monkey Testing

According to its way of execution, **Monkey Testing** is divided into various types. To get a quick idea of it we can see in the below images:



- **Dumb Monkey Testing**
- **Smart Monkey Testing**
- **Brilliant Monkey Testing**

Now, let us see them one by one for our better understanding:

Dumb Monkey Testing

The first type of monkey testing is **Dumb Monkey testing**. It is the easiest and direct type of monkey testing that does not know their or system's abilities or the application's flow.

In this testing, we can identify fewer bugs than the smart monkeys but can naively discover significant bugs that are hard to find.

In this, the **test Manager or lead** appoints a test engineer who does not have the knowledge of particular module of an application in order test the product.

Even those test engineers do not have any idea whether the given inputs are valid or invalid. That is why dumb monkey testing is also known as **Ignorant monkeys**.

In other words, we can say that the dumb monkey does not aware of the work flow or data being sustained to the application or the system.

The test engineer is asked to test the application automatically and enter random data where the test engineer feels suitable.

The following features can acknowledge a dumb monkey:

- The test engineer's behaviour may be like a user who does not have a technical knowledge but is trying to use the application.
- The test engineer can test the application according to their understanding and enter data that is invalid.
- When the test engineer does not have any idea about the application.
- To see if it performs as expected we can note down the application's behaviour.
- The tester doesn't know that the inputs whether its valid or invalid.
- When the tester doesn't have much knowledge about the user interface and functionality.

Smart Monkey Testing

The next type of monkey testing is **Smart Monkey testing**, wherein the **test engineer is entirely attentive of the system or the application**.

Additionally, the testing team knows the functionality of the exact product and consequently gives inputs, which are suitable and valid that helps to execute tests on the product.

Furthermore, the smart monkey tests are aware of where the pages of the application will redirect to.

In smart monkey testing, the test engineer is focused on surpassing the application, and if they identify an error, they are smart enough to report a bug. Also, smart testing is an excellent choice to perform **load testing and stress testing**.

In other words, we can say that the test engineer has a piece of specific knowledge about the system, its objective, and functionality, and the test engineer navigates through the system and provides valid inputs in order to execute the testing.

Brilliant Monkey Testing

The last and third type of monkey testing is brilliant monkey testing. In this type of monkey testing, the test engineer has a good knowledge of the system.

This type of testing can also identify some bugs, which might be found in the software in the future.

Furthermore, the test engineer knows about the pattern of using the product, and henceforth, they can perform testing from the user's viewpoint.

In other words, we can say that brilliant monkey testing is one step ahead of smart monkey testing.

In this, the test engineer or a team of test engineers is assigned for the task, as they have complete knowledge about the particular software domain and its features.

Difference between Smart Monkey Testing and Dumb Monkey Testing

Let us see some significant comparisons between **Smart Monkey Testing** and **Dumb Monkey Testing** in the below table.



S.NO.	Smart Monkey Testing	Dumb Monkey Testing
1.	In this, the smart monkeys are aware of the system's workflow, which means where the product is and where it will be managed.	In this, the dumb monkeys are not aware of the workflow of the system.
2.	In smart monkey testing, they have some working ideas about the software product.	In dumb monkeys, they do not have any knowledge related to the system and its feature.
3.	In this, the smart monkeys can report a bug on detected errors or defects.	Compared to the smart monkey testing, it finds lesser bugs but may discover bugs that are difficult to detect, even by the smart monkeys.
4.	In smart monkey testing, the tester can understand the competencies of the system together with its strength.	In dumb monkey testing, the tester is not able to establish the system's abilities.
5.	In smart monkey testing, the smart monkeys are well aware of inputs sustained to the system. That means they understand the valid and invalid types of input.	In dumb monkeys, the dumb monkeys are not aware of the input type and provide a valid or invalid system.

Is Monkey Testing being similar as Gorilla Testing and Fuzz Testing?

- As we know, the **gorilla testing** is pre-planned whereas the monkey testing is very much Adhoc in nature, and as compared to gorilla testing, we must be muddled many times.
- If we compared fuzz testing with both gorilla testing and monkey testing are very much different to each other.
- The concern about fuzz testing with the randomly selected data input, while monkey testing deals with the random actions to execute the testing.
- Hence, we can conclude that **Monkey testing** is different from other types of testing procedures, which serve different purposes. We can select the appropriate type of testing as per our requirement, whether it is **monkey testing, gorilla testing, and fuzz testing**.

Note: Some people find monkey and Adhoc testing as a similar type of testing technique, which is incorrect because ad-hoc testing is performed without any planning or documentation work.

To clarify the above statement, we can see the difference between monkey and Adhoc testing.

Monkey Testing vs Adhoc Testing

Let's see some significant differences between monkey testing and **Adhoc testing** in the below table:



S.NO.	Monkey Testing	Adhoc Testing
1.	It is performed randomly with no explicitly predefined test cases.	It is performed against the client's requirements.
2.	It can be implemented by anybody, even individuals who are not aware with computers or the application.	It can be executed by a developer as well as the test engineer who has a good knowledge of the application.
3.	The primary objective of implementing the monkey Testing is to execute the test randomly with random or invalid data to check if the application is failed or not.	Adhoc testing aims to check whether the system crashes the application or finds a defect by randomly using the application.
4.	In monkey testing, we can use the test cases since it is random.	Adhoc testing is also performed randomly but does not depend on or use Test Cases.
5.	In monkey testing, the test engineer may not know what the system is all about, and its objective.	In Adhoc testing, the test engineer must understand the system extensively before executing the testing process.

Benefits and Drawbacks of Monkey Testing

It is a software testing procedure that analyzes the test scenarios, which are random and Adhoc. Despite giving various benefits to the end-users and test engineers, monkey testing has numerous drawbacks, which can suppress its positive abilities.

Hence, there are some of the essential advantages along with the disadvantages of monkey testing are as discussed below:

Advantages of Monkey Testing

Following are some of the significant benefits of monkey testing:

- There is no need for a skilled testing engineer in order to perform the monkey testing.
- Monkey testing is very cost-effective.
- Individuals can quickly classify bugs with the help of this testing, which may significantly impact the software's efficiency and performance.
- To test the reliability of the software, it is an excellent approach.
- The monkey testing technique is easy to set up and implement.

- New bugs can be identified during the monkey tests; they would not have been discovered throughout the traditional testing.
- It is the best approach in order to execute the stress testing and load testing in an Adhoc manner.

Disadvantages of Monkey Testing

Following are the drawbacks of Monkey testing:

- Evaluate the unexpected issues identified during the monkey testing makes this process very difficult and time-consuming.
- In monkey testing, the identified bugs can be out of scope or out of business needs.
- The test engineer cannot assure the accuracy of test cases as they have difficulty to define the exact test scenarios.
- The implementation of monkey testing may consume lots of time before classify a bug as it does not have any predefined tests.
- It is done randomly; that's why test case coverage can't be specific.

Overview

In this tutorial, we have learned that **Monkey Testing** is a type of software testing that is relatively new and popular among software test engineers and very beneficial in some testing areas.

Monkey testing is knowingly different from **Gorilla testing** and **Adhoc Testing**.

Monkey testing is divided into three types of testing is **Dumb monkey testing**, **Smart monkey testing**, and **Brilliant Monkey testing**.

In Monkey testing, the users entering the random inputs and then implementing the testing to check its performance and to know whether the system or application fails or not.

Furthermore, monkey testing is an automated test conducted by a team of test engineers without any detailed test in mind.

And at last, we can say that Monkey testing is the easiest way of checking the quality of software features and the performance of the software; because of this quality, monkey testing is also known as random testing.

Agile Testing

The **Software Testing** process contains several testing procedures and techniques that simplify the testing process and increase its efficiency.

In this tutorial, we are going to understand the following topic related to specified testing known as **Agile Testing**, which helps us to enhance our knowledge of Agile Testing:

- **Introduction to Agile Testing**
- **Principle of Agile Testing**
- **How is Agile Methodology used in Testing?**
- **Agile Testing Strategies**
- **Agile Testing Quadrants**
- **Agile Testing Life Cycle**
- **What are the different challenges we faced during the agile Testing?**
- **Advantages of Agile Testing**
- **Disadvantages of Agile Testing**

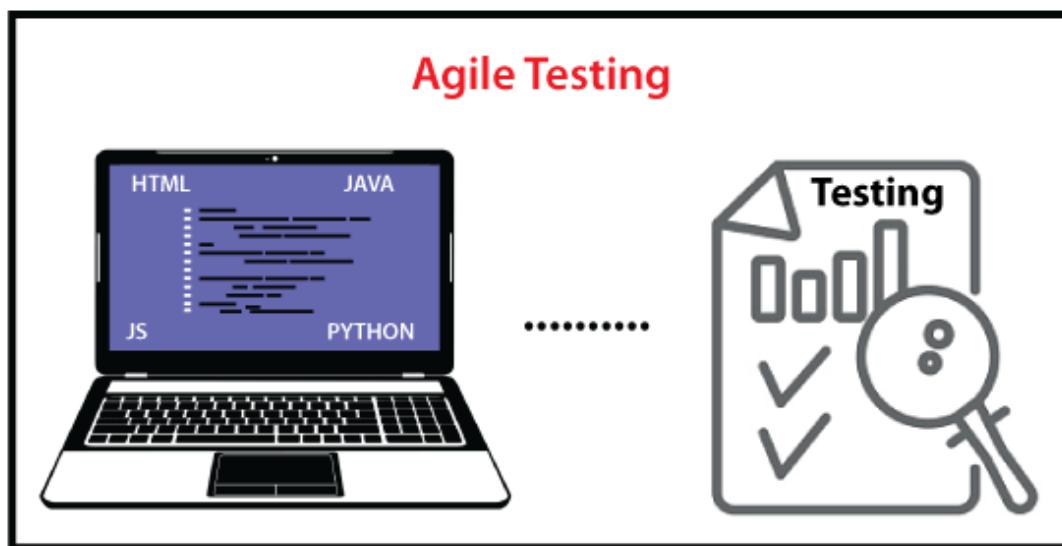
Introduction to Agile Testing

Here, we are discussing another essential **software testing** technique, which is known as **Agile Testing**. Agile testing follows the standards of agile software development.

Agile testing is an **iterative and incremental method**, and the necessities, which develop during the cooperation between the **customer** and **self-establish teams**.

In agile testing, the word "**Agile**" primarily signifies something that can be performed quickly and immediately, but also in the area of **software development**.

The core-functional agile team implements it in order to test the software product and its several modules. The implementation of agile testing makes sure to deliver a high quality product as bugs or defects get deleted in the initial stage of the project itself.



Unlike the **Waterfall model**, Agile Testing can create at the beginning of the project with endless incorporation between development and testing. It is not a sequential but the continuous process.

The agile testing process is a smart way of testing complicated software, which accepts more effective results as compared to the traditional testing procedures.

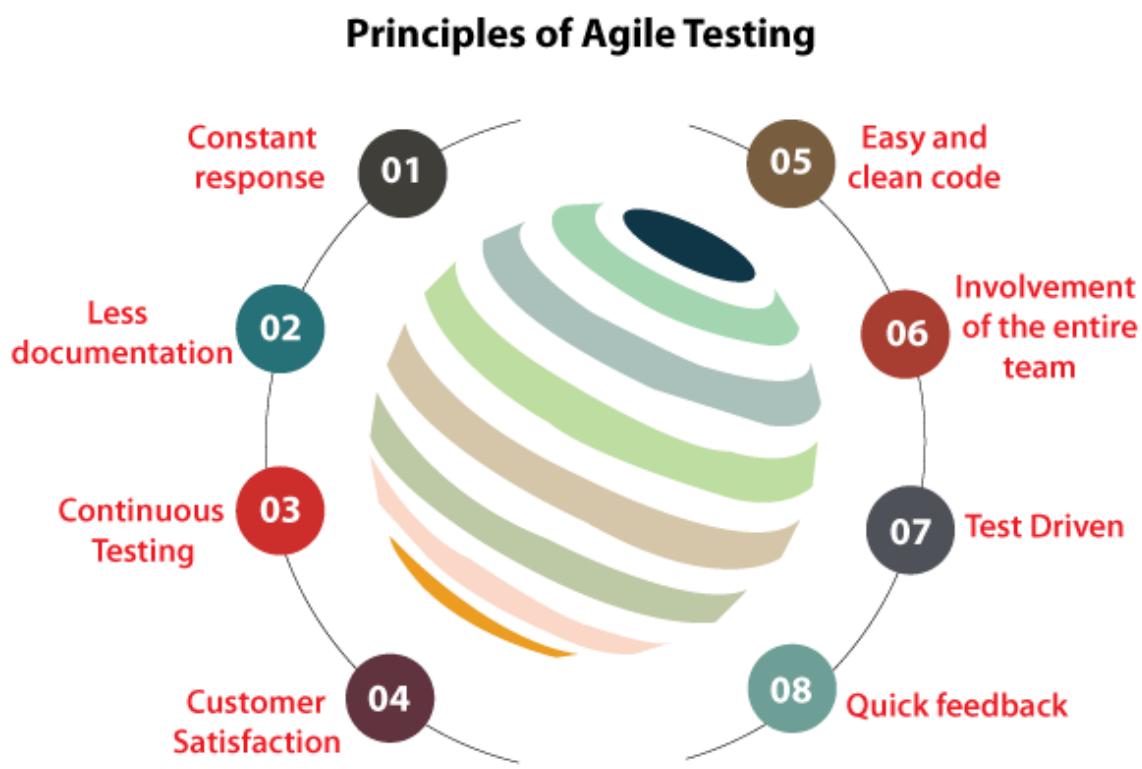
In the modern days of software testing, agile testing has gained a lot of **acceptance** and significance. The execution of agile testing will help us identify the initial error and elimination, giving better results in less development time and costs.

Principles of Agile Testing

Agile Testing includes various different principles that help us to increase the productivity of our software.

1. Constant response
2. Less documentation
3. Continuous Testing
4. Customer Satisfaction
5. Easy and clean code
6. Involvement of the entire team
7. Test-Driven
8. Quick feedback

For our better understanding, let's see them one by one in detail:



1. Constant Response

The implementation of Agile testing delivers a response or feedback on an ongoing basis. Therefore, our product can meet the business needs.

In other words, we can say that the Product and business requirements are understood throughout the constant response.

2. Less Documentation

The execution of agile testing requires less documentation as the Agile teams or all the test engineers use a reusable specification or a checklist. And the team emphasizes the test rather than the secondary information.

3. Continuous Testing

The agile test engineers execute the testing endlessly as this is the only technique to make sure that the constant improvement of the product.

4. Customer Satisfaction

In any project delivery, customer satisfaction is important as the customers are exposed to their product throughout the development process.

As the development phase progresses, the customer can easily modify and update requirements. And the tests can also be changed as per the updated requirements.

5. Easy and clean code

When the bugs or defects occurred by the agile team or the testing team are fixed in a similar iteration, which leads us to get the easy and clean code.

6. Involvement of the entire team

As we know that, the testing team is the only team who is responsible for a testing process in the **Software Development Life Cycle**. But on the other hand, in agile testing, the **business analysts (BA)** and the developers can also test the application or the software.

7. Test-Driven

While doing the agile testing, we need to execute the testing process during the implementation that helps us to decrease the development time. However, the testing is implemented after implementation or when the software is developed in the traditional process.

8. Quick response

In each iteration of agile testing, the business team is involved. Therefore, we can get continuous feedback that helps us to reduce the time of feedback response on development work.

How is Agile Methodology used in Testing?

Agile Testing is a fast and informal testing process. In simple terms, we can say that it is specified as an advanced and **dynamic type of Testing** that is performed regularly throughout every iteration of the **SDLC (Software Development Life Cycle)** by the agile test engineers.

If we deliver the software quickly with the best of the attributes, and the customer satisfaction is the primary concern at some stage in the agile testing process.

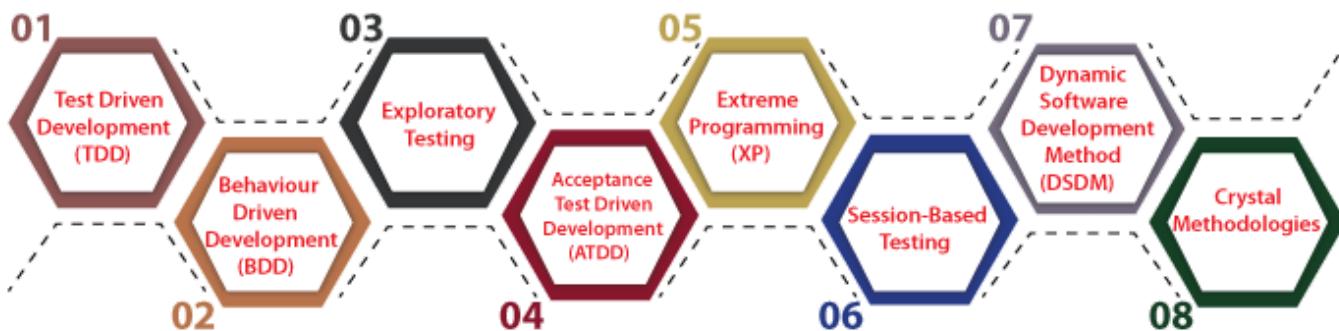
Agile Testing Methodologies

When we are executing the agile testing, the team takes help from several agile methodologies, which support them in accomplishing the precise results.

Some of the effective agile testing methodologies are as follows:

- **Test-Driven Development (TDD)**
- **Behavior Driven Development (BDD)**
- **Exploratory Testing**
- **Acceptance Test-Driven Development (ATDD)**
- **Extreme Programming (XP)**
- **Session-Based Testing**
- **Dynamic Software Development Method (DSDM)**
- **Crystal Methodologies**

Agile Test Methodologies



Test-Driven Development (TDD)

The test-driven development method begins with the test itself. As the name proposes, the TDD varies upon the repetition of the development cycle.

We already knew that the first step in of development cycle is to create **a unit test case**. And in the next step, we will be designing the code that fits the test case in order to execute the test cases.

Hence, the whole code is designed until the unit test passes. Generally, the test-driven development is executed by using the automated testing tools and implement on units and components of the code.

Behavior-Driven Development (BDD)

The following method in agile testing is **behavior-driven development**. The BDD enhances the communication between the project stakeholders to facilitate the members adequately and understood all the components before the development process begins.

It is constructed on the same rules as TDD and ATDD. Therefore, the code is developed as per the test case designed in this testing methodology too.

The primary purpose of this development is to emphasize the identification of business needs and outputs. And the development should be consistent to a business output.

In behavior-driven development, we need to follow the below steps:

1. Describe the behavior.
2. Generating the test case.
3. Writing code as per the test case is specified.
4. Continuing the process until the code passes the test case.

Exploratory Testing

In Software testing, **exploratory testing** is one particular type where the test engineers have the fundamental freedom to explore the code and create the most effective software.

In simple words, we can say that if we don't have the requirement, then we do one round of exploratory testing.

Exploratory testing is a very significant part of the agile test as it helps discover the unknown risks from the software that a simple testing approach could not have noticed.

To explore each aspect of the software functionality, the test engineer creates various test cases, executes different tests, and records the process to learn it and understand its particular flow.

While performing the exploratory testing, we need to follow the below steps:

- Exploring the application in all possible ways
- Understanding the flow of the application
- Preparing a test document

- Testing the application

For more information related to the exploratory testing, refers to the following link: <https://www.javatpoint.com/exploratory-testing>.

Acceptance Test-Driven Development (ATDD)

Another methodology of agile testing is **Acceptance Test-Driven Development (ATDD)**. The ATDD approach emphasizes the customer's requirements by involving the team members with different viewpoints.

The team's members of development, testing, and the customers come together in order to develop the acceptance test from the customer's perspective.

In **Acceptance Test Driven Development**, the code is acquired along with the developed acceptance test case.

It is a very customer-centered methodology; the primary objective of using the ATDD methodology is to develop a program based on the user's view.

Extreme Programming (XP)

The next significant agile methodology is **Extreme Programming** which is denoted as **XP**. When there is a continuous modification in the user requirements, we will go for the extreme programming methodology.

Just like other agile testing methodologies, **Extreme Programming** is also a **customer-centric** methodology.

The XP will help us deliver a quality product, which meets the customer's requirements that are made clear throughout the process of development and testing.

Session-Based Testing

In the row of various agile testing methodologies, the next methodology is Session-based testing. It is mainly created on the values of exploratory testing.

Though session-based testing contains some structure and on the other hand, exploratory testing is performed unexpectedly without any planning. It is used to help us identify the hidden bugs and defects in the particular software.

The session-based testing structure is prepared by executing the tests in continuous sessions where test engineers must report the tests, which took place throughout the process.

Dynamic Software Development Method (DSDM)

Another effective method of agile testing is **Dynamic Software Development Method**. It is a **Rapid Application Development** (RAD) approach that offers a delivery framework to agile projects.

In other words, we can say that the **Dynamic Systems Development technique (DSDM)** is a correlate degree agile code development approach, which gives a framework for developing and maintaining systems.

The **Dynamic Software Development Method** can be used by users, development, and testing teams.

Crystal Methodologies

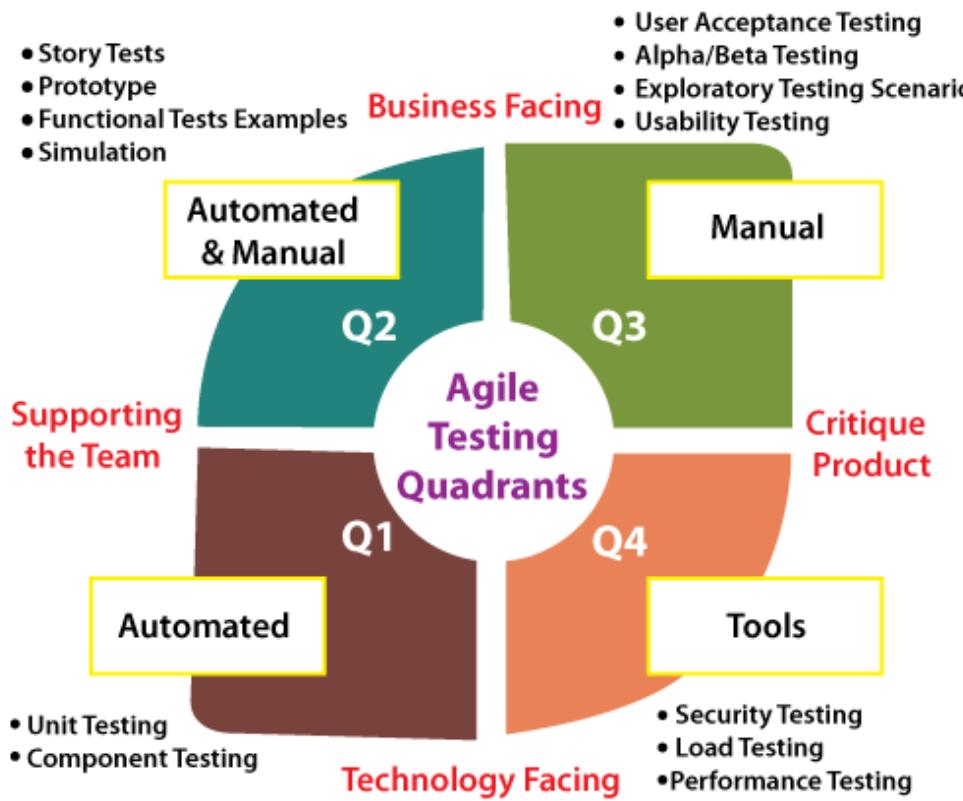
The subsequent agile testing is **crystal methodologies**. This methodology mainly emphasizes recording, cyclic delivery, and wrap-up, which is to make sure during the various analyses.

Agile Testing Quadrants

It has different quadrants to easily understand agile testing, which divides the whole testing process into **four** parts.

In addition to the four quadrants, the left two specify the test engineer that code to write, and the right two quadrants help them understand the code improved with the support of response to the left quadrants.

These agile testing quadrants may be understood as a traditional process or strategies to perform the end-to-end agile testing of a software application in four different stages, as we can see in the following image:



Let us discuss them one by one to understand the process of agile testing:

1. **Quadrant 1 (Automated)**
2. **Quadrant 2 (Automated and manual)**
3. **Quadrant 3 (Manual)**
4. **Quadrant 4 (Tools)**

Quadrant 1 (Automated)

In the first quadrant of Agile testing, we will see mainly emphasis on the quality of the code. We can say internal code quality, which contains the test cases and test components that is executed by the test engineers.

And these test cases are technology-driven and used for automation testing in order to enhance the code and support the testing team to perform their tasks.

All through the first quadrant of agile testing, we can execute the following testing:

- Unit Testing
- Component Testing

Quadrant 2 (Automated and manual)

In the second quadrant of Agile testing, we will see mainly emphasis on the customer requirements given to the team before and throughout the testing procedures, which expands the business results of the newly created software.

The test case involved in this second quadrant is **business-driven**, usually manual and automated functional tests, prototypes, and examples of test scenarios performed by the testing team.

In quadrant 2, we can execute the following tests:

- Testing scenarios which may occur and workflow
- Implementing the pair testing

- Testing the user story and experiences like prototypes.

Quadrant 3 (Manual)

The third quadrant of agile testing primarily emphasizes the response for the previous two phases (**Quadrant 1 and Quadrant 2**).

The execution of agile testing involves many iterations. And in this quadrant, these reviews and responses of the particular iterations are sustained that helps to strengthen the code.

To test the user experience and determine business results allows the testing team to learn as the test develops.

The team, business owners, and even customers realistically use the product. In the third quadrant, the test cases have been designed to implement automation testing, which helps us develop certainty in the particular product.

In quadrant 3, below types of testing can be executed:

- **Usability testing**
- **Collaborative testing**
- **Exploratory testing**
- **User acceptance testing**
- **Pair testing with customers**

Quadrant 4 (Tools)

The last and fourth Quadrant of agile testing primarily emphasizes the product's non-functional requirements, including compatibility, performance, security, and constancy.

In other words, we can say that the fourth Quadrant ensures that the code fulfills all the non-functional requirements.

Like other Quadrants, various types of testing are performed in quadrant 4 to deliver the non-functional qualities and the expected value.

- **Non-functional testing such as Stress testing, performance testing, and load testing, etc.**
- Scalability testing
- **Security Testing**
- **Data Migration Testing**
- Infrastructure testing

Agile Test Plan

As compared to the waterfall model, the agile test plan is created and updated for every release. Furthermore, the agile test plan contains those types of testing executed in a specific iteration, such as test environments, test data requirements, test results, and infrastructure.

The agile test plans emphasize the following:

- **Testing Scope:** The testing scope specifies the sprint goals, test scope, and test coverage in which the test will be implemented.
- **Performance and Load Testing:** Here, it specifies the different testing methods and procedures.
- **Types of testing or levels as per the feature's complexity:** It defines those types of testing or levels of testing which are going to be used. And also specifies the data and configurations for the test and the environment in which the test will be executed.

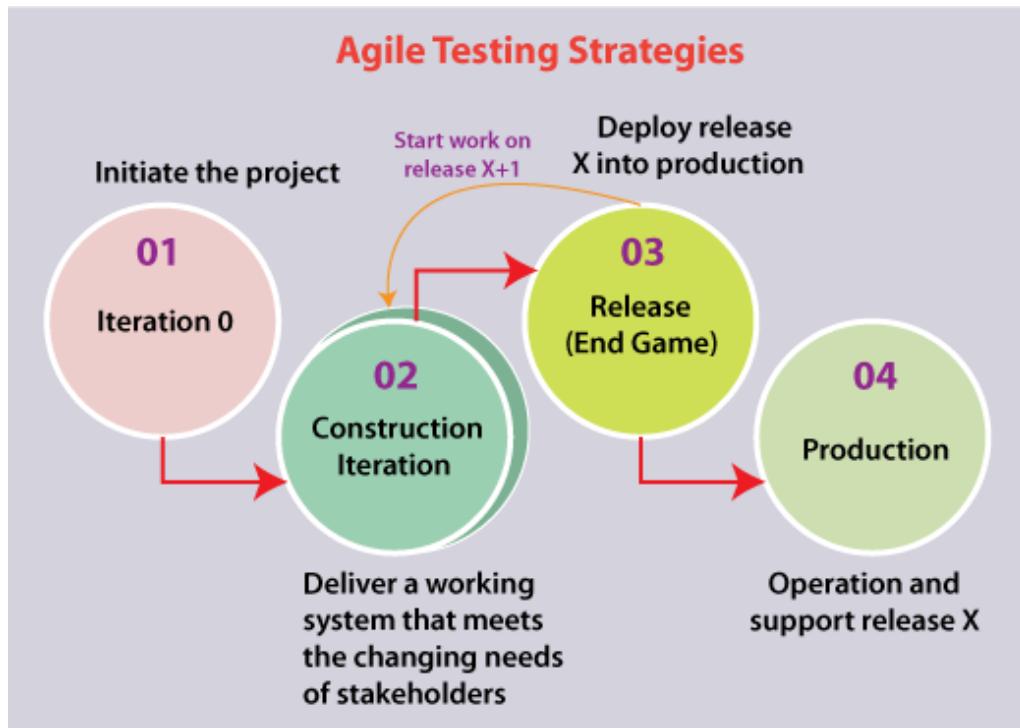
- **Mitigation or Risks Plan:** It defines the backup plan prepared to overcome the risks or issues. And it also identifies the challenges which might face at the time of testing of the application in the current release.
- **Deliverables and Milestones:** It sets the deliverables and milestones of the tests as per the customer's perspective.
- **Infrastructure Consideration:** It governs the infrastructure which is required to perform the tests.
- **Resourcing:** It lists out the test tasks and the occurrence of tests, which defines how many times the tests will be executed.
- **Establish the New functionalities** which are being tested.

Agile Testing Strategies

Agile testing has four different approaches, which help us to enhance our product quality.

- 1. Iteration 0**
- 2. Construction iteration**
- 3. Release End Game or Transition Phase**
- 4. Production**

Let us discuss them one by one in detail:



1. Iteration 0

The first strategy or approach of agile testing is **iteration 0**. In this, we execute the preliminary setup tasks such as **finding people for testing, establishing testing tools, preparing resources or usability testing lab, etc.**

In Iteration 0, the below steps are accomplished:

- Verifying a business case for the project and boundary situations, and the project scope.
- Summarise the important requirements and use cases that will determine the strategic trade-offs.
- Plan the initial project and cost valuation
- Detecting the risk.
- Outline one or more candidate designs

2. Construction Iteration

The next strategy of agile testing is **Construction Iteration**. During this approach, the majority of the testing is performed.

The construction iteration is performed as a set of iterations in order to create an increment of the solution.

In simple words, we can say that the agile team follows the listed requirement within each iteration where they can acquire the most significant business needs or requirements left behind from the work item stack and then execute them.

The construction iteration process divided into the following two types of testing:

- **Confirmatory Testing**
- **Investigative Testing**

1. Confirmatory Testing

To ensure that the product meets all the stakeholders' requirements, we will execute the confirmatory testing.

Confirmatory testing can be further separated into another two types of testing, which are as follows:

- **Agile Acceptance Testing**
- **Developer Testing**

Agile Acceptance Testing: It is a combination of [functional testing](#) and [acceptance testing](#). The agile acceptance testing can be executed by the development team and stakeholders together.

Developer Testing: It is a combination of [unit testing](#) and [integration testing](#). And it validates both the application code as well as the database schema.

Note: We can automate both (agile acceptance testing and developer testing) to ensures that continuous regression testing has occurred.

2. Investigative Testing

In order to test deep and identify all the issues, which are overlooked in [confirmatory testing](#), we will execute the [investigative testing](#).

3. Release End Game or Transition Phase

The third approach of agile testing is [release](#). The objective of this specific approach is to implement our system effectively in production.

The test engineer will be working on its defect stories in the end game. In the release end game or transition stage, we have the following activities:

- Support Individuals
- Training of end-users
- Operational People

Similarly, it involves some additional activities as well:

- Back-up and Restoration
- Marketing of the product release
- User documentation
- Completion of system

The last agile methodology testing stage encompasses whole system testing and acceptance testing. To complete our final testing phase without having any difficulties, we should have to test the product more thoroughly in construction iterations.

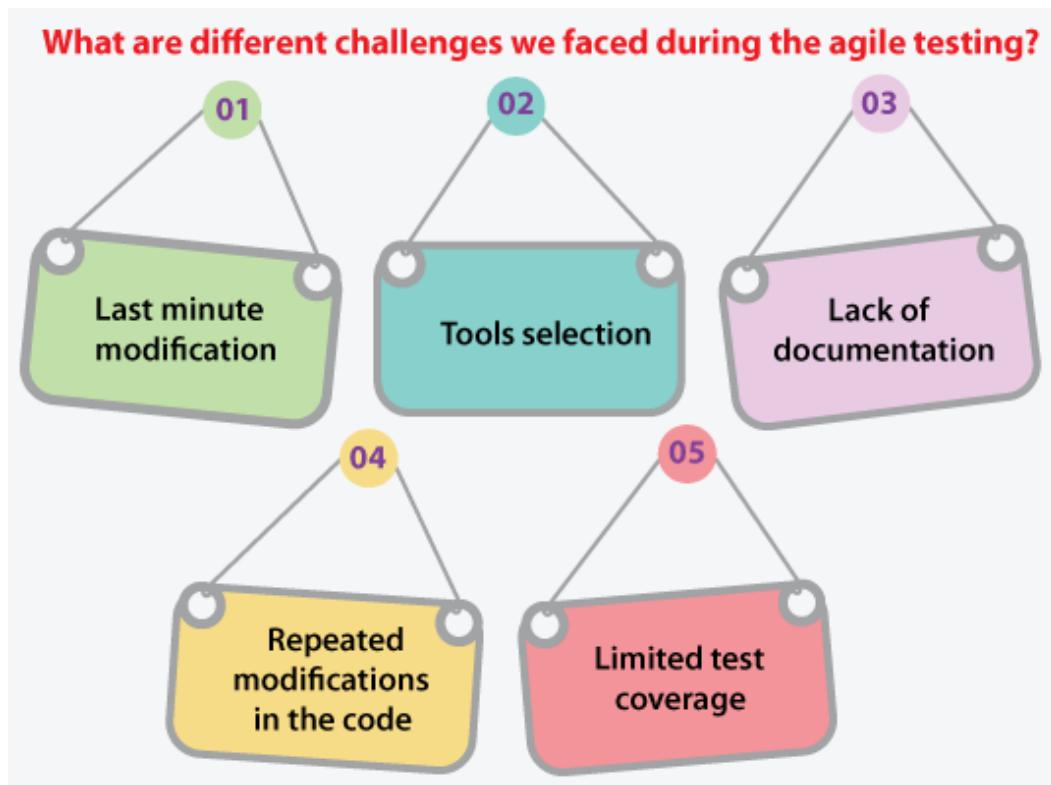
4. Production

The product will move on to the **production stage** as soon as the **release stage** is completed.

What are the different challenges we faced during the agile testing?

Generally, while performing agile testing, a testing team may face some challenges. Let see those challenges all together for our better understanding:

- **Last-minute modification**
- **Tools selection**
- **Lack of documentation**
- **Repeated modifications in the code**
- **Limited test coverage**



- **Last-minute Modification**

The most faced challenges during the agile testing are last-minute modifications by the client, which gives significantly less time to the testing team to design the test plan, which may affect the product quality. And sometimes, the test engineer is often required to play a semi-developer role.

- **Tools Selection**

The selection of tools during agile testing is essential because if we select the wrong tool, it will waste our time as well as money.

As we already knew, the Test execution cycles are highly reduced, and for the regression testing, we will have minimal timing.

- **Lack of Documentation**

Another frequently faced challenge while executing agile testing is the lack of documentation. The probabilities of error are more agile as documentation is given less importance and ultimately puts more burden on the testing team.

- **Repeated Modifications in the code**

In an agile method, requirement modification and updation are fundamental, making it the major challenge for the Quality assurance team.

- **Limited Test Coverage**

In agile testing, new features are initiated quickly, decreasing the available time for the testing teams to find whether the latest features are as per the requirement and address the business suits.

After seeing all the frequent challenges, the question arises how do we overcome them? Therefore, in the below topic, we are going to discuss that:

How do we overcome Agile testing challenges?

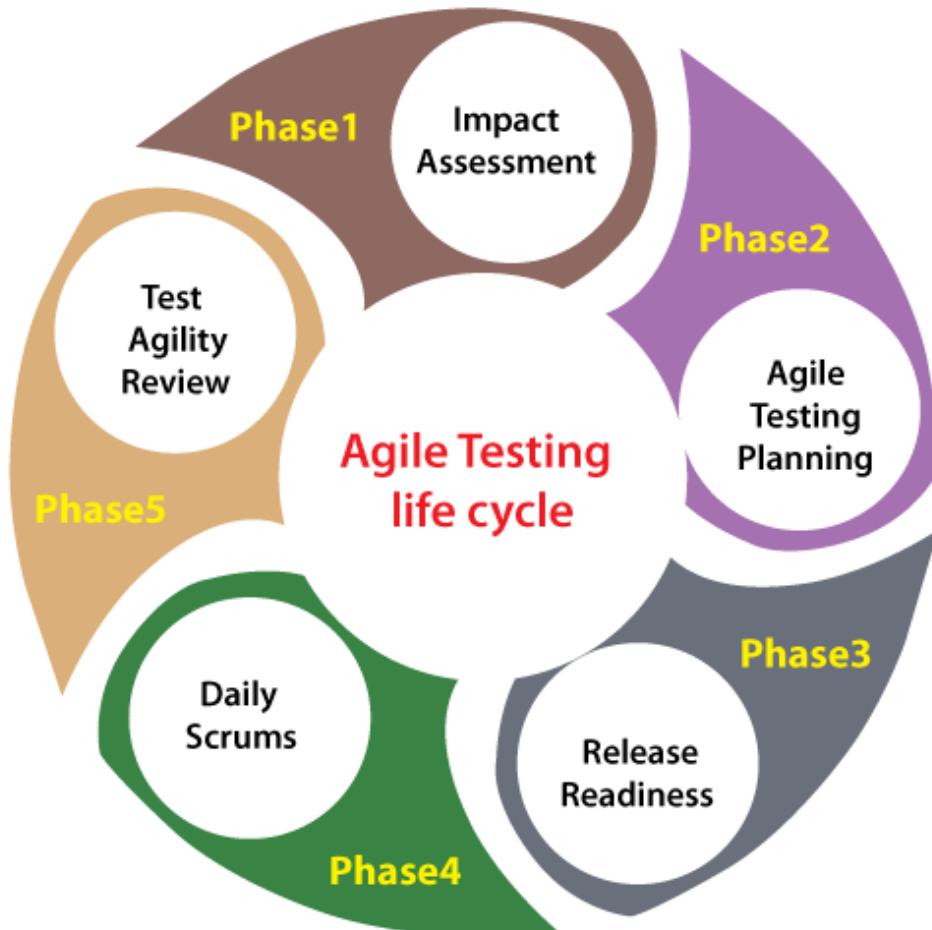
As we understood from the definition of Agile Testing, it comprises less or no documentation, which creates problems for the testing team to predict the expected results and becomes the obstacle in the testing process.

And it also makes it challenging to choose the direction and path of the testing to be performed. Hence, to overcome the agile testing challenges, we can implement the following best options:

- We can execute the Exploratory Testing to conquer the agile testing challenges.
- We can perform the automated unit tests to speed up the agile testing process.
- Test-Driven Development could be a good option in order to overcome the agile testing challenges.
- Furthermore, we can overcome these issues or challenges with the help of the agile testing specification and make sure to perform improved and qualitative Testing in a guided way.
- We can implement automated Regression Testing.

Agile Testing life cycle

Just like other types of testing has their life cycle process, Agile Testing life cycle completed into five different phases, as we can see in the following image:



Let understand all the phases in detail:

Phase1: Impact Assessment

The first phase of the Agile testing life cycle is **Impact assessment**. Here, we collect the inputs and responses from users and stakeholders to execute the impact assessment phase. This phase is also known as the feedback phase, which supports the test engineers to set the purpose for the next life cycle.

Phase2: Agile Testing Planning

The second phase of the Agile testing life cycle is **agile testing planning**. In this phase, the developers, test engineers, stakeholders, customers, and end-users team up to plan the testing process schedules, regular meetings, and deliverables.

Phase3: Release Readiness

The next phase of the Agile testing life cycle is **release readiness**, where test engineers have to review the features which have been created entirely and test if they are ready to go live or not and which ones need to go back to the previous development phase.

Phase4: Daily Scrums

Daily scrums are the next phase of the Agile testing life cycle, which involves the daily morning meetings to check on testing and determine the objectives for the day.

And, in order to help the test engineers to understand the status of testing, the goals and targets of the day are set daily.

Phase5: Test Agility Review

The last and final phase of the Agile life cycle is the **test agility review**. The test agility phase encompasses the weekly meetings with the stakeholders to evaluate and assess the progress against goals.

In other words, we can say that the agility reviews are implemented regularly in the development process to analyze the progress of the development.

Advantages of Agile Testing

For the past few years, Agile software testing has been a significant part of the IT field. Here, we are discussing some essential benefits of Agile testing:

- Agile testing gives way to get regular **feedback** and reviews directly from the end-user, which helps us enhance the software product's quality and attribute.
- The implementation of agile testing will save lots of **time** and **money**, making the estimation of cost more transparent.
- Through daily meetings, we can determine better issues.
- Agile testing reduces **documentation**, or we can say it requires less documentation to execute the agile testing.
- The most significant advantage of implementing Agile software testing is reducing errors and enhancing software productivity.
- As we understood from the above discussion, the workload is divided into small parts in agile software development and restricts the developer from going off the track. And in the results, we will get more minor inconsistencies and higher efficiency.

Disadvantage of Agile Testing

Agile testing is a creative method to test the software application, but still, we have some disadvantages of using Agile Testing:

- The most significant disadvantage of agile testing is that if two or more members leave a job, it will lead to project failure.

- Determination is needed while performing any testing to test the application or the software and becomes inconsistent for the testing team.
- It makes it difficult for us to predict expected results because there are no or fewer documentation results into explicit conditions and requirements.
- Sometimes, it leads us to introduce new bugs in the system because the bug fixes, modifications, and releases repeatedly happen in agile testing.

Overview

In this tutorial, we have seen the in-depth knowledge of agile testing like **Principle of Agile Testing, Strategies, Quadrants, agile testing life cycle, different challenges we faced during the agile testing, and Advantages/Disadvantages of Agile Testing.**

After understanding all the topic mentioned earlier, we can say that Agile testing is one of the best approaches for the present-day software as this software are highly complex and demand comprehensive testing.

It allows the test engineer to be flexible and able to encompassing any requirement modification.

Agile testing is becoming very famous among significant software development organizations as it a very customer-centric one that ensures a high-quality product.

The execution of agile testing requires the involvement of customers, a high level of communication among the developers, test engineers as they all are working together in order to test the particular software.

And in a result, we can deliver a better quality of software that fulfils the customer and user expectations.

In software testing, the agile methodology encompasses testing as soon as possible in the **Software Development Life Cycle.**

At last, we can say that the communication among the teams (development team, testing team, customers) plays an essential role in making the agile testing successful.

Component Testing

A software is developed with the help of several modules or components. Here, we are going to explore and discuss the following topics related to **Component Testing**, which help us understand the requirement of component testing and product dependability in software testing.

- **What is Component Testing?**
- **The goal of Component Testing**
- **Who performs Component Testing?**
- **Component Testing process**
- **When do we need to perform the Component testing?**
- **Why Is Component Testing Essential?**
- **The Different test strategy of Component Testing**
- **Example of Component Testing**

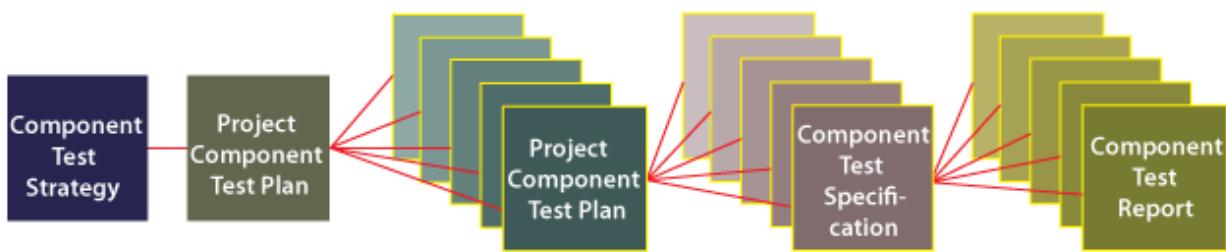
Before going deep into all the topics mentioned above, we are very clear that no matter what, all the testing methods have an objective, and that's why all of those should be performed accurately.

What is Component Testing?

Another type of software testing is **Component Testing**. It is used to test all the components separately as well as the usability testing; interactive valuation is also done for each specific component. It is further known as **Module Testing or Program Testing and Unit Testing**.

In order to implement the component testing, all the components or modules require to be in the individual state and manageable state. And all the related components of the software should be user-understandable.

This type of testing provides a way to finding defects, which happen in all the modules. And also helps in certifying the working of each component of the software.



Components testing is one of the most repeated types of **black-box testing** executed by the **Quality Assurance Team**.

It can be performed individually, that is, in separation from the remaining system. However, it has relied on the model of the preferred life cycle.

The **debugging** or **test structure tools** can be used in the component testing.

In component testing bugs can be fixed as soon as possible when they are identified without keeping any records.

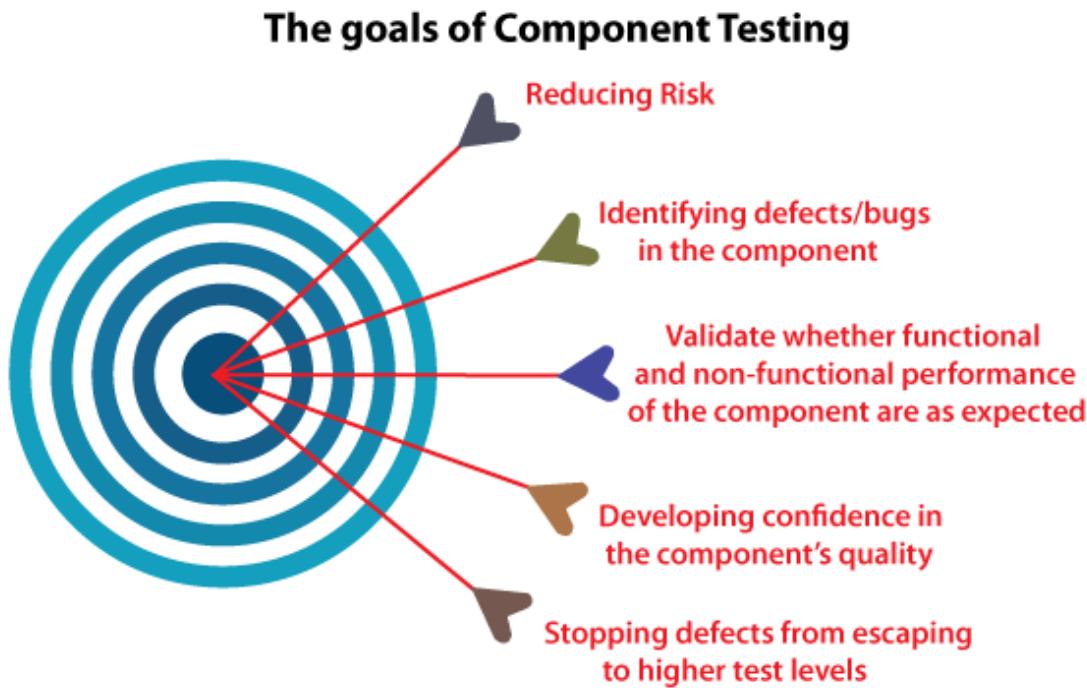
In simple words, we can say that the execution of components testing makes sure that all the application component is working correctly and according to the requirements. Component testing is executed before handing out with the **integration testing**.

The Goals of Component Testing

The primary purpose of executing the component testing is to validate the input/output performance of the test object. And also makes sure the specified test object's functionality is working fine as per needed requirement or specification.

Let's see some of the other vital goals of component testing:

- Reducing Risk
- Identifying defects/bugs in the component
- Validate whether the functional and non-functional performance of the component is as expected
- Developing confidence in the component's quality
- Stopping defects from escaping to higher test levels



- Reducing Risk

The implementation of components testing validates every single unit of the application. And helps the developers in order to identify the bugs in the code and fix them. Thus, we can say that component testing diminishes the possibilities of risk at a fundamental level.

- Identifying defects/bugs in the component

Another essential purpose of performing the component testing is to **identify errors** in the source code. Furthermore, it also validates **control flow, functions, data structure, etc.**, which are used in the program.

- Validate whether functional and non-functional performances of the component are as expected

The purpose of executing the component testing is to verify that the functional and non-functional features of the component are functioning correct or not.

In other words, we can describe that the execution of component testing guarantees their design and specifications are performing as expected.

It may involve functional features such as **the correctness of calculations** and **non-functional** features like **exploring memory leaks**.

- Developing confidence in the component's quality

As we understood from the definition of the component testing, it has occurred on the unit level, and most of the errors are identified and fixed while coding itself.

The components testing plays a crucial role in building confidence in the component, which means fewer bugs or defects in the additional testing.

- Stopping defects from escaping to higher test levels

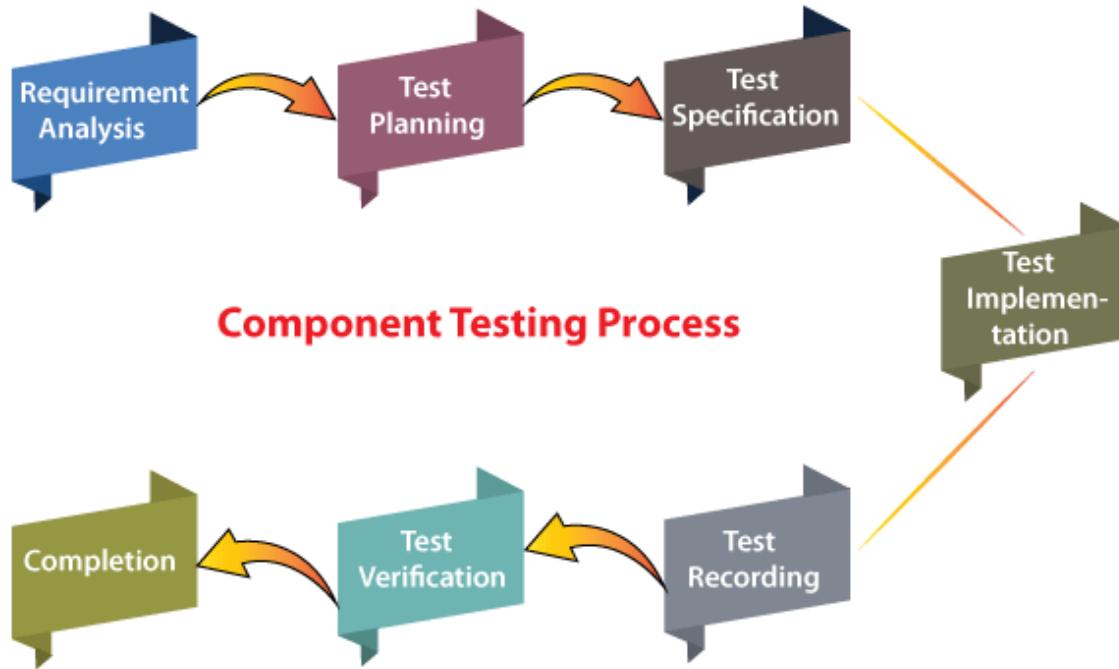
Conclusively, in Component testing, the coding errors are identified and fixed by the developers. And as an outcome, component testing diminishes the existence of errors in the higher level of testing.

Who performs Component Testing?

The component testing is performed by **test engineers or the developers** who write the codes with IDE help. As we already knew, **Unit Testing** is executed by the developers to perform the testing of the particular features or methods.

Component Testing Process

The process of component testing can be completed in the following seven steps, as we can see in the below image:



Let's discussed them one by one for our better understanding:

Step1: Requirement Analysis

The first step of component testing is **requirement analysis**, where the user requirement associated with each component is detected.

Step2: Test Planning

Once the **requirement analysis** phase is done, we will mov to the next step of component testing process, which is **test planning**. In this phase, the test is designed to evaluate the requirement given by the users/clients.

Step3: Test Specification

Once the **test planning phase** is done, we will move to the next phase, known as **test specification**. Here, we will identify those test cases that needs to be executed and missed.

Step4: Test Implementation

The forth step in the component testing process is **Test implementation**. When the test cases are identified as per the user requirements or the specification, then only we can implement the test cases.

Step5: Test Recording

When all the above steps have been completed successfully, we will go to the next step that is, **Test Recording**. In this step of the component testing process, we have the records of those defects/bugs discovered during the implementation of component testing.

Step6: Test Verification

Once the bugs or defects have been recorded successfully, we will proceed to the **test verification** phase. It is the process of verifying whether the product fulfils the specification or not.

Step7: Completion

After completed all the above steps successfully, we will come to the last step of the component testing process. In this particular step, the results will be evaluated in order to deliver a good quality product.

When do we need to perform the Component testing?

When the unit testing is done on the particular application, we can proceed with the component testing. The components are tested once they are developed; thus, the output retrieved from a component under test depends on other components that are not created so far.

According to the **development lifecycle model**, component testing might be executed in segregation with other system components.

And the segregation is performed to stop the outside effects. Therefore, we will use **Stubs and Drivers** to pretend the interface between software components to test that component.

Note: The Integration testing is performed only after the execution of component testing.

The major functionality of all the modules is tested in component testing, such as:

Entry criteria for component testing	Exit criteria for component testing
The least number of components involved in the unit testing should be created, and unit tested.	The features of all the components/ modules should be working fine. And there is no Critical or High, or Medium severity and priority defects available on the Defect log.

Why Is Component Testing Essential?

In Software Testing, component testing plays an essential role. Let's understand this with the help of the following points:



Allows Detailed Examination

The execution of components testing is allowing the detailed examination once each module or component has been acquired.

It can be tested entirely or exhaustively for all the possible bugs or errors on actual web servers. And this fact is entirely exceptional to the **Component Testing** as we compared to the Unit Testing where every unit is not tested on live servers.

Early Error Detection

As we understood from the above explanation of Component Testing, it can be implemented at any stage, which helps us identify and fix those errors relatively soon, and as a result, we save both money and lots of time.

And the development team can test the component for all the preventable bugs or defects before giving it over to the quality assessment team.

Certifies Contract

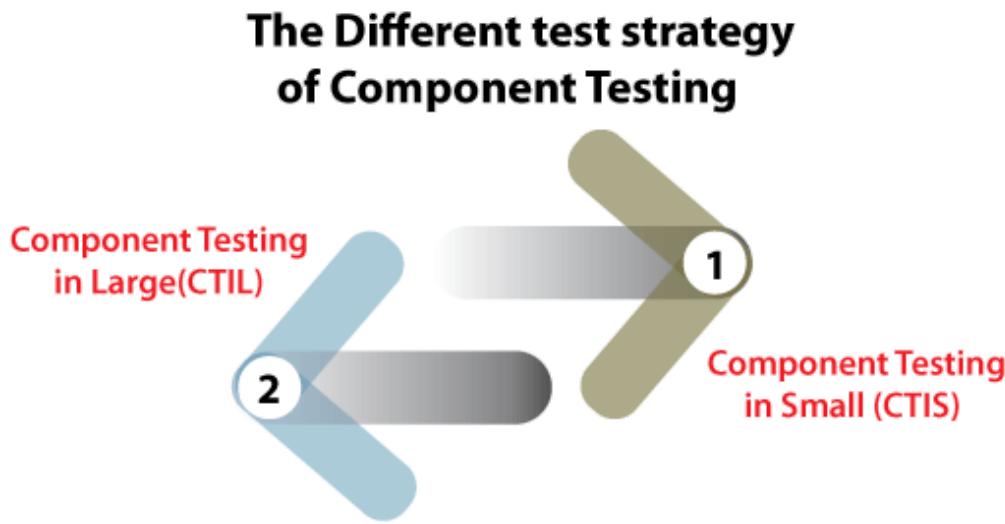
The implementation of Component Testing will help us to **certify the contract**, which implies that the component testing is repeatable.

So, we can say that component testing is the best way to authenticate our module, which delivers the whole thing that it promised. And as a consequence, any integration with the given software can be performed without any uncertainties and doubts.

The Different test strategy of Component Testing

The Component Testing is separated into the following two types, based on the testing level's complexity.

1. **Component Testing in Small (CTIS)**
2. **Component Testing in Large (CTIL)**



For our better understanding let's discussed them one by one:

1. Component Testing in Small (CTIS)

Some component testing may be performed with or without other modules in the particular application or the software under a test. If the component testing is executed in the segregation with other modules is signified as **component testing in small**, which is also denoted as **CTIS**.

Let see one sample example, where we can clearly understand how Component testing in small works.

For Example: Suppose we have one website, which includes five different web pages. Therefore, testing each webpage individually and with the isolation of other components is signified as **Component testing in Small**.

2. Component Testing in Large (CTIL)

The **component testing in large** is testing, where we execute the component testing without segregation with having other modules of the software.

This type of testing has occurred when there is a dependency on the performance flow of the modules, and consequently, we can't separate them. Those components on which we have the dependency are not created yet, and then we use the replicate modules instead of the actual modules or components.

Note: These replicated modules are known as the Stubs (called function) and Drivers (calling function).

Let us see an example of **component testing in large** for our better understanding.

For Example

Let us assume that we have a web application containing three different modules: **Module P**, **Module Q**, and **Module R**.

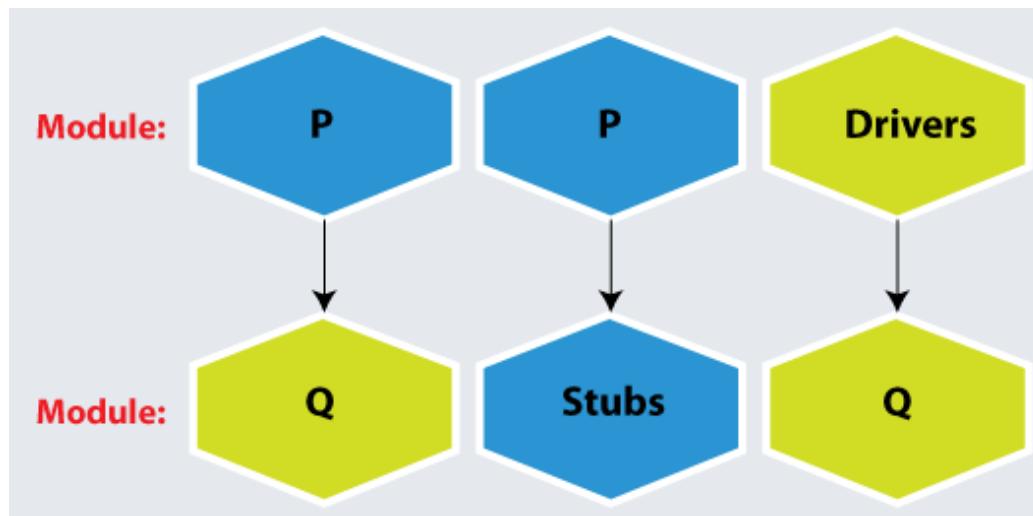
Here, the developer has created **Module Q**, which needs to be tested.

However, to test **Module Q** entirely, some of its features are reliant on **Module P**, and some of its features are reliant on **Module R**.

So that, performance flow of the particular modules are as follows:

Module P → Module Q → Module R

This performance flow implies that there is a dependency to **Module Q** from both **Module P** and **Module R**. As we can see in the below image, we have Stubs and Driver where stubs are called as the **called function**, and the drivers are called as the **calling function**.



But the **Module P** and **Module R** have not been established until now. So, in this scenario, if we want to test **Module Q** entirely.

Thus, we can interchange **Module P** and **Module R** with the help of **Stub and Drivers** as per the requirement.

Hence, **Module P** and **Module R** are primarily changed by **stub and Driver** that are performed as a replica object until they are created.

Note: Stub: A Stub is called from the software module or component needed to be tested; as we can see in the above image, Stubs is called by a Module P. Driver: A driver calls the module/component, which needs to be tested as we can see in the above image that Driver calls module Q.

Example of Component Testing

Suppose we have a web application containing three different modules as **Login**, **Home**, and **User**.

The first module (**Login**) is installed in the testing environment, but the other two modules, **Home** and **User**, need to be called by the **Login module** that is yet to be finished.

For testing purpose, the developer will add a segment of code that replicates the call methods of the remaining modules. And this specific segment of code is known as **Stubs** which is considered a **top-down approach**.

If **second module (Home)** and **third module (User)** are ready, but the **Login** module is yet to be developed, from where both components get their return values, the developer will add a code that replicates the Login module.

And this specific segment of code is known as a **Drivers**, which is considered a **bottom-up approach**.

Conclusion

In this section, we have discussed the in-depth knowledge of **Component Testing**. And, we can conclude that it is an exact way of estimate the functioning of any module at any stage of development.

In software development, component testing is cost-saving and saves an unnecessary headache at a future stage by removing all avoidable bugs.

Al last, we can say the **Unit testing and Component testing** are executed simultaneously. As opposed to Unit testing executed by the development team, the Component/module testing is executed by the Testing team.

It is always suggested to have complete Component testing performed before starting the **Integration testing**. Because if the Component testing is accurate, we will identify more minor defects during the integration testing.

GUI Testing

In this section, we are going to discuss all GUI (**Graphical User Interface**) Testing, which includes the following important topics:

- **What is GUI Testing?**
- **Why do we need to perform GUI Testing?**
- **Features of GUI Testing**
- **Types of GUI Testing**
- **Examples of GUI Testing**
- **Different challenges faced during the GUI testing**
- **GUI Testing Tools**

Before getting deep into all the above mention topics, firstly, we will understand Graphical User Interface (GUI).

What is GUI?

In the computer application, we have mainly two types of interfaces such as:

- **Command Line Interface (CLI)**
- **Graphical User Interface (GUI)**

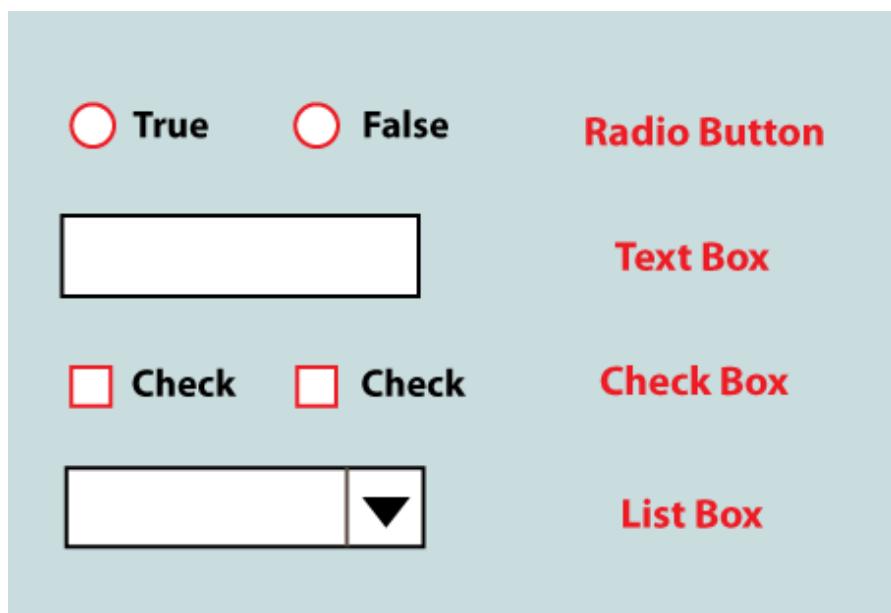
The **Command-line interface** is used when we need to type text, and at the same time computer responds to that command.

On the other hand, the **Graphical user interface** is used to interrelate along with the computer by using the images rather than text.

We have some of the critical GUI elements that can be used for communication between the user and application.

- **Check Box**
- **List Box**
- **Radio button**
- **Text Box**

As we can see in the below image:



Now, we will move to our main point of discussion which is GUI testing.

What is GUI (Graphical User Interface) Testing?

It is one of the unique **types of software testing** that is frequently used to check the Graphical user interface features for the application or the software.

Usually, the GUI testing is used to assesses a design of elements or features like:

- **Text boxes**
- **Font size**
- **Font color**
- **Buttons**
- **Menus**
- **Links**
- **Layout**
- **Labels**
- **Text Formatting**
- **Lists**
- **Captions**
- **Icons**
- **Content**

The primary goal of the GUI testing is to validate the features of the software, or the application performs as per the given requirement/specifications.

The **Graphical user interface testing** process implemented either manually or automatically and repeatedly executed by the **third-parties organization** instead of the developers or the end-users.

In other words, we can say that GUI testing is an approach in which the application's user interface is tested if the software or an application works as expected relating to the user interface performance.

Why do we need to perform GUI Testing?

After understanding the definition of GUI testing, we get the basic idea of it. But some question will arise like:

- Is GUI testing required to test the Application?
- Why do we need to perform GUI testing?
- And does testing of features and logics of Application is not too much?? Then why do we need to waste time on executing the UI testing?

If we want an answer to all the questions mentioned above, we need to think like a user, not a test engineer. Because a user doesn't have any exposure to the specified application or the software, it is a user interface of the application that determines whether a user will use the application more or not.

Here, firstly, a regular user understands the looks and design of the application or the software and how easy it is for her/him in order to understand the user interface.

He/she would never be going to use that Application again if a user is not happy with the interface or unable to identify the Application's difficulties in order to understand it. Because of the above scenarios, graphical user interface testing must be implemented to ensure that GUI testing delivers a defect-free application.

Why is Graphical user Interface testing being necessary?

In software testing techniques, the implementation of a Graphical user interface is essential for executing the other types of **software testing**.

As we know, delivering a quality product and matching the customer's requirement and a bug-free product is the primary concern of executing any type of testing.

And it has been stated that "**we cannot review quality into a product.**"

That's why in order to enhance the quality of a product, the development teams seek to develop it into their projects from the beginning.

To improve the quality of the product, we can use the testing process earlier in the **SDLC (Software Development Life Cycle)**, which is also known as **Shift Left testing**.

The development teams enhance the time and resources expended in the unit and interface testing instead of waiting for **system testing** after an application is complete. And in result, the early error detection in the development process will cut down the costs of fixing them.

As we are already aware, the **unit and interface/API tests** are compatible for automation because the developers develop the unit tests as they code; on the other hand, the APIs tests lean to be very stable and involve less maintenance than the APIs GUI tests.

We can observe that the importance of **Shift Left testing** is enchanting, making the GUI testing vulnerable. Despite everything, manual GUI testing can be a time taking and resource-intensive process.

Whereas the test automation is a bit more stimulating for GUIs because the user interface can frequently modify, earlier functioning automated GUI tests may fail, demanding a considerable determination to maintain them.

However, the **unit and interface testing** cannot assess all areas of a system, particularly the crucial features of **workflow** and **usability**. And, this is the importance of GUI testing, which is implemented from the user's point of view rather than a developer.

By evaluating an application from a **user's perspective** helps us to deliver the information to the project team where they need to choose whether an application is ready to deploy or not.

For example, The Drop-down list that appears in the **Windows Firefox browser** will be different from the **mac-Firefox**. These issues can be adequate because these are operating system functions, and we need to accept them in the same way.

Features of GUI Testing

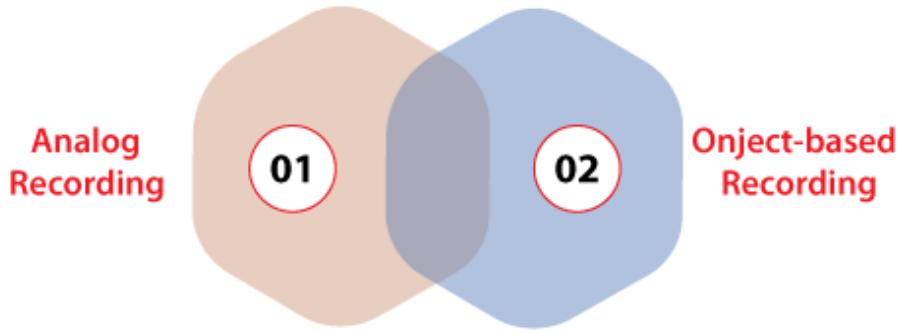
Some of the most significant features of Graphical user interface (GUI) testing are as discussed below:

- The GUI testing is used to execute the tests in matching or allocate on a Selenium Grid with fixed Selenium Web Driver.
- The execution of GUI testing will allow us to test the feature of an application from a user's point of view.
- As a result of Graphical user interface testing, we can get the customize test report.
- It also produces a consistent object documentation, at the same time for web elements along with the dynamic IDs.
- Sometimes the internal performance of the system works correctly, but the user interface doesn't; that's why GUI testing is an excellent approach in order to test other types of applications as well.

Types of GUI Testing

The Graphical User Interface testing divided into two different types, which are as discussed below:

Types of GUI Testing



Analog Recording

The first type of Graphical user interface testing is **Analog Recording**. With the help of analog recording, people will always be connected with the GUI testing tools.

Essentially, the GUI testing tools are used to encapsulates the precise **keyboard presses, mouse clicks, and other user activities** and then stores them in a file for playback. Let see one example to understand the basic functionality of Analog Recording.

Example

The analog recording might record that a user left-clicked at position X = 700 pixels, Y = 600 pixels, or entered the word "**Sign-in**" in a box and then pressed the **ENTER** key on their keyboard.

Object-based Recording

Another GUI testing type is **Object-based recording**. In this, the testing tool can connect programmatically to that application, which needs to be tested and **observe** each of the specific user interface modules, such as a **text box, button, and hyperlink, as a distinct object**.

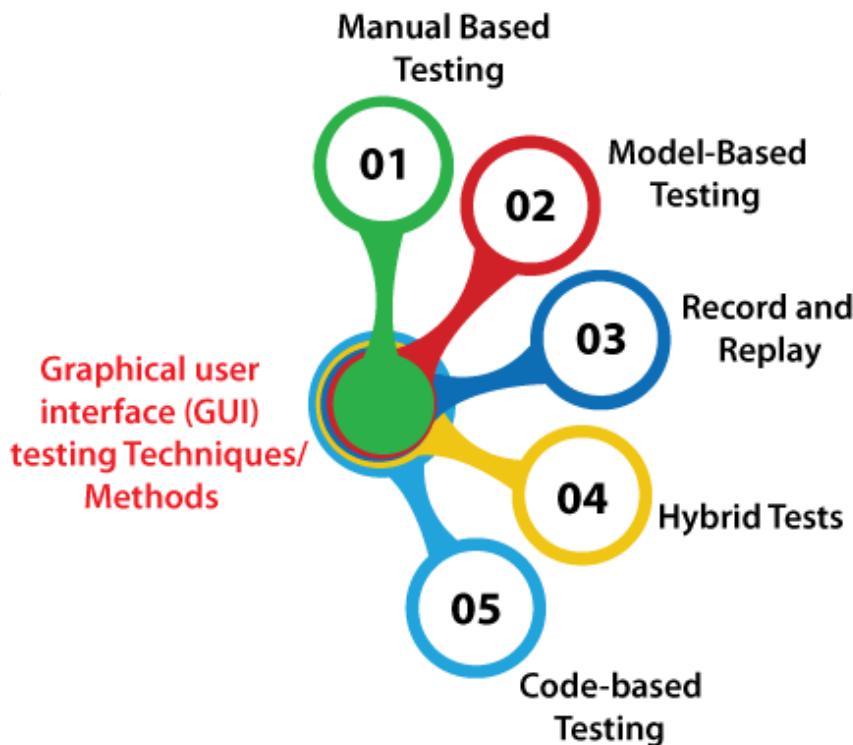
In the object-based recording, we can execute the following activities such as

- **Click**
- **Enter text**
- **Read the state(whether it is enabled or disabled)**

After seeing all the types of GUI Testing, we will move to our next topic, which is **Graphical user Interface Testing Techniques or different methods**.

Graphical user interface (GUI) testing Techniques/Methods

We have some unique techniques in order to execute the GUI testing, which are as follows:



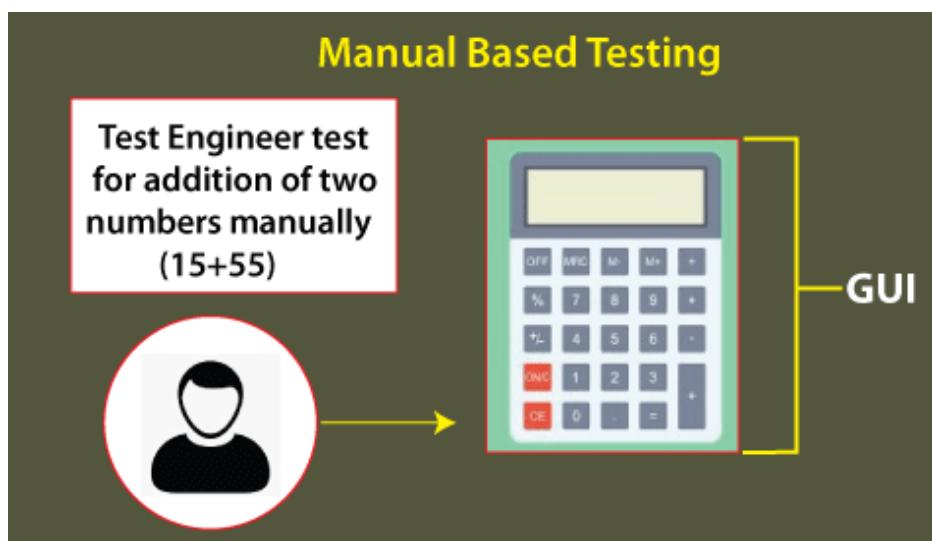
- **Manual Based Testing**
- **Model-Based Testing**
- **Record and Replay**
- **Hybrid Tests**
- **Code-based Testing**

Let see them in details for our better understanding:

1. Manual Based Testing

The first method of GUI testing is **Manual based testing**. The simplest way of executing the GUI testing is purely using the application manually. Usually, manual-based testing is implemented by enthusiastic parament test engineers.

In other words, we can say that in this approach, the graphical projects are tested manually by the test engineer following the requirements specified in the **BRS (Business Requirements Specification)** document.



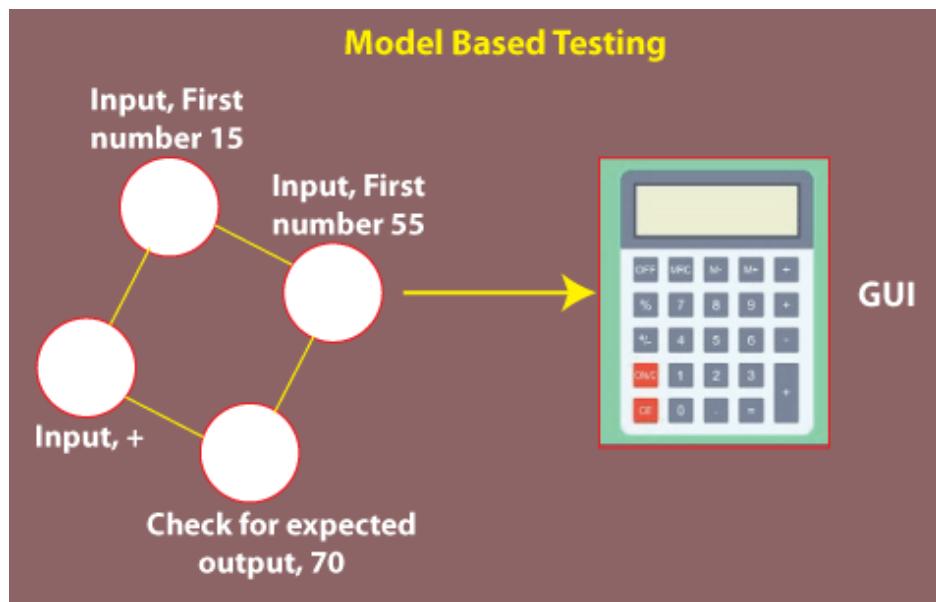
As we already knew that the **Manual testing** is not productive because sometimes the execution of manual testing is slow, monotonous, and error-prone. If we want to release high-quality software appropriately, we should try to automate our testing strategy insistently.

But in a current testing strategy, manual testing still plays an important role. We identify the right balance to execute the manual testing. Precisely talking about GUI testing, the manual test engineer could have more subjective facets of the interface, like its **look and feel and usability**.

2. Model-Based Testing

The next approach of GUI testing is **Model-based Testing** as we knew that a model is a visual narrative of System performance, which helps us to understand and predict the system performance or activity.

The models are beneficial in order to develop a practical test case with the help of the system requirements.



Some of the essential requirements need to be considered while executing the model-based testing approach:

- Create the model
- Verify the inputs for the model
- For the particular model analyze the expected result
- Implement the tests
- Balance the actual result with the expected result
- An evaluation on added action on the model

It is also a growing procedure in order to create the test case with the help of the given requirements. Compared to the other GUI testing approaches, model-based testing provides the benefits of fixing the adverse states that our GUI can accomplish.

We can obtain the test cases with the help of some the other model-based approaches:

- **Decision tables**
- **Charts**

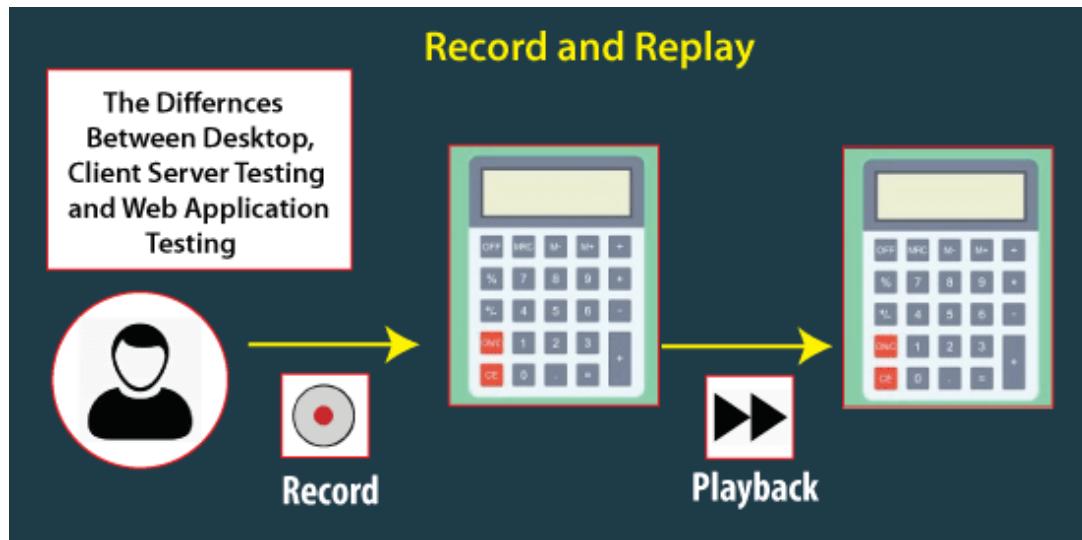
Decision Tables: The decision tables are use to control the outputs for each related input.

Charts: The charts technique represents the state of a system and tests the state after some input.

3. Record and Replay

We can perform the GUI testing with the help of Automation tools, which can be completed in two types. Throughout the record part, the test steps are encapsulated by the automation tool. And in the playback, these recorded test steps are implemented on the application under test. **For example: QTP.**

The automation approach is the most common technique in which the GUI automated testing demonstrates itself is beyond record-and-playback methods.



As its name recommends, the record and replay approach depends on having a test engineer use a specific tool to record a testing session. The significant benefits of the Record and replay approach is that it doesn't need any coding skills that reduce the barrier for us to use it. And the major drawback of record-and-replay tests is their **weakness**.

Since the **user interface** is a part of the application that frequently modifies, and rely on the strategy used for interrelating with the elements on the screen.

4. Hybrid Tests

The hybrid tests are the different approach in order to perform GUI testing at the current time. It is a beneficial technique for non-technical background users to develop a test case by recording their sessions. And after that, the user who is familiar with coding can further control these recorded tests technically.

And the person who have the coding knowledge can further manipulate these recorded tests to modify them for more complex situations.

5. Code-based Testing

Another method of performing Graphical user interface testing is Code-based testing. In order to develop test cases by using the code the GUI testing provides some GUI testing tools. To discover more difficult test scenarios, we can use the code-based testing approach.

Subsequently, their code, test cases can be design in source control along with the application's code.

The perceptible disadvantage of the code-based methods that they either involve us to take developers away from coding and have them write test cases or teach our test engineer to code or programming.

Examples of GUI Testing

After describing the Graphical user interface, the description of GUI testing follows quickly. And we can state that it is a method of testing, which is used to validate whether the GUI of a product.

Let see various elements/ objects/ modules, which can be tested under the GUI testing. Essentially, the Graphical User Interface (GUI) testing includes the following:

- The GUI Testing, check the various sections of the screen and analyze the screen in different resolutions.
- It checks whether the interface is attractive or pleasant or not and whether the image has good clarity.
- The GUI testing analysis the headings whether it is correctly aligned or not.
- The GUT testing tests the **spelling, position size, width, height of the features or elements, color of the fonts, the color of hyperlinks, alignment of the images, size of the images, color of the error messages, warning messages,**
- Testing of the scrollbars as per the size of the page, if any, font whether it is transparent/ readable or not.
- It also checks the error messages, which are getting displayed or not, and the disabled fields, if any.

- It also makes sure that the user must not get frustrated while using the system interface.

Different challenges faced during the GUI testing

In **Software testing**, the most frequent drawback or problem while performing Regression Testing is that GUI changes regularly. And it became a very tough situation for a test engineer in order to perform the test and find whether it is a problem or improvement.

The problem displays when we don't have any documents related to the GUI modification. Some of the most frequent challenges during the execution of Graphical User Interface (GUI) testing are as follows:

- **Stability of Objects**
- **Technology Support**
- **Instrumentation**

GUI Testing Tools

In order to identify the bugs or defects, which occurred during the design phase, we will use the **Graphical User Interface (GUI) testing tools** that help us improve the quality of the software.

By using these tools, we can easily detect the loopholes rather than implementing the GUI testing manually.

Based on the application behavior, we will test the application that is involved the **mouse and keyboard actions** and some of the additional GUI items such as **Dialog boxes, buttons, Menu bars, toolbars, and the edit fields**.

Some of the most commonly used GUI testing tools are as follows:

- **Ranorex Studio**
- **Eggplant**
- **Squish**
- **AutoIT**
- **RIATest**

For more information about GUI testing tools, refers to the following link: <https://www.javatpoint.com/gui-testing-tools>.

Conclusion

After seeing all the GUI testing topics thoroughly, we can say that the execution of Graphical User Interface testing is essential. And the accomplishment of the software product relies on how the GUI interrelates with the end-user and helps in using its several attributes.

The execution of Graphical User interface testing is much needed as it ensures the application present looking and works equally in the different platforms and browsers. Consequently, GUI testing is very significant as it will ensure a considerable customer base and business value.

In GUI testing, sometimes executing the Manual GUI testing can be a repetitive and tedious process. However, Automation is highly suggested for the GUI testing process.

Test Strategy

In this section, we are going to understand the **test strategy documentation**, which is an integral part of testing documentation.

And we also learn about **features of test strategy, components of test strategy, types of test strategies, and different testing activities**, which include the test strategy document.

What is Test Strategy?

A high-level document is used to validate the test types or levels to be executed for the product and specify the **Software Development Life Cycle's** testing approach is known as Test strategy document.

Once the test strategy has been written, we cannot modify it, and it is approved by the **Project Manager, development team**.

The test strategy also specifies the following details, which are necessary while we write the test document:

- **What is the other procedure having to be used?**
- **Which module is going to be tested?**
- **Which entry and exit criteria apply?**
- **Which type of testing needs to be implemented?**

In other words, we can say that it is a document, which expresses how we go about testing the product. And the approaches can be created with the help of following aspects:

- **Automation or not**
- **Resource point of view**

We can write the test strategy based on **development design documents**.

The development design document includes the following documents:

- **System design documents:** Primarily, we will use these documents to write the test strategy.
- **Design documents:** These documents are used to specify the software's functionality to be enabled in the upcoming release.
- **Conceptual design documents:** These are the document which we used Infrequently.

Note: A corresponding test strategy can create to test the new feature sets for each stage of development design.

The Objective of Test Strategy

The primary objective of writing the test strategy is to make sure that all purposes are covered entirely and understood by all stakeholders, we should systematically create a test strategy.

Furthermore, a test strategy objective is to support various quality assurance stockholders in respect of **planning of resources, language, test and integration levels, traceability, roles and responsibilities**, etc.

Features of Test Strategy Document

In **SDLC (Software Development Life Cycle)**, the test strategy document plays an important role. It includes various significant aspects, such as who will implement the testing, what will be tested, how it will be succeeded, and what risks and incidents will be are related to it.

Some of the additional characteristics of the Test Strategy document are as follows:

- The test strategy document is approved and reviewed by the following's peoples:
 - **Test Team Lead**
 - **Development Manager**
 - **Quality Analyst Manager**
 - **Product Manager**
- For different testing activities, the test strategy document specifies the resources, scope, plan, methodology, etc.
- In order to direct how testing will be achieved, it is used by the project test team once it is ready or completed.
- Primarily, it is obtained from the **BRS (Business Requirements Specifications)** documents.
- The test strategy document is a high-level document, which generally remains constant, implying no frequent and pointless modification is made in the document.
- The respective team easily accomplishes the objectives of testing with the help of a test strategy document.
- The respective team easily accomplishes the objectives of testing with the help of test strategy document.

Components of Test Strategy Document

We understand that the test strategy document is made during the requirements phase and after the requirements have been listed.

Like other testing documents, the test strategy document also includes various components, such as:



- **Scope and Overview**
- **Testing Methodology**
- **Testing Environment Specifications**
- **Testing Tools**
- **Release Control**
- **Risk Analysis**

- **Review and Approvals**

Let's see them one by one for our better understanding:

1. Scope and Overview

- The first component of the test strategy document is **Scope and Overview**.
- The overview of any product contains the information on who should approve, review and use the document.
- The test strategy document also specified the testing activities and phases that are needed to be approved.

2. Testing Methodology

- The next module in the test strategy document is **Testing methodology**, which is mainly used to specify the **levels of testing, testing procedure, roles, and responsibilities** of all the team members.
- The testing approach also contains the change management process involving the modification request submission, pattern to be used, and activity to manage the request.
- Above all, if the test strategy document is not established appropriately, then it might lead to **errors or mistakes** in the future.

3. Testing Environment Specifications

- Another component of the test strategy document is **Testing Environment Specification**.
- As we already aware of the specification of the **test data** requirements is exceptionally significant. Hence, clear guidelines on how to prepare test data are involved in the testing environment specification of the test strategy document.
- This module specifies the information related to **the number of environments and the setup demanded**.
- The backup and restore strategies are also offered to ensure that there is no data loss because of the coding or programming issues.

4. Testing Tools

- **Testing tools** are another vital component of the test strategy document, as it stipulates the complete information about the **test management** and **automation tools** necessary for test execution activity.;
- For **security, performance, load testing**, the necessary methodologies, and tools are defined by the details of **the open-source or commercial tool** and the number of users that can be kept by it.

5. Release Control

- Another important module of the test strategy document is **Release Control**.
- It is used to ensure that the correct and effective **test execution** and release management strategies should be systematically developed.

6. Risk Analysis

- The next component of the test strategy document is **Risk Analysis**.
- In the test strategy document, all the possible risks are described linked to the project, which can become a problem in test execution.
- Furthermore, for inclining these risks, a clear strategy is also formed in order to make sure that they are undertaking properly.
- We also create a contingency plan if the development team faces these risks in real-time.

7. Review and Approvals

- The last component of the Testing strategy document is **Review and Approval**.
- When all the related testing activities are specified in the test strategy document, it is reviewed by the concerned people like:
 - **System Administration Team**
 - **Project Management Team**
 - **Development Team**
 - **Business Team**
- Together with the **correct date, approver name, comment, and summary** of the reviewed variations should be followed while starting the document.
- Likewise, it should be constantly reviewed and updated with the testing process improvements.

Types of Test Strategies

Here, we are discussing some of the significant types of test strategies document:

Types of Test Strategies



- **Methodical strategy**
- **Reactive strategy**
- **Analytical strategy**
- **Standards compliant or Process compliant strategy**
- **Model-based strategy**
- **Regression averse strategy**
- **Consultative strategy**

Let's understand them one by one in detail:

1. Methodical Strategy

- The first part of test strategy document is **Methodical strategy**.
- In this, the test teams follow a **set of test conditions, pre-defined quality standard**(like ISO25000), **checklists**.
- The Standard checklists is occurred for precise types of testing, such as **security testing**.

2. Reactive Strategy

- The next type of test strategy is known as **Reactive strategy**.

- In this, we can design the test and execute them only after the real software is delivered, Therefore, the **testing is based upon the identified defects** in the existing system.
- Suppose, we have used the **exploratory testing**, and the test approvals are established derived from the existing aspects and performances.
- These test approvals are restructured based on the outcome of the testing which is implemented by the test engineer.

3. Analytical strategy

- Another type of test strategy is **Analytical strategy**, which is used to perform testing based on requirements, and requirements are analyzed to derive the test conditions. And then **tests are designed, implemented, and performed** to encounter those requirements. **For example, risk-based testing or requirements-based testing.**
- Even the outcomes are recorded in terms of **requirements**, such as **requirements tested and passed**.

4. Standards compliant or Process compliant strategy

- In this type of test strategy, the test engineer will follow the **procedures or guidelines created by a panel of industry specialists** or **committee standards** to find test conditions, describe test cases, and put the testing team in place.
- Suppose any project follows the **Scrum****Agile** technique. In that case, the test engineer will generate its complete test strategy, beginning from classifying test criteria, essential test cases, performing tests, report status, etc., around each **user story**.
- Some **good examples** of the standards-compliant process are **Medical systems following US FDA (Food and Drugs Administration) standards**.

5. Model-based strategy

- The next type of test strategy is a **model-based strategy**. The testing team selects the **current or expected situation** and produces a model for it with the following aspects: inputs, **outputs, processes, and possible behavior**.
- And the models are also established based on the current data speeds, software, hardware, infrastructure, etc.

6. Regression averse strategy

- In the regression averse strategy, the test engineer mainly **emphasizes decreasing regression risks** for **functional or non-functional** product shares.
- **For example**, suppose we have one web application to test the regression**issues** for the particular application. The testing team can develop the test automation for both **typical and exceptional use cases** for this scenario.
- And to facilitate the tests can be run whenever the application is reformed, the testing team can use **GUI-based automation tools**.

7. Consultative strategy

- The consultative strategy is used to consult**key investors as input** to **choose the scope** of test conditions as in user-directed testing.
- In order of priority, the client will provide a list of **browsers and their versions, operating systems, a list of connection types, anti-malware software**, and also the contradictory list, which they want to test the application.

- As per the need of the items given in provided lists, the test engineer may use the various testing techniques, such as **equivalence partitioning**

We can combine the two or more strategies as per the needs of the product and organization's requirements. And it is not necessary to use any one of the above listed test strategies for any testing project.

Test strategy selection

The selection of the **test strategy** may depend on the below aspects:

- The selection of test strategy depends on the **Organization type and size**.
- We can select the test strategy based on the **Project requirements**, such as **safety and security** related applications require rigorous strategy.
- We can select the test strategy based on the **Product development model**.

What kind of details may include in the test strategy document?

The final document of the test strategy contains important details about the following factors:

- Scope and Overview
- Re-usability of both software and testing work products.
- Details of different Test levels, relationships between the test levels, and procedure to integrate different test levels.
- Testing environment
- Testing techniques
- Level of automation for testing
- Different testing tools
- Risk Analysis
- For each test level Entry as well exit conditions
- Test results reports
- Degree of independence of each test
- Metrics and measurements to be evaluated during testing
- Confirmation and regression testing
- Managing defects detected
- Managing test tools and infrastructure configuration
- Roles and responsibilities of Test team members

Conclusion

After understanding the **test strategy document**, at last, we can say that the test strategy document provides a vibrant vision of what the test team will do for the whole project.

The test strategy document could prepare only those who have good experience in the **product domain** because the test strategy document will drive the entire team.

And it cannot be modified or changed in the complete project life cycle as it is a static document.

Before any testing activities begin, the Test strategy document can distribute to the entire testing team.

If the test strategy document is written correctly, it will develop a high-quality system and expand the complete testing process.

Software Testing Tools

Software testing tools are required for the betterment of the application or software.

That's why we have so many tools available in the market where some are open-source and paid tools.

The significant difference between open-source and the paid tool is that the open-source tools have limited features, whereas paid tool or commercial tools have no limitation for the features. The selection of tools depends on the user's requirements, whether it is paid or free.

The **software testing** tools can be categorized, depending on the licensing (paid or commercial, open-source), technology usage, type of testing, and so on.

With the help of testing tools, we can improve our software performance, deliver a high-quality product, and reduce the duration of testing, which is spent on manual efforts.

The software testing tools can be divided into the following:

- **Test management tool**
- **Bug tracking tool**
- **Automated testing tool**
- **Performance testing tool**
- **Cross-browser testing tool**
- **Integration testing tool**
- **Unit testing tool**
- **Mobile/android testing tool**
- **GUI testing tool**
- **Security testing tool**

Test management tool

Test management tools are used to keep track of all the testing activity, fast data analysis, manage manual and automation test cases, various environments, and plan and maintain **manual testing** as well.

For more details about test management tool, refers the below link: [Click Here](#)

Bug tracking tool

The defect tracking tool is used to keep track of the bug fixes and ensure the delivery of a quality product. This tool can help us to find the bugs in the testing stage so that we can get the defect-free data in the production server. With the help of these tools, the end-users can allow reporting the bugs and issues directly on their applications.

For more details about bug tracking tool, refers the below link: [Click Here](#)

Automation testing tool

This type of tool is used to enhance the productivity of the product and improve the accuracy. We can reduce the time and cost of the application by writing some test scripts in any programming language.

For more details about automation testing tool, refers the below link: [Click Here](#)

Performance testing tool

Performance or Load testing tools are used to check the load, stability, and scalability of the application. When n-number of the users using the application at the same time, and if the application gets crashed because of the immense load, to get through this type of issue, we need load testing tools.

For more details about load testing tool, refers the below link: [Click Here](#)

Cross-browser testing tool

This type of tool is used when we need to compare a web application in the various web browser platforms. It is an important part when we are developing a project. With the help of these tools, we will ensure the consistent behavior of the application in multiple devices, browsers, and platforms.

For more details about the cross-browser testing tool, refers the below link: [Click Here](#)

Integration testing tool

This type of tool is used to test the interface between modules and find the critical bugs that are happened because of the different modules and ensuring that all the modules are working as per the client requirements.

For more details about the mobile and android testing tool, refers to the below link: [Click Here](#)

Unit testing tool

This testing tool is used to help the programmers to improve their code quality, and with the help of these tools, they can reduce the time of code and the overall cost of the software.

For more details about the unit testing tool, refers the below link: [Click Here](#)

Mobile/android testing tool

We can use this type of tool when we are testing any mobile application. Some of the tools are open-source, and some of the tools are licensed. Each tool has its functionality and features.

For more details about the mobile or android testing tool, refers to the below link: [Click Here](#)

GUI testing tool

GUI testing tool is used to test the User interface of the application because a proper **GUI** (graphical user interface) is always useful to grab the user's attention. These type of tools will help to find the loopholes in the application's design and makes its better.

For more details about GUI testing tool, refers the below link: [Click Here](#)

Security testing tool

The security testing tool is used to ensure the security of the software and check for the security leakage. If any security loophole is there, it could be fixed at the early stage of the product. We need this type of the tool when the software has encoded the security code which is not accessible by the unauthorized users.

For more details about security testing tool, refers the below link: [Click Here](#)

Test Management Tool

Test management tools are used to keep track of all the testing activity, fast data analysis, manage manual and automation test cases, various environments, and plan and maintain manual testing as well.

The test management tool is connected with the automation software. These types of tools had various strategies for testing and multiple sets of features. Some of the test management tools had capabilities to design the test case with the help of requirements.

It is best for test managing, scheduling, defect logging, tracking, and analysis.

Some of the most commonly used test management tools are as follows:

- **Quality center**
- **RTH**
- **Testpad**
- **Test Monitor**
- **PractiTest**



Quality center

The quality center is a test management tool that is launched by **HP**, which is also known as **ALM [Application life cycle management]** tool.

This is very helpful for both test management and **SDLC** because it supports multiple stages of the **software development life cycle**.

It is a web-based testing tool which helps us to control the software from scratch like collection of requirement, planning, designing, testing, and maintenance, but it is a time taking process.

This tool offers integration to other HP products like Load runner and UFT.

Features of Quality center

Following are the commonly used features of HP quality center/ ALM:

- This tool is used to check the test configurations.
- With the help of this tool, we can manage the release and authoring the test and execution.
- This tool helps in project planning and tracking.
- It provides cross-project customization and test resources.
- It is used to manage risk-based quality and lab management.

RTH

RTH is another web-based open-source tool, and it stands for **the Requirement and testing hub**. It is used to manage the requirements, test results, and also have bug tracking facilities. This tool follows a structured approach to extend the visibility of the testing process with the help of a common repository for **test cases**, test result, requirements, and test plans.



Testpad

It is a test plan tool which helps us to identify the **defects or bugs**. It is a combination of simple checklists and spreadsheets. And these spreadsheets are used by the developers for making the notes. While performing **exploratory testing**, **regression testing**, and **Adhoc testing**, this tool is a perfect choice for the test engineer because it provides the keyboard-driven interface and checklist approach.



Features of Testpad tool

- With the help of the Testpad tool, we can create our templates.
- This tool provides secure hosting, secure communication, and reliable data.
- Testpad is mobile and tablet friendly.
- In the test pad, we can upload the screenshots and images.
- We can easily copy and paste the data from the word, excel, or any other test files.
- It has a keyboard-driven editor having JavaScript that is the responsive User interface.
- This tool invites the guest tester if we don't need the accounts.
- This tool is organized with drag 'n' drop and group the checklists into the folder.

For more information, we can refer to the below link:

TestMonitor

It is a powerful test management tool, which helps us to manage a wide range of test cases, milestone, and test runs. With the help of this tool, we can get a real-time understanding of our testing process. It is a test editor, which can run the test case within minutes. For the tester, it offers a simple interface built where the tester can execute the test case at any time, any place with no required experience.



Features of TestMonitor tool

Some of the commonly used features of TestMonitor tool are as follows:

- This tool offers the requirement specification to execute the test case and analysis of the risks.
- This tool will monitor our planning and progress.
- It gives a complete result tracking.
- It is designed to make the testing process more effective, faster, and more structured than excel.
- This tool has a reporting capabilities.

For more information, we can refer to the below link:

PractiTest

This tool is used to increase the end-to-end visibility of our complete testing process and also see the test execution in real-time. With the help of this tool, we can only focus on testing rather than reporting or releasing our product on time.



Features of PractiTest tool

Following are some critical features of PractiTest:

- This tool is used to track the bug and issues by simply e-mailing the system, and it provides the advance filters which help us to stop the duplicate bugs even before they are reported.
- It makes sure the visibility of our project with customized dashboards and reports.
- It helps us to plan our testing process with better time management.
- This tool had a smart filter feature, which helps us to manage our work.
- PractiTest enables easy customization for our projects without writing code.

For more information, we can refer to the below link:

Defect/Bug tracking tool

We have various types of bug tracking tools available in software testing that helps us to track the bug, which is related to the software or the application.

Some of the most commonly used bug tracking tools are as follows:

- **Jira**
- **Bugzilla**
- **BugNet**
- **Redmine**
- **Mantis**
- **Trac**
- **Backlog**



Jira

Jira is one of the most important bug tracking tools. Jira is an open-source tool that is used for bug tracking, project management, and issue tracking in [manual testing](#). Jira includes different features, like reporting, recording, and workflow. In Jira, we can track all kinds of bugs and issues, which are related to the software and generated by the test engineer.



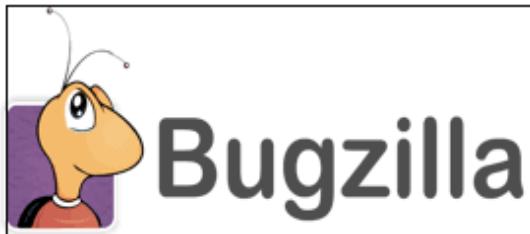
To get the complete details about Jira tool, refer to the below link:

<https://www.javatpoint.com/jira-tutorial>

Bugzilla

Bugzilla is another important bug tracking tool, which is most widely used by many organizations to track the bugs. It is an open-source tool, which is used to help the customer, and the client to maintain the track of the bugs. It is also used as a test management tool because, in this, we can easily link other test case management tools such as ALM, quality Centre, etc.

It supports various operating systems such as Windows, [Linux](#), and [Mac](#).



Features of the Bugzilla tool

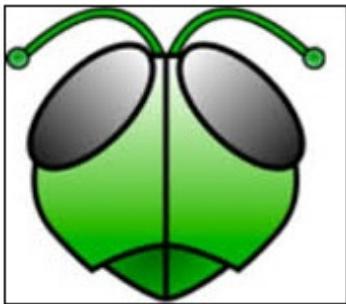
[Bugzilla](#) has some features which help us to report the bug easily:

- A bug can be listed in multiple formats
- Email notification controlled by user preferences.
- It has advanced searching capabilities
- This tool ensures excellent security.
- Time tracking

BugNet

It is an open-source defect tracking and project issue management tool, which was written in [ASP.NET](#) and [C#](#) programming language and support the Microsoft [SQL](#) database. The objective of BugNet is to reduce the complexity of the code that makes the deployment easy.

The advanced version of BugNet is licensed for commercial use.



Features of BugNet tool

The features of BugNet tool are as follows:

- It will provide excellent security with simple navigation and administration.
- BugNet supports various projects and databases.
- With the help of this tool, we can get the email notification.
- This has the capability to manage the Project and milestone.
- This tool has an online support community

Redmine

It is an open-source tool which is used to track the issues and web-based project management tool. Redmine tool is written in [Ruby](#) programming language and also compatible with multiple databases like [MySQL](#), Microsoft SQL, and [SQLite](#).

While using the Redmine tool, users can also manage various projects and related subprojects.



Features of Redmine tool

Some of the common characteristics of Redmine tools are as follows:

- Flexible role-based access control
- Time tracking functionality
- A flexible issue tracking system
- Feeds and email notification
- Multiple languages support (Albanian, Arabic, Dutch, English, Danish and so on)

MantisBT

MantisBT stands for **Mantis Bug Tracker**. It is a web-based bug tracking system, and it is also an open-source tool. MantisBT is used to follow the software defects. It is executed in the PHP programming language.



Features of MantisBT

Some of the standard features are as follows:

- With the help of this tool, we have full-text search accessibility.
- Audit trails of changes made to issues
- It provides the revision control system integration
- Revision control of text fields and notes
- Notifications
- Plug-ins
- Graphing of relationships between issues.

Trac

Another defect/ bug tracking tool is Trac, which is also an open-source web-based tool. It is written in the Python programming language. Trac supports various operating systems such as Windows, Mac, UNIX, Linux, and so on. Trac is helpful in tracking the issues for software development projects.

We can access it through code, view changes, and view history. This tool supports multiple projects, and it includes a wide range of plugins that provide many optional features, which keep the main system simple and easy to use.



Backlog

The backlog is widely used to manage the IT projects and track the bugs. It is mainly built for the development team for reporting the bugs with the complete details of the issues, comments, updates and change of status. It is a project management software.



Features of backlog tool are as follows:

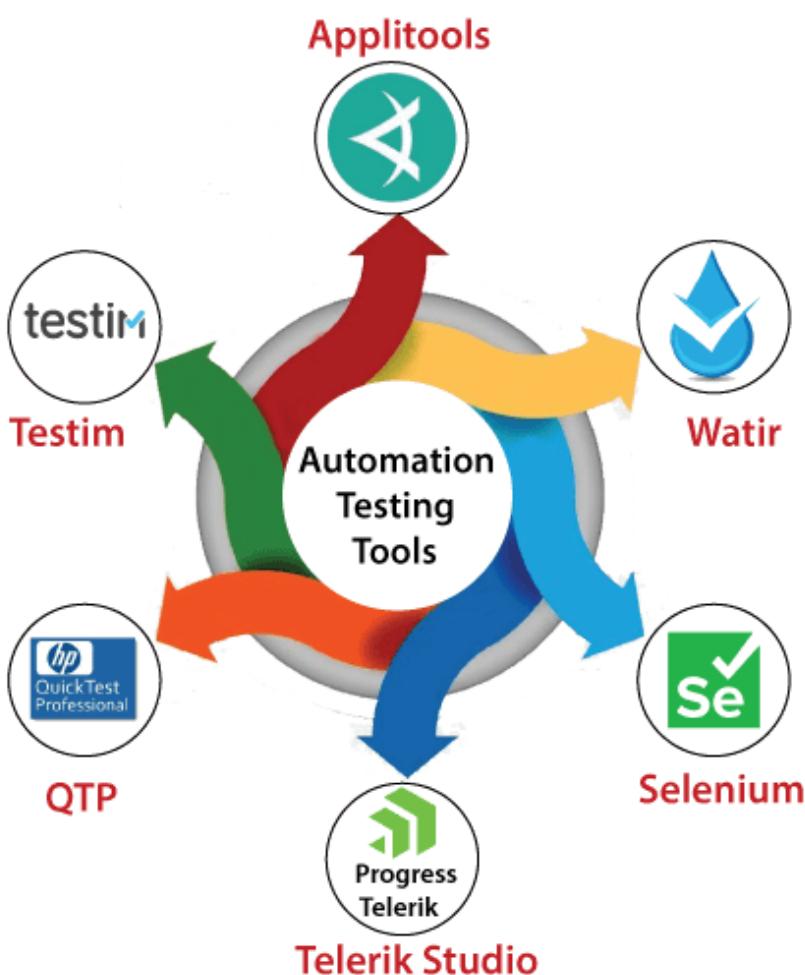
- Gantt and burn down charts
- It supports Git and SVN repositories
- It has the IP access control feature.
- Support Native iOS and Android apps

Automation testing tool

The automation testing is used to change the manual test cases into a test script with the help of some automation tools.

We have various types of automation testing tools available in the market. Some of the most commonly used automation testing tools are as follows:

- **Selenium**
- **Watir**
- **QTP**
- **Telerik Studio**
- **Testim**
- **Applitools**



Selenium

It is an open-source and most commonly used tool in automation testing. This tool is used to test web-based applications with the help of test scripts, and these scripts can be written in any programming language such as java, python, C#, Php, and so on.



Features of Selenium

This tool is most popular because of its various features. Following are standard **features of Selenium**:

- Selenium supports only web-based application, which means that the application can be opened by the browser or the URL like Gmail, Amazon, etc.
- Selenium does not support a stand-alone application, which means that the application is not opened in the browser or URL like Notepad, MS-Word, Calculator, etc.

- Selenium web driver is the latest tool in the selenium community that removes all the drawbacks of the previous tool (selenium-IDE).
- Selenium web-driver is powerful because it supports multiple programming languages, various browsers, and different operating systems and also supports mobile applications like iPhone, Android.

For more information about selenium, refers to the below link: <https://www.javatpoint.com/selenium-tutorial>

Watir

Watir stands for **web application testing in ruby** which is written in the **Ruby** programming language. testing in ruby. It is a web application testing tool, which is open-source and supports cross-browser testing tool and interacts with a platform like a human that can validate the text, click on the links and fill out the forms.



Features of Watir

Following are the characteristics of Watir testing tools:

- It supports various browsers on different platforms like Google Chrome, Opera, Firefox, Internet Explorer, and Safari.
- Watir is a lightweight and powerful tool.
- We can easily download the test file for the UI.
- We can take the screenshots once we are done with the testing, which helps us to keep track of the intermediate testing.
- This tool has some inbuilt libraries, which helps to check the alerts, browser windows, page-performance, etc.

QTP

QTP tool is used to test functional regression test cases of the web-based application. QTP stands for **Quick Test Professional**, and now it is known as **Micro Focus UFT [Unified Functional Testing]**. This is very helpful for the new test engineer because they can understand this tool in a few minutes. QTP is designed on the scripting language like VB script to automate the application.



Features of QTP

Following are the most common features of QTP:

- This tool support record and playback feature.
- QTP uses the scripting language to deploy the objects, and for analysis purposes, it provides test reporting.
- Both technical and non-technical tester can use QTP.

- QTP supports multiple software development environments like Oracle, SAP, JAVA, etc.
- With the help of QTP, we can test both desktop and web-based applications.
- In this tool, we can perform BPT (Business process testing).

Telerik Studio

It is modern web application which support functional test automation. With the help of this tool, we can test the load, performance, and functionality of the web and mobile applications and also identify the cross-browser issues.



Features of Telerik test studio

Below are some essential features of Telerik test studio:

- Telerik test studio allows us to deliver quality products on time.
- This tool supports all types of applications, such as desktop, web, and mobile.
- This tool supports Asp.Net, AJAX, HTML, JavaScript, WPF, and Silverlight application testing.
- It supports multiple browsers like Firefox, Safari, Google Chrome, and Internet Explorer for the test execution process.
- With the help of this tool, we can do the sentence based UI validation.

For more information about Telerik test studio, refer to the below link:

Testim

It is another automation testing tool, which can execute the test case in very little time and run them in various web and mobile applications. This tool will help us to enhance the extensibility and stability of our test suites. It provides the flexibility to cover the functionalities of the platform with the help of **JavaScript** and **HTML**.

A screenshot of the Testim.io website. At the top, there's a navigation bar with icons for back, forward, and search, followed by the URL 'testim.io'. Below the navigation is a pink header bar with the text 'Justifying and Realizing a Strong ROI on Test Automation' and a 'SIGN UP NOW' button. The main content area has a white background. It features the 'testim' logo at the top left. The central part of the page has a large, bold heading 'Tests that deliver'. Below this, two lines of text read 'Super-fast authoring. Amazingly stable tests.' and 'Your way—coded, codeless, or both!'. At the bottom, there are two prominent buttons: a blue 'START FREE TRIAL' button and a white 'SCHEDULE A DEMO' button. Below these buttons, three small text statements are aligned horizontally: '✓ Free 14-day trial', '✓ Easy setup', and '✓ No credit card required'.

Features of Testim

- The Test stability is very high in testim tool.
- This tool will support parallel execution.
- In this tool, we can capture the screenshots.
- This tool will automatically create the tests.
- With the help of this tool, we can perform requirements-based and parameterized testing.

For more information, refer to the below link:

Applitools

This tool is used to check the look and feel and user's feedback on the application. It can easily incorporate with the presented test instead of creating a new analysis. Applitools is a monitoring software, which provides visual application management and AI-powered visual UI testing. It is an open-source tool that helps us to deliver a quality product.



Features of Applitools

Below are the characteristics of Applitools:

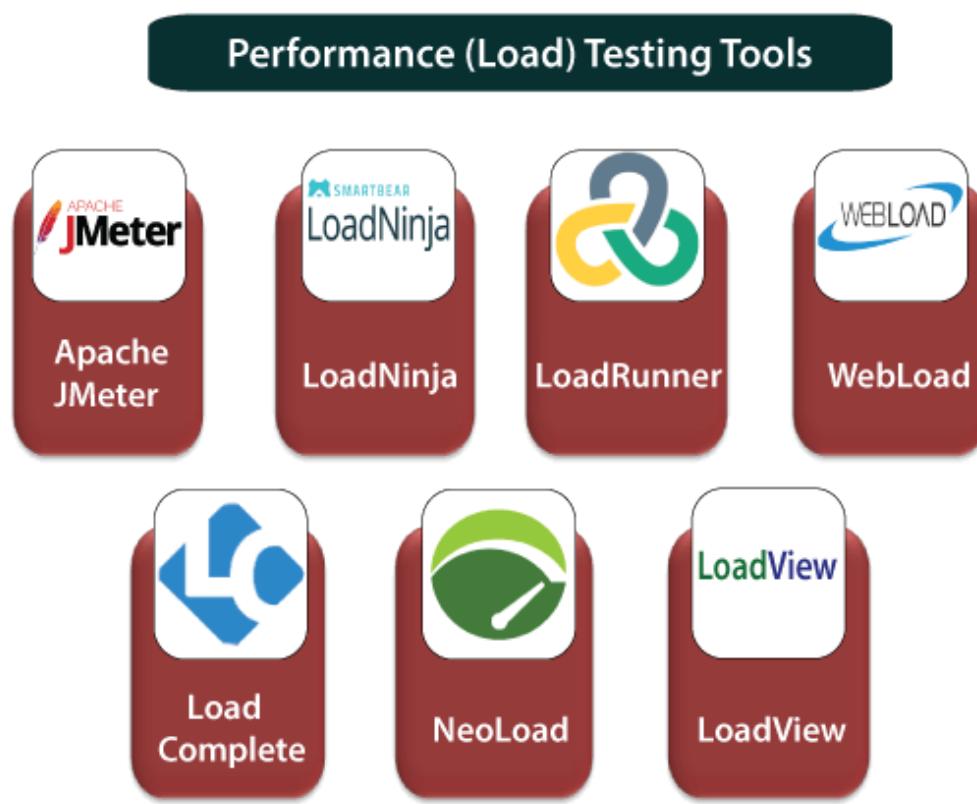
- This tool has active user access management.
- For various devices, it allows cross-browser testing.
- It will deliver the visual test reports to the users.
- It is available on the public and dedicated cloud.

Performance testing tools (Load testing tools)

When we have to measure the load, stability, response time of the application, we required some performance (load) testing tools, which help us to test the performance of the software or an application. Performance testing tools can be open-source and commercial.

We have various types of performance testing tools available in the market; some of the most used performance (load) testing tools are as follows:

- **Apache JMeter**
- **LoadRunner[HP]**
- **LoadNinja**
- **WebLOAD**
- **LoadComplete**
- **NeoLoad**
- **LoadView**



Apache JMeter

It is used to test the performance of both static and dynamic resources and dynamic web applications. This tool is completely designed on the JAVA application to load the functional test behavior and measure the performance of the application. It is an open-source tool that facilitates users or developers to use the source code for the development of other applications.

It can be used to reproduce the huge load on a server, object, or network, group of servers to test its power, or to explore the complete performance in multiple load types. Previously it was used to test the web application, but now it is expanded to other test functions also.



Features of JMeter

Below are some essential elements of JMeter:

- This tool supports a user-friendly GUI, which is interactive and straightforward.
- JMeter maintains multiple testing approaches, such as functional, distributed, and load testing.
- It is incredibly extensible to load the performance test in the multiple types of servers such as database server: LDAP, JMS, JDBC, web server: SOAP, HTTPS, HTTP, and mail server: POP3
- It is a platform-independent because it is designed with the help of JAVA, so it can run on any platform, which accepts a JVM like Window, Mac, and Linux, etc.

For more information about JMeter, refers to the below link:

<https://www.javatpoint.com/jmeter-tutorial>

LoadRunner

It is one of the most powerful tools of performance testing, which is used to support the performance testing for the extensive range of protocols, number of technologies, and application environments.

It quickly identifies the most common causes of performance issues. And also accurately predict the application scalability and capacity.



Feature of LoadRunner

- It will support XML; that's why we can easily view and handle the XML data within the test scripts.
- It supports a large range of applications, which will reduce the time to understand and explain the reports.
- With the help of this tool, we can get detailed performance test reports.
- It will reduce the cost of distributed load testing.
- It will provide the operational tool for deployment tracking.
- This tool is used to reduce the cost of software and hardware.

LoadNinja

LoadNinja is powered by SmartBear. With the help of this tool, product teams and the test engineer will construct the application with more concentration rather than writing the load testing scripts. We can keep track of user interactions, find the performance issues directly, and debug them in real-time. It will change the load emulators with the real browsers.



WebLOAD

WebLOAD testing tool is used to test the test application with the help of load testing, performance testing, and stress testing. For the authentication of web and mobile applications, the WebLOAD tool combines the performance, scalability, and integrity as a single process. It will support the multi-protocols such as HTTPS, XML, HTTP, and so on, which helps us to control the load of the large number of users.



Features of WebLOAD

Following are the most commonly used features of WebLOAD:

- It will provide a flexible test scenario creation.
- This tool detects the bottleneck automatically.
- Customer support can be easily approachable.
- It can evaluate the performance test results from any browser or mobile device.
- It will generate the load from the cloud.

LoadComplete

It is another performance (load) testing tool. It is used to create and run automated tests for web services and web servers. It supports all types of browsers, web services. It will check our web server's performance when we have encountered a huge load. With the help of this tool, we can observe multiple server metrics such as CPU usage, throughout the test runs.



Features of LoadComplete

- It will provide load modeling for performance testing, which means that it allows us to generate a massive load for stress testing.
- With the help of this, we can record and playback our actions in the web browser.
- It supports various platforms like Windows, UNIX.
- During the load testing, it will verify the server message body by taking the help of template-based rules, which make sure that the correct functioning of the server.
- It can test various types of applications like Flash, Flex, Silverlight, and Ajax.
- It will generate the load test reports, which include the customization of the user interface.

NeoLoad

Neotys develop a testing tool which is called NeoLoad. The NeoLoad is used to test the performance test scenarios. With the help of NeoLoad, we can find the bottleneck areas in the web and the mobile app development process.

The NeoLoad testing tool is faster as compared to traditional tools. It will support the complete range of web, mobile, and packaged applications, like SAP, Oracle, Salesforce, and so on, which cover all our testing needs. And also share and manage the test resources.



Features of NeoLoad

Following are some essential features of NeoLoad:

- It will support various frameworks and protocols such as HTTP/2, HTML5, API, AngularJS, Web Socket, SOAP, etc.
- It has a robust code-less design.
- It will change the functional test script into the performance test scripts.
- It will automatically update the test scripts.
- It will generate real-time test results.

LoadView

The **By dotcom-monitor** powers it. With the help of this tool, we can display the real performance of the application. It is used to perform load testing in the real browsers that will give the correct data. It is a cloud-based tool that can be deployed in less time.

A screenshot of the LoadView web interface. At the top, there's a navigation bar with icons for back, forward, refresh, and a star. The main header says "loadview by dotcom-monitor". Below the header, the URL "loadview-testing.com" is shown. The main content area is titled "Stress Testing" and shows a "Load Test Scenario: dotcom-monitor.com". It includes fields for "Load Device" (dotcom-monitor.com), "Task Type" (BrowserView), "Task Name" (https://www.dotcom-monitor.com), "URL" (https://www.dotcom-monitor.com), and "Status" (OK). A "VALIDATE" button is present. There are three curve selection options: "Load Step Curve" (generates loads with a pre-determined number of concurrent users for specified time durations), "Goal-Based Curve" (automatically adjusts concurrent users to reach a required rate of transactions per time interval), and "Dynamic Adjustable Curve" (manually adjust concurrent users in real-time while the test is running). Below these are "Stress Test Scenario Steps" and a "Load Curve" graph showing user count over time. The graph shows a peak around 20:00. At the bottom, there are summary details: "Test Duration: 42 min", "Max Users: 210", "Estimated Resources: -", "Wallet Balance: \$500.00", "Cost for this Test: \$184.95", "Validation Result: OK", and a "RUN TEST" button.

Features of LoadView

- It is used to find the bottlenecks and ensure the scalability of the applications.
- It will perform cloud-based load testing in real browsers.
- With the help of this tool, we can easily build our test scripts.
- It will support various Rich internet applications like Java, PHP, Ruby, HTML5, Flash, Silverlight, and so on.
- It includes global cloud-based testing, point, and click scripting.
- It provides the dedicated Static IPs, which can be configured and allows us to execute our tests for the target resource behind a firewall.

Cross-browser testing tools

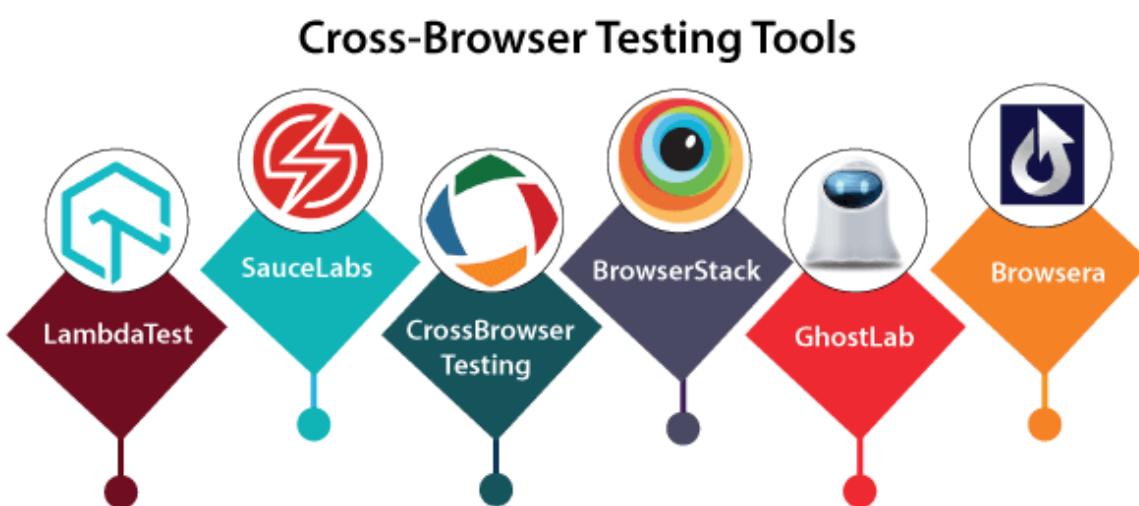
When we have to test our application on multiple browsers, we need cross-browser testing tools. These tools will help us to ensure that our web application is working fine across the various browsers. This tool will take place when both server-side and client-side are accessing the web application in multiple web browsers.

With the help of these tools, we can perform compatibility testing through various browsers for our application. Sometimes, testing a software in a single web browser is not enough; that's why we need the cross browsers testing tools.

We have various cross-browser testing tools available in the market.

Here, we will see some essential tools for cross-browser testing.

- **LambdaTest**
- **Sauce Labs**
- **CrossBrowser Testing**
- **BrowserStack**
- **GhostLab**
- **Browsera**



LambdaTest

It is a cloud-based tool. It uses selenium and Appium test scripts through multiple iOS mobile and android browsers. With the help of the LambdaTest tool, we can test our web application on the latest browsers.



Features of LambdaTest

- LambdaTest tool will provide the localhost web testing to save our web application before the deployment of bugs.
- This tool helps us to debug the issues in live testing.
- With the help of this tool, we can test our application from multiple locations and ensuring that our user gets the perfect experience through all positions.
- It will provide the screenshot feature, which helps us to perform visual cross-browser compatibility testing across multiple mobile and desktop browsers.
- We can verify the responsiveness of our application by just a single click.

- The issue tracker is already integrated with the LambdaTest tool, which helps us to achieve and track our bugs directly from the LambdaTest platforms.

Sauce Labs

It is another cross-browser testing tool, which offers us to execute our tests in the cloud, various browsers, devices, and operating systems. Sauce Labs is a cloud-based testing tool; that's why we don't need the virtual machine set up. With the help of this tool, we can test our application behind the client firewalls because it will provide a secure testing protocol.

It is the first platform, which provides automation testing for the Microsoft Edge browser and supports the Firebug plug-in for the Firefox browser.

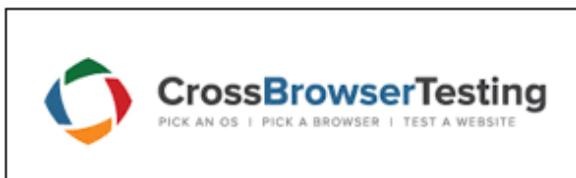


Features of Sauce labs

- It is helpful to increase the productivity of the software because if we are doing constant testing, it will give quick feedback all over the development cycle, which makes them easy and fast debugging.
- It will execute multiple tests such as integration tests, automated end-to-end tests, and unit tests on the Sauce testing cloud.
- It will make sure that our client gets the bug free software.
- Some additional features include extended debugging, test analytics, and sauce performance.
- For automation, it has a clean user interface.

CrossBrowser Testing

CrossBrowser testing is the most famous licensed tool. It supports various operating systems, a large number of multiple browsers, mobile browsers, and their versions. The additional feature includes the automated screenshot, localhost support, and so on.



Features of CrossBrowser testing

Following are the key features of the cross browser testing tool:

- This tool is used to test the application on real mobile devices.
- It is used to verify the public and locally hosted pages across various browsers to check the compatibility of the application.
- It is used to check the test cases step-by-step in the live environment through different devices and multiple browsers.
- It will test our Appium and selenium test script in any programming language.
- It is helpful to execute the screenshot, after running the selenium scripts.

BrowserStack

It is a cloud-based mobile and web testing platform, which empowers the developers to test their web and mobile applications through different operating systems, browsers, and mobile devices. BrowserStack has four main products, such as Live, App Live, Automate, and App Automate.

In this, we don't need to install or maintain any device and the VM [Virtual Machine]. It will help us to reduce the cost, maintenance, and time, and provide stability to structure the right quality product and services.



Features of BrowserStack

- It is used to identify the bugs and fix them directly.
- We can test the application on the extensive collection of browsers like Safari, Google Chrome, Opera, Internet Explorer, Firefox, and so on.
- It will quickly test our layouts and design by creating screenshots on 1500+ desktop and mobile browsers with just a click.
- It is used to test the responsive web design on a variety of screen sizes without trying out each browser combination manually.
- It is highly scalable because it fulfills our testing needs, which help in team growth, after all, the devices are accessible to every member.

GhostLab

GhostLab tool is used to test our application on any website on multiple browsers and mobile devices concurrently. With the help of this tool, we can open our locally installed browsers directly, and to connect a mobile device; we can use the near QR code. It supports us in developing our sites when we are working on a local site. It is available for both Windows and Mac operating systems with no additional setup.



Feature of GhostLab

Some of the vital elements of GhostLab are as follows:

- It will take the screenshot of any connected device, and explain it in the joined editor, and also drag and drop it to our bug tracker.
- It will provide synchronized browsing.
- It will help us to debug the JavaScript with any connected customers.
- It will verify the CSS and DOM on any devices, if any changes happen in DOM, it will automatically change in all connected devices.
- It will restore all our browsers whenever we make any modifications to the local files.
- In a click, we can build and open the various browsers and connect them to the GhostLab.

Browsera

It is an online tool, which is used to test and report the cross-browser design modification and scripting issues on our website. It will compare the output of every browser repeatedly and check the conflicts in our pages, so that we can fix them quickly. It can test the various pages of our websites immediately. It will also create a report once the test has been completed.



Feature of Browsera tool

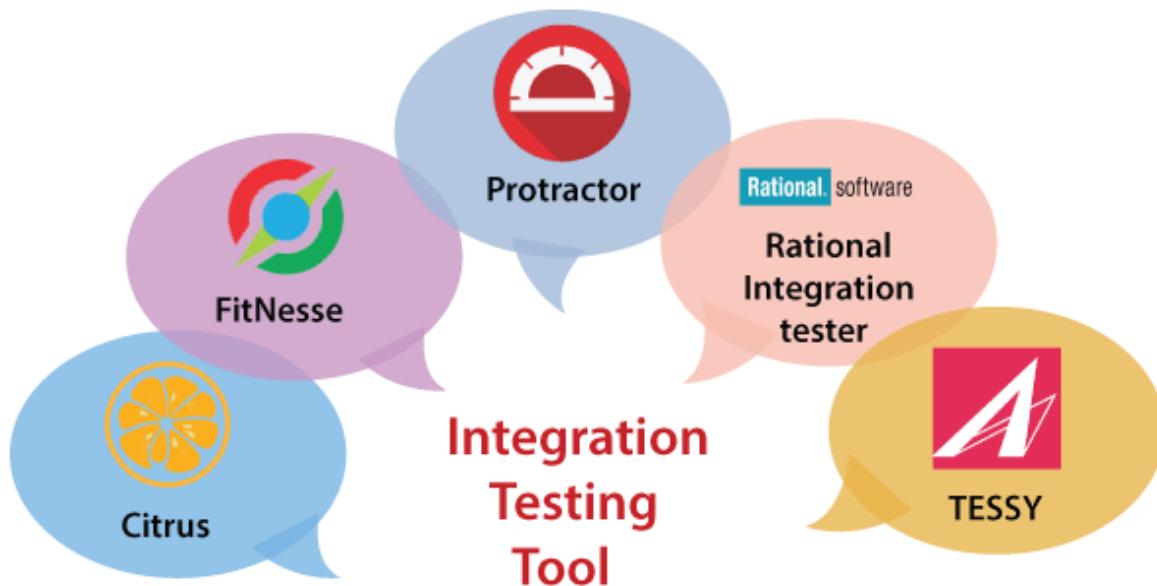
- It is used to locate the JavaScript errors because these types of errors can lead us to the loss of functionality of our site and give a wrong impression to the users.
- It will help us to test the complete websites easily.
- It will automatically detect the cross-browser layout problems.
- It supports both HTTP basic authentication as well as application-based logins. For this, we need to give a unique Id for the login field and access it.
- It tests those websites, which uses AJAX and DHTML technologies. And it will wait until the pages have completed the loading before the testing.
- In this tool, we don't require any installation because everything runs from the server cluster.
- In this tool, we need a web browser to use the service, and we can access the results from anywhere.

Integration testing tools

Integration testing tools are used to test the interface between modules and find the bugs; these bugs may happen because of the multiple modules integration. The main objective of these tools is to make sure that the specific modules are working as per the client's needs. To construct integration testing suites, we will use these tools.

Some of the most used integration testing tools are as follows:

- **Citrus**
- **FitNesse**
- **TESSY**
- **Protractor**
- **Rational Integration tester**



Citrus

It is the most commonly used integration testing tool, which is a test framework and written in the [Java programming language](#). It is used to take the request and respond to both server-side and client-side. It is used to authenticate the [XML](#), [JSON](#) files. It supports multiple protocols such as [HTTP](#), [JMS](#), and [SOAP](#) to achieve end-to-end use case testing.



Feature of Citrus

- It is both an open-source and a licensed tool; that's why it provides a low-cost solution.
- Citrus is used to send and receive messages.
- With the help of this tool, we can validate the database.
- It will describe the sequence of messages.
- Error replication.
- It produces the message and authenticates responses.
- It will use the advance logic in test cases.
- It provides the test plan and documents the test coverage.

FitNesse

It is an open-source tool that does not need separate installation, we need to download the java jar file, and we can use it directly. It is written in Java language and supports another programming language such as [Python](#), [C++](#), [C#](#), [Ruby](#), and so on.

With the help of this tool, we can get a quick response from the users.



Features of FitNesse

- It is used to verify the needs of the actual software application for any software project.
- It is used to run the test and match the real output to the expected output.
- With the help of this tool, we can easily use the wiki web server.
- It also supports an agile style of black-box testing, regression, and acceptance testing.

TESSY

It is an essential tool for integration testing that is used to execute the integration and unit testing for the embedded software. It will take care of the whole test organization along with the requirements, traceability, test management, and the coverage measurement.

TESSY helps us to find the code coverage of an application. With the help of CTE (classification of tree editor), we can design the test cases. And we can edit the test data by using TDE (test data Editor).



Features of TESSY

Following are the standard features of TESSY tool:

- It has the floating license application rights
- The three primary functions of TESSY are TIE (Test interface Editor), TDE (Test Data Editor), and workspace.
- It is used to analyze the interface of the function and defines the variable used by that function.
- TESSY supports C++ and C programming languages.
- For the test execution results, it creates the test report.

Protractor

It is an open-source end-to-end testing framework, which is designed for the [AngularJS](#) and Angular application, and written in JavaScript. It is a [NodeJS](#) program that is used to find the web elements in the [AngularJS](#) applications. Once the application is running in a real browser, it will execute the tests against our application.



Features of Protractor

- With the help of Protractor, we can execute the instance of our application.
- It is used for integration testing.
- From the end-users point of view, it performs the tests.
- It is used to write the end-to-end test.
- It is used for dynamic web applications.

Rational Integration Tester

Previously it is known as a Green hat, but now it is acquired by IBM and known as rational integration tester. It gives the scripting free environment for developing business process integration projects and tests for SOA messaging. It belongs to a rational test workbench.

This tool provides the cost-productive test environments that enable clients to test application early during the development life cycle. With the help of this tool, we can avoid the integration issues by using iterative and agile development process.



Rational. software

Features of Rational Integration Tester

- It will allow us to perform integration, functional, and regression testing.
- If some modules are missing, but the testing will continue, generates code and reusable stubs.
- It will provide the recording facilitating, which upload and export from rational integration tester to the rational test control panel.
- In this, we can build the tests from the requirements and also generate the virtual services to delete the test needs.
- It is used for constant integration in the software life cycle.

Unit testing tools

When we have to find and authenticate the particular module or unit of the code, we need the unit testing tools. With the help of these tools, we can build secure design and documentation and decrease the bug count.

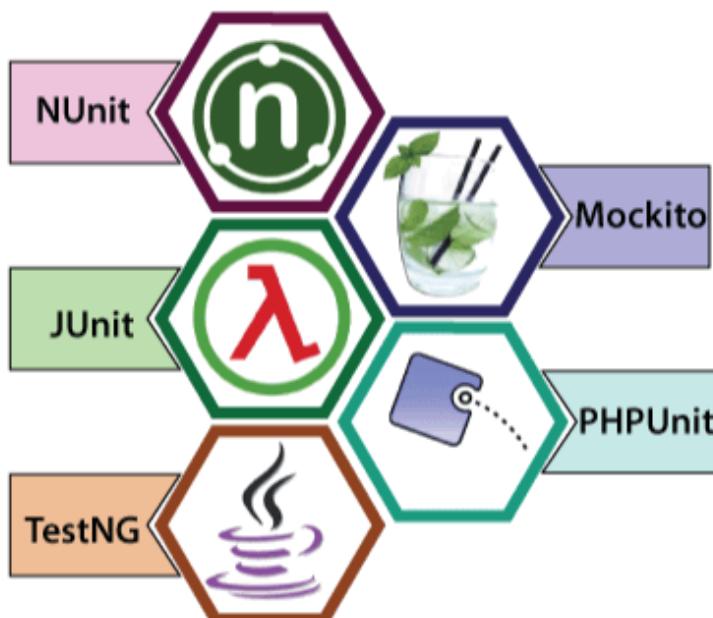
Generally, unit testing is a manual process, but now some of the organization has automated the unit test with the help of these tools. By using unit testing tools, we can cover the maximum coverage, performance, compatibility, and integration testing.

All the unit testing tool is implemented as a plug-in for the eclipse. Unit testing tools are used by the developers to test the source code of the application or to achieve the source code of the application.

Following are the most commonly used tools of **unit testing**:

- **NUnit**
- **JUnit**
- **TestNG**
- **Mockito**
- **PHPUnit**

Unit Testing Tools



NUnit

One of the most commonly used unit testing tools is NUnit. It is an open-source tool and initially ported from the JUnit, which works for all .Net languages. NUnit was written entirely in the **C# language** and fully redesigned to get the advantage of many .Net language features. Like custom attributes and other reflection related capabilities.



Features of NUnit

- It powerfully supports the data-driven tests.
- In this, we can execute the tests parallelly.
- It allows assertions as a static method of the asset class.
- It uses a console runner to load and execute tests.

- NUnit supports various platforms such as Silverlight, Xamarin mobile, .NET core, and compact framework.

JUnit

It is another open-source unit testing framework, which was written in **Java programming language**. It is mainly used in the development of the test-driven environment. Junit offers the annotation, which helps us to find the test method. This tool helps us to enhance the efficiency of the developer, which provides the consistency of the development code and reduces the time of the debugging.



Feature of JUnit

- It offers the assertions for testing expected results.
- In this tool, we can quickly develop a code that enhances the quality of the code.
- This tool can be structured in the test suites, which have the test cases.
- To run the test, it gives the test runners.
- It will take less time to run the test cases.

For more details about the Junit, refers to the below link:

<https://www.javatpoint.com/junit-tutorial>

TestNG

It is an open-source tool, which supports Java and .Net programming languages. **Test Next Generation** (TestNG) is an advance unit testing tool, which is stimulated from JUnit and NUnit testing frameworks. Still, few new functionalities (Additional Annotation, Parallel Execution, Group Execution, Html Report, and Listener) make a TestNG more powerful tool.

For the automation process, TestNG will be used to handle the framework component and achieve the batch execution without any human interference.



Feature of TestNG

Following are the some commonly used **features of TestNG**:

- TestNG support various instance of the same test cases, parameterized, annotation, data-driven, functional, integration, and unit testing.
- In the case of development, TestNG will be used to develop a unit test case, and each unit test case will test the business logic of the source code.
- It will provide a flexible test configuration.
- It will have the Dependent methods for application server testing
- With the help of TestNG, we have full control over the test cases and the execution of the test cases.

- It is supported by multiple plug-in and tools such as IDEA, Eclipse, and Maven, and so on.

For more details about the TestNG tool, refers to the below link:

<https://www.javatpoint.com/testng-tutorial>

Mockito

It is a mocking framework that is used in the unit testing, and it was written in the Java programming language. Mockito is also an open-source tool introduced by the MIT (**Massachusetts Institute of Technology**) License.

With the help of Mockito, we can develop the testable application. The primary objective of using this tool is to simplify the development of a test by mocking external dependencies and use them in the test code. It can be used with other testing frameworks such as TestNG and Junit.



Feature of Mockito

- It will be used to support exceptions.
- With the help of the annotation feature, we can produce the mocks.
- We do not need to write the mocks object on our own.
- It will support return values.
- It provides multiple methods like verify(), mock(), when(), etc., which are helpful to test Java applications.

For more details about the Mockito tool, refers to the below link:

<https://www.javatpoint.com/mockito>

PHPUnit

Another unit testing tool is PHPUnit, which was written in PHP programming language. It is an instance of the xUnit architecture and based on the JUnit framework. It can generate the test results output in many various formats with **JSON**, JUnit XML, TestDox, and Test anything protocol. We can run the test cases on the cross-platform operating system.



Features of PHPUnit

- PHPUnit will provide the logging, code coverage analysis.
- Its development is hosted on GitHub.
- PHPUnit uses assertions to validate the performance of the specific component.
- With the help of this tool, developers can identify the issues in their newly developed code.

Mobile Testing Tools

To test the mobile application, we need these types of tools, which help us to check the usability, functionality, security, and consistency of the application. In the current scenarios, the mobile applications are widely used over the android and iOS platforms, which enhances the client's reliability towards the applications.

So, here we will understand some of the best tools of mobile testing, which are as follows:

- **Appium**
- **Calabash**
- **Testdroid**
- **Kobiton**
- **TestComplete**
- **TestingBot**



Appium

Appium is one of the leading mobile testing tools which is established by the Sauce Labs, and it is an open-source tool. It is used to test the mobile web application, hybrid, and native applications. It supports cross-browser testing, that's why we can execute our application on various platforms like Windows, Mac, Android, iOS, and so on with the help of WebDriver Protocol. The backend of the Appium is Selenium, which provides control over Selenium functionality for our testing needs.



Features of Appium tools

- Appium can control Safari and Chrome on mobile devices.
- It supports various programming languages such as Java, Python, Ruby, and C#.

- It can be combined with multiple frameworks and other tools.
- There is no need for application source code or library.
- It will provide a reliable & active community.
- We can easily set up Appium on different platforms.

Calabash

Another mobile testing tool is Calabash, which is an open-source tool that helps us to test the Android and iOS applications. For the mobile application, we can write and run automated acceptance tests. It is developed and maintained by [Xamarin](#) cloud services.



Features of Calabash tool

- It is used to help in enhancing the productivity of the application.
- It will provide distinct automation libraries for Android and iOS applications.
- It is used to expand the robustness of the product.
- It is used to perform automated functional testing for local mobile applications.

Testdroid

It is a product of Bitbar technologies, which is a set of mobile software development. It is a cloud-based mobile testing tool that is used to save the expenses behind the application progress. It will provide remote [manual testing](#) and [API](#) access to the real devices, which execute Android before an application is introduced.

With the help of the Testdroid, we can easily announce our application, which helps us to reduce the operational costs.



Features of Testdroid tool

Following are the characteristics of Testdroid tool:

- It will help us to decrease random and operational costs.
- It helps us to enhance the application rating based on every day dynamic clients.
- It reduces the risk of agile testing and real devices.
- For the iOS and Android games, it will provide robust mobile gaming testing platforms.

Kobiton

It is a mobile experience platform, which is used to test the mobile application and speed up the delivery of the product. It allows automation and manual testing on real devices. Kobiton will produce the activity logs automatically. It is used to resolve and verify the issues easily because it captures all actions happened while performing testing. It will work for both Android and [iOS applications](#).



Kobiton

Features of Kobiton

- It will access the 100+ real devices.
- It is a highly responsive tool.
- It will provide a parallel execution for manual and automation testing.
- It can collaborate with various other tools like GitHub, Jira, Jenkins, Travis CI, and TeamCity.
- It will Integrate powerful APIs and also support the Appium tool.
- It provides a secure and private connection to our Kobiton cloud.

TestComplete

It is an automated UI testing tool which is established by SmartBear Company. It ensures the delivery of high-quality software that enhance test coverage. It supports multiple platforms such as Windows, iOS, Mac, Android, and so on. It offers us to create, execute, and maintain the test scripts for web, mobile, and desktop applications. This tool gives us full control over mobile device sensor data like GPS, Gyroscopes, and Accelerometers.



Features of TestComplete tool

- It supports various bug tracking tools such as Bugzilla, Jira, etc.
- It will work for android and iOS applications.
- It has built-in keyword-driven test editors, which include the keyword operations that are parallel to the automated testing actions.
- It contains defect tracking templates that can be used to create or modify the items stored in the defect tracking systems.
- It will capture the screenshots while the tests are recorded and playback, and we also get quick evaluations between probable and real screens during the test.
- It will provide real-time information on the progress and status of our web, desktop, or mobile UI tests from a single interface.
- It will support data-driven testing.

TestingBot

It is the primary cloud-based tool for web and mobile applications. With the help of this tool, we can access and debug any browser or device from our computer. The TestingBot users can run Appium, selenium, and JavaScript tests across 1500+ browsers and devices. In this tool, we can also change the internal selenium grid with our cloud-based selenium and the Appium grid. It will increase productivity and quick releases.



Features of TestingBot tool

Following are some standard features of TestingBot:

- We can perform headless testing in the cloud.
- It supports the latest versions of Selenium and Appium.
- With the help of this tool, we can perform live web testing.
- It will be helpful to take the screenshot on all browsers and also compare the results.
- It will provide codeless automation.

GUI testing tools

GUI (Graphical User Interface) testing tool is used to find the defects that happened in the design phase, which enhance the quality of the software. With the help of these tools, we can identify the loopholes quickly rather than performing GUI testing manually. We will test the application based on application performance, which is related to mouse and keyboard actions, and some of the GUI items like buttons, toolbars, Dialog boxes, Menu bars, and the edit fields.

Following are some essential strategies that we can perform under GUI testing:

Navigation validation, verify the check screens, data integrity validation, verification of usability situations, and also check the numeric, date field formats.

Some of the following GUI testing tools are as follows:

- **Eggplant**
- **AutoIT**
- **Ranorex Studio**
- **Squish**
- **RIATEST**

GUI Testing Tools



Eggplant

Eggplant is a GUI test automation tool, which is developed by Test Plant. It is a licensed tool. To execute the end-to-end testing process, eggplant can be integrated into the micro focus quality center, [Jenkins](#), and [IBM](#) rotational quality manager. It will use the two-system model, where the first one contains the controller machine where scripts are written and executed, and another one is SUT (system under test) that runs on the VNC server.



Features of Eggplant tool

- It supports various operating systems such as Windows, Linux, and Mac.
- It includes everything from the most modern highly dynamic websites and the legacy back-office systems to the point of sale and command and control systems.

- It can test any devices such as browser, operation systems, form UI to APIs to the database.

AutoIT

It is a freeware scripting language that is used in the Microsoft windows. It is creating the Graphical user interfaces, which contain the input boxes and message.



Features of AutoIT

- It will execute on the console application and access the standard streams.
- It will provide the Add-on libraries and modules for the specific application.
- It is used to manipulate windows and processes.
- It will contain the data files in the compiled files to be extracted when running.
- It will support the COM (component object model).

Ranorex Studio

It is the most widely used GUI Test Automation tool, which is developed by **Ranorex GmbH**, and it is used to test the mobile, desktop, and web-based applications. It supports the development of the automated test modules, which are written in VB.NET and **C# programming languages**. It will provide cross-browser testing for multiple browsers like Safari, **Chrome**, Firefox, Internet Explorer, and Microsoft Edge.



Features Ranorex Studio

- It can execute on the Windows Server and Microsoft Windows.
- It supports various web technologies like JavaScript, HTML, Flash, Ajax, HTML5, and Silverlight, and so on.
- Ranorex Studio will support native Android and iOS mobile applications.
- It will produce the customize test reports with the video reporting of the test execution.
- It will provide consistent object identification.
- It will generate the reusable code modules, shareable object repository, and also reduce the maintenance cost.

For more information about Ranorex Studio, refers to the below link:

Squish

It is a commercial cross-platform GUI testing tool, which is produced by **Froglogic**, and it is used to test the application based on the diversity of GUI technologies that contains Flex, Android, **JavaFX**, Qt, and so on. It will support various operating systems such as Windows, **Linux**, **Android**, **iOS**, and QNX. With the help of the Squish tool, we can run the sets of scripts and analyze the complete logging and performance results.



Features of Squish

- It will provide test verification and validation.
- It has the power and inbuilt test creation environment.
- Squish had record and playback options.
- It will support behavior-driven development and compatible with the Gherkin language.
- It is used to map and identify the objects which help us to produce stable and robust test scripts.
- It will provide the advanced verification options of elements and groups of controls.

RIATest

It is another GUI testing tool, which is used to test the adobe Flex applications, and it also supports Flex 2, Flex 3, Flex 4, and AIR applications. It is a licensed tool which provides the modified error handling, overthrowing an exception or logging the error. It will highlight the syntax, which makes our test scripts more comfortable to write and readable.



Features of RIATest

Following are some standard characteristics of RIATest tool:

- It will support the Flex, JavaScript, JQuery, HTML applications.
- It will provide the advance automatic and manual synchronization abilities which save our time.
- Its built-in script debugger will help us to identify to resolve the error in our test scripts.
- The action record features are used to record the end-user actions when the application is under test and generate human-readable test scripts.

Security testing tools

Security testing tools are used to make sure that the data is saved and not accessible by any unauthorized user. To protect our application data from the threats, we will use these tools. These tools help us to find the flaws and security leakage of the system in the earlier stage and fix it, and test whether the application has encoded security code or not and accessible by the unauthorized users.

These may initially work on authorization, confidentiality, authentication, and availability types of aspects. With the help of these tools, we can avoid the loss of relevant information, the client's trust, sudden breakdown, additional costs required for repairing websites after an attack, and unpredictable website performance.

For this, we have the following tools available in the market:

- **SonarQube**
- **ZAP**
- **Netsparker**
- **Arachni**
- **IronWASP**



SonarQube

It is an open-source security tool which is established by Sonar Source. It is used to test the quality of the code and execute the automatic reviews with the help of identifying the bugs, code analysis and security exposures on various programming languages such as Java, C#, JavaScript, PHP, Ruby, Cobol, C/C++ and so on of the web applications. SonarQube tool is written on the [JAVA programming language](#).

It will generate the reports of the code coverage, complexity of code, repeated code, security weakness, and bugs. It offers complete analysis with multiple tools like [Ant](#), [Maven](#), [Gradle](#), [Jenkins](#), and so on.



Features of SonarQube

- It will integrate with multiple development environments like Visual Studio, Eclipse, and IntelliJ IDEA over the SonarLint plug-ins.
- It also supports some external tools such as GitHub, LDAP, and Active Directory.

- It can record the metric history and deliver the evolution graphs.
- It will help us to identify the complex issues.
- It will provide application security.

ZAP [Zed Attack Proxy]

It is another security testing tool, which is established by **OWASP**, where it stands for (Open Web Application Security Project). It is an open-source tool that was written on the Java Programming language. If we use this tool as a proxy server, it offers the user to deploy all the traffic which passes over it. We can run this tool on the daemon mode that is exact through the REST API.



Features of ZAP

- It will support the command-line access for advance users.
- It can be used as a scanner.
- It will provide the automatic scanning of the web application.
- It supports different operating systems like Windows, OS X, and Linux.
- It uses the powerful and Old AJAX spiders.

Netsparker

It is used to find the vulnerabilities of the web application uniquely and also validates that the weaknesses of the application are correct or incorrect. It can be easily accessible as Windows software. With the help of this tool, we can do automatic vulnerability assessment and fix the issues and avoid the resources-intensive manual procedures.



Features of Netsparker

- It will automatically scan modern web applications like Web 2.0, HTML5, and SPA (single page applications), and all types of legacy.
- For different purposes, it will provide a multitude of out-of-the-box reports for both developers and management.
- We can generate custom reports with the help of our templates.
- We can collaborate this tool with CI/CD platforms such as Bamboo, Jenkins, or TeamCity to protect our application.

Arachni

It is another open-source security testing tool, which is used to find the security vulnerabilities of the web application. It supports the integrated browser environment, which helps us to identify the security issues of the highly complex web applications.



Features of Arachni

- It will provide vulnerability exposure, test coverage, and correctness of the web application technologies.
- It supports the various platform and all-important Operating systems like Linus, Mac, OS X, and MS Windows.
- It will support different technologies like HTML5, JavaScript, AJAX, and DOM manipulation.

For more information about Arachni, refers to the below link:

IronWASP

It is an open-source tool, which is used to identify the vulnerability of the web application. It stands for the **Iron Web Application Advanced Security Testing Platform**. With the help of this tool, a user can make their custom security scanners. It was developed by using **Python** and **Ruby** programming languages.



Features of IronWASP

- It will support the recording login sequence.
- It will produce the reports for both RTF and HTML formats.
- It is a GUI based tool.
- It will support false Positives and negatives detection.

Penetration Testing Tools

Computer programs used to search for cyber vulnerabilities are penetration testing techniques.

Some specific advantages are given by each application on this list. A simple comparison allows you to decide if the program is the right option for your organization. Let's explore to find the latest choices for security apps on the market.

What is Penetration Testing

Penetration testing, also called as pen testing, ensures that information security experts use security bugs in a computer program to find and take advantage of them. These specialists, often classified as white-hat hackers or **ethical hackers**, make things simpler by detecting attacks by cyber attackers known as black-hat hackers in the modern environment.

In reality, performing penetration testing is equivalent to hiring experienced analysts to conduct a safe facility security breach to figure out how it could be achieved by actual criminals. Businesses and companies are using the results to make the frameworks more stable.

Need of Penetration Testing

Pen testing often demonstrates that where and how the system might be abused by a malicious intruder. This helps you to eliminate any vulnerabilities before a real attack happens.

According to a recent Optimistic Technologies study, every other business has vulnerabilities that can be abused by attackers. Pen testers have been able to violate the company network and enter the network in 93 percent of instances. Four days was the average time taken to do so. An untrained hacker would've been able to access the internal system at 71 percent of the firms.

Working Functionality of Penetration Testing

Firstly, penetration testers need to think about the operating systems that they are going to try to hack. Then, to identify vulnerabilities, they usually use a collection of software tools. Penetration monitoring can also include hacking risks from social engineering. By entrapping a member of the group into having access, testers will attempt to obtain access to a device.

Penetration testers also provide company with the outcomes of their checks, and that is responsible for introducing improvements that either fixes the vulnerabilities or minimize them.

Classification of Penetration Tests

Penetration testing contains the following essential types that are listed below.

- **Blind Tests**
- **White box Tests**
- **External tests**
- **Double-blind tests**
- **Internal Tests**

TYPES OF PENETRATION TESTS

NETWORK PENETRATION TEST

- BLACK BOX
- WHITE BOX
- GRAY BOX



WIRELESS PENETRATION TEST

APPLICATION SECURITY TESTING



PHYSICAL PENETRATION TEST

SOCIAL ENGINEERING

- REMOTE
- PHYSICAL



Let's discuss each one in detail.

Blind Tests

The Companies offers penetration testers with little security details about the device being exploited in a blind test, referred to as a black-box test. The aim is to find vulnerabilities that wouldn't ever be discovered.

White box Tests

A white box test is one where companies offer a range of security details related to their structures to penetration testers to help them improve vulnerabilities.

External Tests

An external test is one where, globally, penetration testers aim to identify vulnerabilities. They are carried out on macro environment-facing software such as domains because of the existence of these kinds of testing.

Double-blind Tests

A double-blind test that is also defined as a covert test is one where sensitive data is not only given to penetration testers by companies. They still may not make the assessments known to their own information security experts. Traditionally, such experiments are strongly regulated by those conducting them.

Internal Tests

An internal examination is one where the examination of penetration exists within the boundaries of an entity. Typically, these checks concentrate on the security weaknesses of which full advantage could be taken by anyone operating from inside an organization.

Best Penetration Testing Tools and Software

1. Wireshark

Typically named as Ethereal 0.2.0, with 600 contributors, [Wireshark](#) is an award-winning network mapper. You can catch and analyze data packets easily with this program. The tool is open-source and is compatible with [Windows](#), Solaris, FreeBSD, and [Linux](#), among other frameworks.

Key Points

- It offers both offline review and options for live-capture.
- Its locating intermediate nodes help you to discover new characteristics, including the protocol of the source and destination.
- It includes the opportunity to inspect the smallest information in a network for operations.
- It contains optional colouring rules for fast, intuitive analysis and are added to the pack.

2. Netsparkar

A common automated application server for penetration testing is the Netsparker vulnerability scanner. From cross-site request to [SQL injection](#), the program can recognize anything from it. This tool can be used by designers on blogs, web infrastructure, and web services.

The platform is efficient enough to simultaneously search anything from 500 to 1000 software applications. With attack tools, verification, and [URL](#) rewriting guidelines, you will have the ability to modify the security scan. In a read-only manner, Netsparker takes advantage of vulnerabilities spots dynamically. Exploitation proof is made. The effect of vulnerabilities can be viewed instantly.

Key Points

- It can search the web-based applications for 1000 + in less than a day!
- For teamwork and easy discoverability of results, you can add several teammates.
- The Advanced scanning reduces the need for a small set up.
- It can search for SQL and XSS bugs in software applications that are hackable.
- You can create the Legal application of the web and reports of regulatory requirements.
- It has Proof-based screening technology to ensure precise identification.

3. BeEF

This is a guide for pen testing and is perfectly suited to a search engine for testing. Applied to fight internet-borne attacks and could help mobile customers. BeEF refers for Platform for Browser Manipulation and uses [GitHub](#) to find problems. BeEF is designed to determine vulnerabilities outside the range of the web client and the network. Rather, than the context of only one reference, the [search engine](#), the framework would look at detectability.

Key Points

- To verify the security infrastructure, you might use client-side known vulnerabilities.
- It helps you to connect with more than one search engine and then begins specific packages with commands.

4. John The Ripper Password Cracker

One of the most common flaws is passwords. To capture information and access sensitive systems, hackers can use credentials. For this reason, John the Ripper is the indispensable tool for password guessing and offers a variety of systems. The pen vulnerability scanner is a free software to use.

- It automatically detects various variations of passwords.
- It also discovers inside databases password vulnerabilities.
- For Linux, Mac OS X, Hash Suite, and Hash Suite Droid, the premium edition is available.
- A personalized cracker is included.
- It helps people to discover online documentation. This provides a description of improvements between variants that are distinct.

5. W3af

In all software development, W3af custom application intrusion and review platforms are based on identifying and exploiting defects. For attack, audit, and discovery, three kinds of plugins are given. Then, the software moves these on to the assessment instrument to evaluate for security flaws.

Key Points

- For professionals, it is easy to use and strong enough, even for developers.

- Automated HTTP request creation and existing HTTP requests can be accomplished.
- It has the ability to configure to function as a proxy for MITM.

6. Metasploit

Metasploit is the world's commonly utilized system for vulnerability assessment optimization. Metasploit allows technical experts to validate and manage safety evaluations, enhance visibility, and arm and inspire defenders to remain in the game a point ahead.

It is helpful to test security, and to find vulnerabilities, to build up a defense. This tool, an open standard program, will enable a system-administrators to crack in and recognize critical flaws. To build up their abilities, novice hackers use this guide. The tool offers social engineers with a way of replicating websites.

Key Points

- It is convenient to use with a scrollable given platform and command-line interface.
- Brute-forcing guides to launch systems to bypass urbanization and modernization, spear spyware, and recognition, an OWASP vulnerability testing app.
- It collects the data from testing for more than 1,500 exploits.
- Meta Modules for experiments of network connectivity.
- This can be used inside infrastructure to discover older vulnerabilities.
- It is also accessible for Mac OS X, Linux and Windows.
- It can be used on servers, software, and channels.

7. Acunetix Scanner

Acunetix is an interactive platform for testing that you will use to execute a penetration test. The technique is capable of evaluating complex management reports and compliance problems. A number of network vulnerabilities can be addressed by the app. Acunetix is also able to include bugs that are out-of-band.

The comprehensive tool combines with WAFs and the widely valued Error detectors. Acunetix is one of the specialized cross-site requests and SQLi testing in the sector, with a high accuracy rate, which includes sophisticated automated XSS detection.

Key Points

- Over 4500 vulnerabilities are protected by the tool, which involves SQL injection and XSS.
- The Login Pattern Recorder is simple to implement and tests sections covered by passwords.
- AcuSensor Technology, Tools for Automatic Penetration, and Built-in order to strengthen and allow abatement, vulnerability assessment streamlines black and white box testing.
- Hundreds of millions of websites can be ransacked without interruption.
- It can operate locally or via a solution in the cloud.

8. Aircrack

Aircrack NG is configured to hack vulnerabilities inside the wireless connections by trapping incoming packets for an efficient protocol to be exported for analysis through word documents. Although the program seemed to have been discontinued in 2010, in 2019, Aircrack has modified again.

This tool is enabled on multiple OS and WEP dictionary-based attack support frameworks. Especially in comparison to many other penetration tools, it provides an increased detection performance and supports several devices and drivers. The suite is responsible for using a credential dictionary and mathematical techniques to crack into WEP after obtaining the WPA handshake.

Key Points

- It is compatible with Solaris, Linux, Windows, OS X, FreeBSD, NetBSD, and OpenBSD.
- To retrieve packages and export data, you will use this method.
- It is intended for wi-fi system testing as well as driver proficiency.
- It focuses on various security fields, such as an attack, surveillance, testing, and cracking.
- In terms of intrusion, you can de-authenticate, establish a fake wireless network and replay attacks.

9. Burp Suite Pen Tester

The Burp Suite for programmers has two separate editions. The free version offers appropriate and essential tool for testing operations that are needed. Or, when you need extensive penetration testing, you can go for the second version. For testing web-based applications, this tool is perfect. Tools for mapping the tack substrate and analyzing transactions between the browser and endpoint servers are available.

Key Points

- It is suitable for web-based software scrolling automatically.
- Mac OS X, Linux, and Windows are accessible in this tool.

10. Kali Linux

A Linux operating system used for vulnerability assessments is [Kali Linux](#) Specialized Penetration Testing Program. This is the perfect instrument for both extracting and password snipping, many analysts claim. However, to achieve the most of the advantages, you might need experience in both [TCP / IP](#) protocols. Tool descriptions, edition management, and meta-packages are supported by an open-source project, Kali Linux.

Key Points

- You will use this technique for brute-force attack password cracking with 64-bit assistance.
- To evaluate the security skills of cybersecurity professionals, Kali uses a live image configured into the RAM.
- Kali Linux contains 600 hacking methods that are ethical.
- Multiple vulnerability assessment security tools, web applications, data collection, wi-fi attacks, reverse engineering, cracking passwords, forensic tools, web services, imitating, sniffing, trafficking tools, and hacking device are obtainable.
- It has fast compatibility with other tools, include Wireshark and Metasploit, for penetration testing.
- BackTrack offers testing, forensic analysis, and sniffing tools for Wireless Lan and LAN vulnerability analysis.

11. Ettercap

The Ettercap software is intended to avoid attacks by a person in the Centre. You will be able to generate the packages you need and perform certain tasks using this tool. The program can submit illegitimate frames and entire technologies via other options that are more complicated.

Key Points

- This method is suitable for deep sniffing of packets as well as LAN tracking and checking.
- Passive and active defense deconstruction is provided by Ettercap.
- The internal control system can be done on the fly.
- For both channel and client evaluation, the tool also offers configurations.

12. Nessus

Over twenty years, Nessus is often used as a privacy vulnerability scanning tool. Twenty-seven thousand leading organizations use the program. Including over 45,000 CEs and 100,000 extensions, the app is among the strongest testing tools and services. Optimal for analyzing IP addresses, domains and conducting critical searches for data. You will be eligible to use this in the frameworks to identify 'weak spots.'

This tool is easy to use and provides reliable testing, and presents an analysis of the vulnerabilities of the network at the click of the mouse. Searches for known vulnerabilities, poor passwords, and data corruption errors by the pen test program.

Key Points

- This tool is optimal for the location and detection of faulty patches and vulnerabilities.
- Per 1 million tests, the device has just .32 errors.
- This tool helps you to customize files containing vulnerability forms, can be generated by the plugin or server.
- The tool provides preference mitigation in addition to make sure software, device scanning, and cloud infrastructure.

13. Zed Attack Proxy

OWASP ZAP is a tool of OWASP group. It is suitable for architects and programmers that are acquainted with penetration testing. The campaign began in 2010 and is being enhanced every day. ZAP executes in a multi-platform strategy that supports a proxy between the domain and the server.

Key Points

- Four modes are available with customizable options.
- JAVA 8 + is necessary for the Windows or Linux platform to access ZAP.
- Getting Started (PDF), Introduction, online help, professional associations, and StackOverflow are extensively supported pages.
- People via program code, Wiki, Developer Community, Crowdin, OpenHub, and BountySource will explore more about Zap growth.

14. SQLmap

SQLmap is a Database SQL Injection Control Tool. It also enable MySQL, SQLite, Sybase, DB2, Access, MSSQL, PostgreSQL database platforms. SQLmap is open-source and streamlines the mechanism of manipulating the application server and bugs for the Attack vector.

Key Points

- This tool allows you to Detect exploits and monitor them.
- It offers assistance for all aspects of injection: Union, Time, Stack, Error, Boolean.
- It executes a command-line interface and can be configured for Linux, Mac OS, and Windows operating systems.

15. Cain and Abel

Cain & Abel is suitable for penetration for the acquisition of network controls and credentials. To detect the vulnerability, the tool makes the utilization of network sniffing.

Key Points

- Utilizing the network sniffers, cryptographic algorithms threats, and brute force, the Windows-based framework can restore passwords.

- It is Superb for missing password restoration.

16. Wapiti

Wapiti is a security tool for programs that enables **black-box testing**. Checking the black box tests web-based applications for possible exposures. Websites are checked at the time of the black box testing procedure, and the tested data is implanted to search for any failures in protection.

Key Points

- With the help of command-line application interface, professionals may find ease-of-usability.
- Wapiti detects file exposure glitches, XSS Intrusion, Database transfusion, XXE injection, Command Execution mitigation, and vulnerable .htaccess settings that are easily evaded.

17. Social Engineering Tool (SET)

The key objective of the tools and techniques is social engineering. Living beings are not the aim of penetration testing beyond purpose and emphasis.

Key Points

- This tool presented at prominent cybersecurity events, include ShmooCon, Defcon, DerbyCon, and is a vulnerability assessment industry norm.
- There have been over 2 million downloads of Package.
- An open-source test system is developed to prevent social engineering.

18. Hydra

When you need to break an encrypted password, including an **SSH** or **FTP** account, **IMAP**, **IRC**, **RDP** and several others, John the Ripper's accomplice, Hydra, plays a role. Place Hydra at the system that you'd like to hack since you like, pass a set of words, and squeeze the trigger. Services like Hydra are a demonstration of why, during a couple of password attempts, rate-limiting password tries and unplugging users can be efficient preventive measures toward the attackers.

19. Hashcat

The self-proclaimed "the quickest and most sophisticated password protection tool in the world" may not always be humble, but the hashcat folks sure know its value. John the Ripper is offered a challenge for his money by Hashcat. It is the perfect solution-to pen- testing method for cracking hashes, and hashcat enables several forms of brute force attacks through password cracking, like a dictionary and disguise attacks.

Pentesting typically entails hashed credentials being exfiltrated, and manipulating those passwords involves turning a software such hashcat unleash on them offline in the expectation of at least a few passwords being guessed or brute-forced.

On a standard **GPU**, Hashcat performs perfectly (sorry, Kali VM users). Legacy hashcat also facilitates **CPU** hash hacking, but it warns users that it is marginally slower than integrating the computational capabilities of the graphics card.

Advantages of Penetration Testing Tools

Here, some advantages of pen-testing tools are defined below.

1. Arrangement and Detection of Security Threats

A penetration test calculates the capability of the organization to secure their apps, servers, users and data sources from international and domestic attempts to avoid its security measures in order to obtain restricted or unsanctioned access to secured properties. The pen test result confirms the danger posed by specific security

problems or defective systems, enabling abatement initiatives to be arranged by IT management and intelligence analysts.

2. Subvert the channel failure intensity

Recovering from a security flaw is costly. IT rescue station, retention measures, consumer security, commercial-grade, reduced sales, decreased employee productivity, and frustrated trade representatives can be included in recovery. Penetration testing helps a company to prevent these financial difficulties by recognizing and resolving risks constructively before data breaches or attacks occur.

3. Meet the needs of tracking and mitigate penalties

The ultimate monitoring/implementation aspects of activities such as HIPAA, SARBANES-OXLEY, and GLBA are discussed by IT agencies, as well as the monitoring needs to be acknowledged in the federal NIST / FISMA and PCI-DSS directions. The detailed reports provided by the vulnerability scanners will help organizations escape significant non-adherence consequences and allow them to demonstrate continuing due diligence in evaluators by retaining the appropriate safety controls for auditors.

4. Service delays and security problems are costly

Security vulnerabilities and the corresponding performance disturbances in service providers can result in crippling economic harm, damage the credibility of an enterprise, decimate customer loyalties, elicit negative attention, and impose unexpected financial penalties. Frequent recruitment in penetration testing by the company prevents these expenses.

Checking penetration enables the company to prevent invaders of the infrastructure. It is safer for the company to protect its protection promptly than to suffer drastic failures, both in terms of its brand value and its financial stability.

5. Secure brand recognition and corporate image

Only a single instance of stolen consumer data may kill the reputation of a business and affect its end result negatively. Penetration testing can help an entity eliminate data accidents that can place the integrity and reliability of the business at risk.

Automation Testing vs. Manual Testing

Automation Testing

Automation testing is a process of changing any manual test case into the test scripts by using automation testing tools, and scripting or programming language is called automation.

Automation testing is used to increase the efficiency, effectiveness, and coverage of Software testing.

Automation test engineer uses automation testing tools to automate the manual design test cases without any human interference.

And these testing tools can control the execution of tests, access the test data, and compares the actual result against the expected result.

Manual Testing

Manual testing is testing, where the tester can test the application without any knowledge of any programming language.

In manual testing, the test engineer tests the application like a user to make it bug-free or stable.

Manual test engineers always search for the fault or bugs in the product before the product released in the market, yet delivered software still has defects.

And there is a chance that the final software product still has a defect or does not meet the customer requirement, even the manual test engineer do their best.

Difference between automation testing and manual testing

Aspects	Automation testing	Manual testing
Definition	When an application or software is tested with the help of some tools is known as automation testing. Whenever multiple releases or multiple regression cycle is going on the application or software, we will go for automation testing.	It is a type of software testing, which is done by the test Engineer to check the functionality of an application based on the customer requirement.
Reliability	It is reliable because it tests the application with the help of tools and test scripts.	It is not reliable because there is a possibility of human error, which may not be delivered the bug-free application.
Reused	The script can be reused across multiple releases.	It could be possible when the test case only needs to run once or twice.
Batch Execution	Batch execution is possible using automation testing because all the written scripts can be executed parallelly or simultaneously.	Batch execution is not possible in manual testing.
Time-saving	The execution is always faster than the manual; that's why the automation testing process is time-saving.	It is time consuming due to the usage of the human resources.
Investment	While using the Automation tool, investment is required.	Human resources needed investment.
Performance testing	To test the performance of the application with the help of load and stress testing, automation test engineer needs to perform Performance Testing.	In manual testing, performance testing is not possible.

Programming knowledge	Without having an understanding of programming language, we cannot write the test script.	There is no need to know programming language but should have the product knowledge to write the test case.
Framework	The automation test engineer can use the different types of frameworks like Data driven, Hybrid, modular driven, and keyword-driven to faster the automation process.	There is no need for a framework while using manual testing.
Operating system compatibility	Automation testing can also be performed on different systems with different operating system platforms and various programming languages.	Operating system compatibility is not possible in manual testing because the different tester is required to perform such tasks.
Regression testing	Whenever the code changes happen due to the enhancement of the release, then automation test engineer performs the regression testing.	When the test engineer executes the test case for the first time, it may be useful, but there is a possibility that it will not catch the regression bugs because of changing requirements frequently.

Load Testing vs. Stress Testing

Load testing

The load testing is the most important essential part of performance testing which is used to check the performance of an application by applying some load like less than or equal to the desired load is known as load testing. And the load is a quantity, which means it only focuses on the numbers of users.

Stress testing

The stress testing is testing, which checks the behavior of an application by applying load greater than the desired load. Since it is non-functional testing, so we use this testing when the application is functionally stable.

Difference between Load testing and Stress testing

Load testing	Stress testing
Load testing is used to find the performance of the application by testing the database, networks, and website servers.	Stress testing is used to find the stability and response time of the given system.
Load testing helps the tester to identify the bottleneck and also able to tell the cause of bottlenecks before deployment to the production server.	Stress testing helps the tester to check the system capacity when number of users increased suddenly before the system failure or crashed.
This type of testing reproduced the load on any application and software.	It is used to figure out the robustness and stability of the application.
Load testing is used to test web-based and client-server types of application.	Stress testing tests suddenly increased traffic of the application.
WebLOAD, LoadView, LoadRunner, SmartMeter.io, and LoadUI NG Pro are some of the load testing tools which help us to perform load testing on the applications.	LoadRunner, JMeter, NeoLoad are some of the stress testing tools which help us to perform stress testing on the applications.
After doing the load testing on the application, the cost of failure may reduce, and the satisfaction of the customer is increased.	After doing Stress testing on the application, if the system got to fail, it will recover quickly by finding the breaking point early and also see where the system got crashed.
For example: if we have one scenario where the load is 100 users using the application at a 2.5\sec of goal time. And, the desired load is 100 user. This scenario got passed because the desired load is equal to the load, which satisfies the load testing condition.	For example: if we took the same scenario where the actual load of 100 users which are using the application at a 2.5\sec of goal time. When the desired load got increased by 200 users, and it will become 300. So now, 300 users using the application at 2.5\sec of goal time. It will pass because the desired load is greater than the load according to the stress testing condition.

Differences Between Smoke Testing and Sanity Testing

In this section, we are going to discuss what is **smoke and sanity testing** in **Software Testing**; and see the major differences between them.

In software testing, we will understand that any testing done at the initial phase of the **SDLC (Software Development Life Cycle)** is determined to be a cost and time-saving process.

Smoke Testing and **Sanity Testing** are a few of the significant types of **testing**, making sure that bugs and defects are exposed in the development cycle's initial phases.

Usually, we all get muddle between the definition of **Sanity Testing and Smoke Testing**. Firstly, both testing is way **different** and executed throughout the different phases of a testing cycle.

Before understanding the smoke and sanity testing, it is suggested to understand **Build** and **Release** and how these terms are related to the Smoke and Sanity Testing.

What is Build?

In manual testing, build is software that contains some set of features/bugs and it is installed on a **test server** that needs to be tested for the product's stability.

In other words, we can say that the build is used to change the code into the application format. Every new build will be the improved version of the new build.

For more information about the software build and build process, refer to the following link: <https://www.javatpoint.com/manual-testing>.

What is Release?

In **software testing**, it is very common terminology used on an everyday basis. The **Release** is a final product or a project, which is delivered to the customer.

It involves the complete activities from the **Requirement, Designing, Development, and Testing phases** until it is handed over to the customer.

In other words, we can say that a release is an entirely developed application, while the build is the part of an application or the software.

Note: Whenever a build is tested and specialized by the testing team, it is handed over to the clients as Release.

Now, let's see a brief introduction of **Smoke Testing and Sanity Testing** procedures.

What is Smoke Testing?

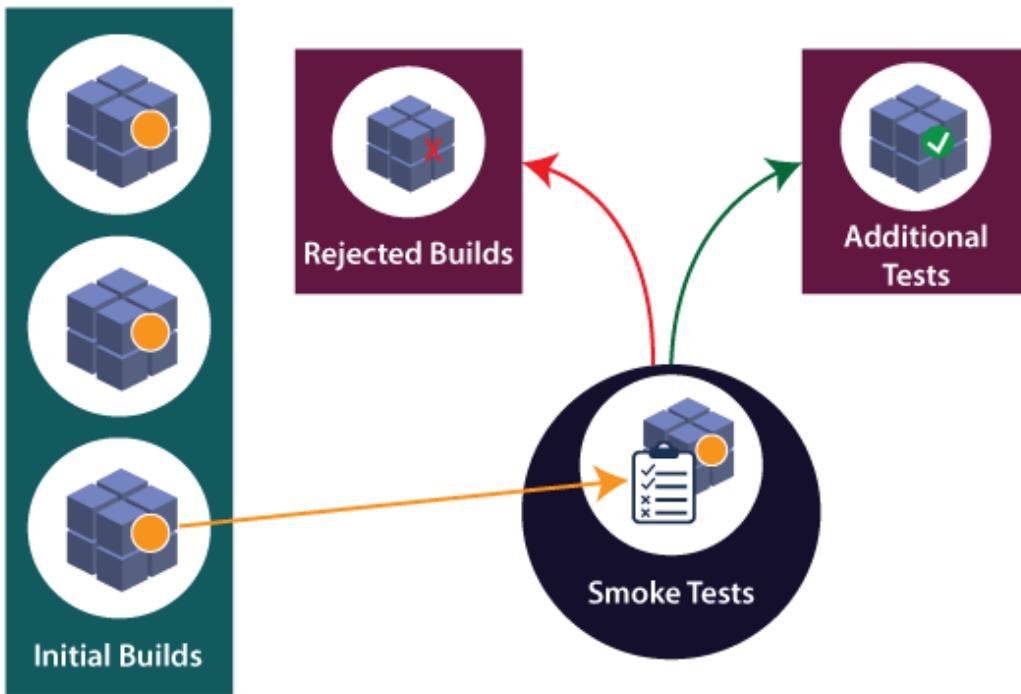
It is a type of testing that guarantees an application's basic and critical features are working fine before doing exhaustive testing or rigorous testing.

Smoke testing is also known as a subcategory of **acceptance testing** or **Build Verification testing**.

In other words, we can say that smoke testing is used to test all the functionality of the software product **or** check whether the build is broken or not.

In smoke testing, we only perform positive testing, which implies that we can enter only the valid data not invalid data.

Whenever a new build comes in, we always start with smoke testing because some changes might have broken a major feature of a new build.

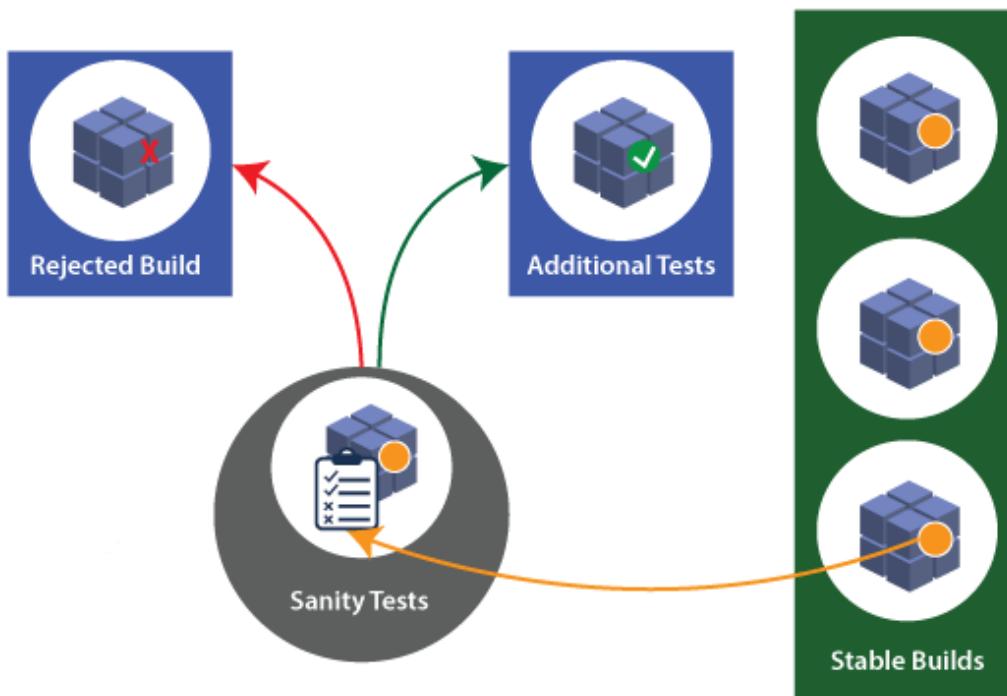


For in-depth information about smoke testing, refers to the following link: <https://www.javatpoint.com/smoke-testing>.

What is Sanity Testing?

It is performed to check whether the bugs have been fixed after the build. Generally, Sanity testing is performed on stable builds. It is also known as a variant of regression testing.

The initial aim of performing sanity testing is to determine that the planned features work roughly as expected. If the sanity test fails, the build is rejected to save the costs and time complex in more severe testing.



For more details about sanity testing, refers to the following link: <https://www.javatpoint.com/sanity-testing>.

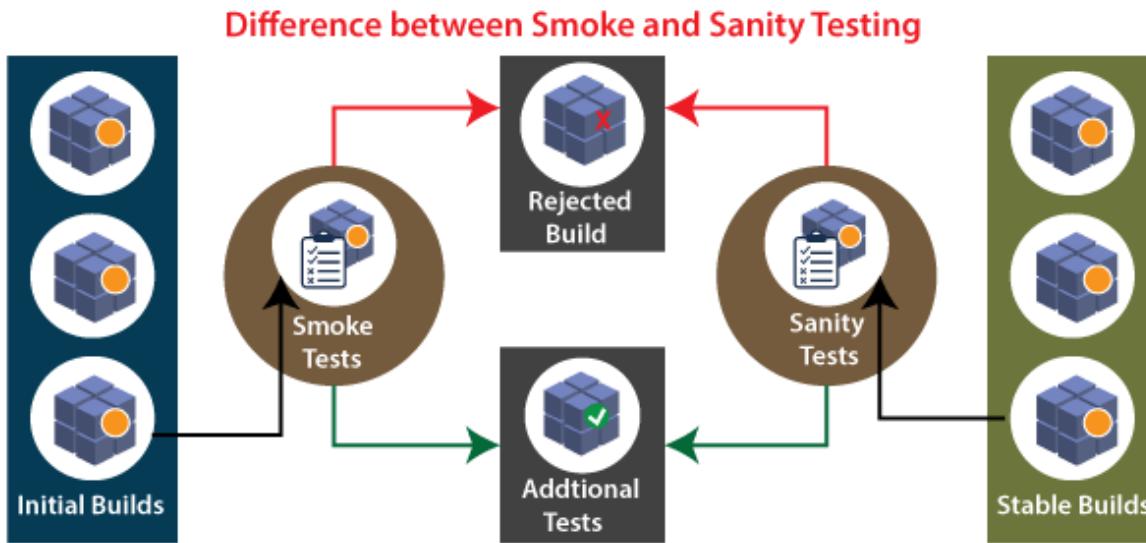
Note: A Dry Run is a testing process where the consequences of a possible failure are purposefully mitigated.

Whenever we talk about **smoke and sanity testing**, we know that these two types of testing are similar, but both smoke testing and sanity testing have their purposes and significance.

Therefore, in this article, we try to overcome the Smoke and Sanity testing's confusion by seeing the key difference between them.

The key difference between Smoke Testing and Sanity Testing

The below facts explain the differences between **smoke** and **sanity** testing:



- Smoke Testing is **scripted**, which means it can be documented, whereas the sanity testing is **unscripted**, which implies that it cannot be documented.
- Smoke testing is considered **shallow and wide** testing, and on the other hand, sanity testing is considered **narrow and deep** testing.
- Smoke testing takes **all important features and performs high-level testing**, whereas sanity testing takes **some very significant features and performs deep testing**.
- Smoke testing is executed **as soon as the build is installed**, and on the other hand, sanity testing is implemented **as soon as the bug fixes are done**.

Smoke Testing vs Sanity Testing

The below comparison table enlightens the important differences between smoke testing and sanity testing in a quick manner:

S.No.	Comparison Basis	Smoke Testing	Sanity Testing
1	Test coverage	It is a broad approach to testing where all parts of the application are tested.	It is a narrow approach to testing where specific parts of the application are tested.
2	Measures	It measures the stability of the system by performing rigorous testing.	It measures the rationality of the system by performing rigorous testing.
3	Technique	Smoke testing can be either manual or automated.	Sanity testing can be done without test cases or scripts.
4	Executed by	It is performed by both testers and developers.	It is performed by only testers.
5	Purpose	Testing is done without getting into deep but whenever needed tester has to go into deep.	Sanity testing does not need to go into deep of the application.
6.	Performed at	Smoke testing is the first testing performed on the initial build.	Sanity testing is performed when the build is comparatively stable.
7	Documentation	Smoke testing is documented.	Sanity testing is not documented.

8	Used to	It is used to test End to End function of the application.	It is used to test only modified or defect fixed functions.
9	Subset	It is considered as a subset of acceptance testing.	It is considered as a subset of regression testing.

Conclusion

In this tutorial, we have made a comparison between **smoke testing and sanity testing**. Here we have concluded that the execution of smoke and sanity testing is required to test the software or the application.

These are the basic testing concepts that is followed by most of the **quality assurance and projects teams** in various software development projects.

The beginner in testing needs to know Smoke and Sanity testing's fundamentals for effective and good Quality Assurance outcomes.

Both smoke testing and sanity testing can either be implemented manually or with some automation tools. When automation tools are used, tests are started to create the build repeatedly.

As per the software's need, we can perform smoke or sanity testing in a similar software build. In a situation like this, we will first implement the **Smoke tests** and then proceed with **Sanity Tests**.

In the software industry, test cases for Sanity Testing are usually shared with smoke tests to accelerate the test execution process.

Both Sanity and Smoke testing are different ways to avoid delay and energy by quickly defining whether an application is too damaged to excellence any rigorous testing.

Difference Between System Testing and Acceptance Testing

In this section, we are going to discuss what is **system testing and acceptance testing** in **Software Testing**; and see the significant differences between them.

The **system and acceptance** testing both are methods of dynamic testing used to verify all the built-in requirements in the software or the application.

Based on testing performance, we can categorize the testing into two subcategories: **System Testing and Acceptance Testing**.

As we already knew, the **SDLC (Software Testing Life Cycle)** includes multiple stages, where the application testing phase involves the process of System testing and Acceptance testing.

Firstly, we will execute the system testing and then the acceptance testing on the application before the software product's beta and alpha releases.

Now, let's see a brief introduction of **System Testing and Acceptance Testing** approaches.

What is System testing?

In **functional testing**, system testing is used to analyze the end-to-end flow of an application.

We navigate all the required modules of an application, analysis if the end features work fine, and test the product as an entire system.

In system testing, the test environment is parallel to the production environment; that's why it is also known as **end-to-end testing**. And it contains the **SIT (System Integration Testing)** and the **ST (System Testing)**.

For more details about system testing, refers to the following link: <https://www.javatpoint.com/system-testing>.

What is Acceptance Testing?

Acceptance testing or User acceptance testing (UAT) is another classification of software testing performed by the **customer** before accepting the final product.

Generally, acceptance testing is used to evaluate whether the application works as per the specified business requirements or real-time scenarios.

UAT testing is executed on the customer's unique environment, which is acknowledged as the **UAT environment**.

For more details about acceptance testing, refers to the following link: <https://www.javatpoint.com/acceptance-testing>.

System Testing VS. Acceptance Testing

Let's see the key difference between **system testing** and **acceptance testing**:



System Testing



Acceptance Testing

S.no.	Comparison basis	System Testing	Acceptance Testing
1	Definition	System testing is performed to test end to end functionality of the software.	Acceptance testing is performed to test whether the software is conforming specified requirements and user requirements or not.
2	Executed by	Only developers and testers can perform System testing.	It can be performed by testers, stakeholders and costumers.
3	Part of	It can be both non-functional and functional testing.	It can be only functional testing.
4	Analysis	In System testing, we test the performance of the whole system.	In Acceptance testing, we test whether the system is conforming requirements or not.
5	Inputs	System testing uses demo input values that are selected by the testing team.	Acceptance testing uses the actual real-time input values provided by the user.
6	Test coverage	In this testing, we include the testing of complete specification including software and hardware, memory and number of users.	Here we test whether the software is fulfilling all the needs of the user or not.
7	Combination of	System Testing is a combination of System Testing and Integration testing.	Acceptance Testing is a combination of alpha testing and beta testing.
8	Order of execution	It is performed before the Acceptance testing.	It is performed after the System testing.
9	Included another type of testing	System testing involves load and stress testing under non-functional testing.	Acceptance testing involves boundary value analysis, equivalence partitioning and decision table under functional testing.
10	Bug fixes	The defects found in system testing are considered to be fixed.	The defects found in acceptance testing are considered as product failure.

Conclusion

In this tutorial, we understood the brief introduction about system testing and acceptance testing and seen the vital difference between them.

Here, we can conclude that both **system and acceptance** testing play a significant role while testing any software or application.

In other words, we can say that System testing is primarily performed to test the needs of a system. On the other hand, acceptance testing is implemented to validate the system performance through an end-user, and UAT is executed by the domain expert(customer) for their satisfaction.

System testing is a type of **black-box testing** that is being performed to validate the integrated software or application in contradiction to the detailed requirements.

User **acceptance or acceptance testing** includes testing the software, compared to the user requirement, to meet the **acceptance measures** and get software products readily known.

In **acceptance testing**, the business representatives' must be satisfied that the developed product sees their needs in the business environment. And **System testing** should meet the productivity requirements of the system.

Quality Assurance vs Quality Control

Quality Assurance

Software quality assurance is (also known as QA) a sequence of tasks to prevent defects and ensure that the techniques, methods, approaches, and processes are designed for a specific application must be implemented correctly. This is an ongoing process within the development of a software system.

The development of units of an application is checked under the quality assurance specifications in the sequence of their development.

Quality assurance test ensures the development of high-quality software because of its main focus on the high-quality processes, good quality management system and periodic conformance audit during the development of software. It is a managerial tool includes planned and systematic activities and documentation to prevent problems related to quality.

The responsibility of quality assurance is not of any specific team, but it is a responsibility of each member of the development team.

1. Quality assurance prevents defects.
2. Quality assurance is process oriented.
3. Quality assurance is proactive in a process and preventive in nature.
4. Quality assurance is a managerial tool.
5. Each developer is responsible for quality assurance.

Quality Control

Quality Control also known as QC is a sequence of tasks to ensure the quality of software by identifying defects and correction of defects in the developed software. It is a reactive process, and the main purpose of this process is to correct all types of defects before releasing the software. The process is done by eliminating sources of problems (which cause to low the quality) through the corrective tools so that software can meet customer's requirements and high quality.

The responsibility of quality control is of a specific team which is known as a testing team that tests the defects of software by validation and corrective tools.

1. Quality Control provides identification of defects.
2. Quality Control is product oriented.
3. Quality Control is a corrective tool.
4. Testing team is responsible for Quality control.
5. Quality Control is a reactive process.

Difference between Quality Assurance and Quality Control

Points	Quality Assurance	Quality Control
Definition	QA is a group of activities which ensures that the quality of processes which is used during the development of the software always be maintained.	QC is a group of activities to detect the defects in the developed software.
Focus	The focus of QA is to prevent defects in the developing software by paying attention to processes.	The focus of QC is to identify defects in the developed software by paying attention to testing processes.

How	Establishment of the high-quality management system and periodic audits for conformance of the operations of the developing software.	Detecting and eliminating the quality problem elements by using testing techniques and tools in the developed software.
What	QA ensures prevention of quality problem elements by using systematic activities including documentation.	QC ensures identification and elimination of defects by using processes and techniques to achieve and maintain high quality of the software.
Orientation	QA is process oriented .	QC is product oriented .
Type of process	QA is a proactive process. It concerns to improve development so; defects do not arise in the testing period.	QC is a reactive process because it concerns to identify defects after the development of product and before its release.
Responsibility	Each and every member of the development team is responsible for QA	Only the specific testing team is responsible for QC
Example	Verification is the example of QA	Validation is the example of QC

Static Testing vs. Dynamic Testing

Static Testing

Static testing is testing, which checks the application without executing the code. It is a verification process. Some of the essential activities are done under static testing such as business requirement review, design review, code walkthroughs, and the test documentation review.

Static testing is performed in the white box testing phase, where the programmer checks every line of the code before handing over to the Test Engineer.

Static testing can be done manually or with the help of tools to improve the quality of the application by finding the error at the early stage of development; that's why it is also called the verification process.

The documents review, high and low-level design review, code walkthrough take place in the verification process.

Dynamic Testing

Dynamic testing is testing, which is done when the code is executed at the run time environment. It is a validation process where functional testing [unit, integration, and system testing] and non-functional testing [user acceptance testing] are performed.

We will perform the dynamic testing to check whether the application or software is working fine during and after the installation of the application without any error.

Difference between Static testing and Dynamic Testing

Static testing	Dynamic testing
In static testing, we will check the code or the application without executing the code.	In dynamic testing, we will check the code/application by executing the code.
Static testing includes activities like code Review, Walkthrough, etc.	Dynamic testing includes activities like functional and non-functional testing such as UT (usability testing), IT (integration testing), ST (System testing) & UAT (user acceptance testing).
Static testing is a Verification Process.	Dynamic testing is a Validation Process.
Static testing is used to prevent defects.	Dynamic testing is used to find and fix the defects.
Static testing is a more cost-effective process.	Dynamic testing is a less cost-effective process.
This type of testing can be performed before the compilation of code.	Dynamic testing can be done only after the executables are prepared.
Under static testing, we can perform the statement coverage testing and structural testing.	Equivalence Partitioning and Boundary Value Analysis technique are performed under dynamic testing.
It involves the checklist and process which has been followed by the test engineer.	This type of testing required the test case for the execution of the code.

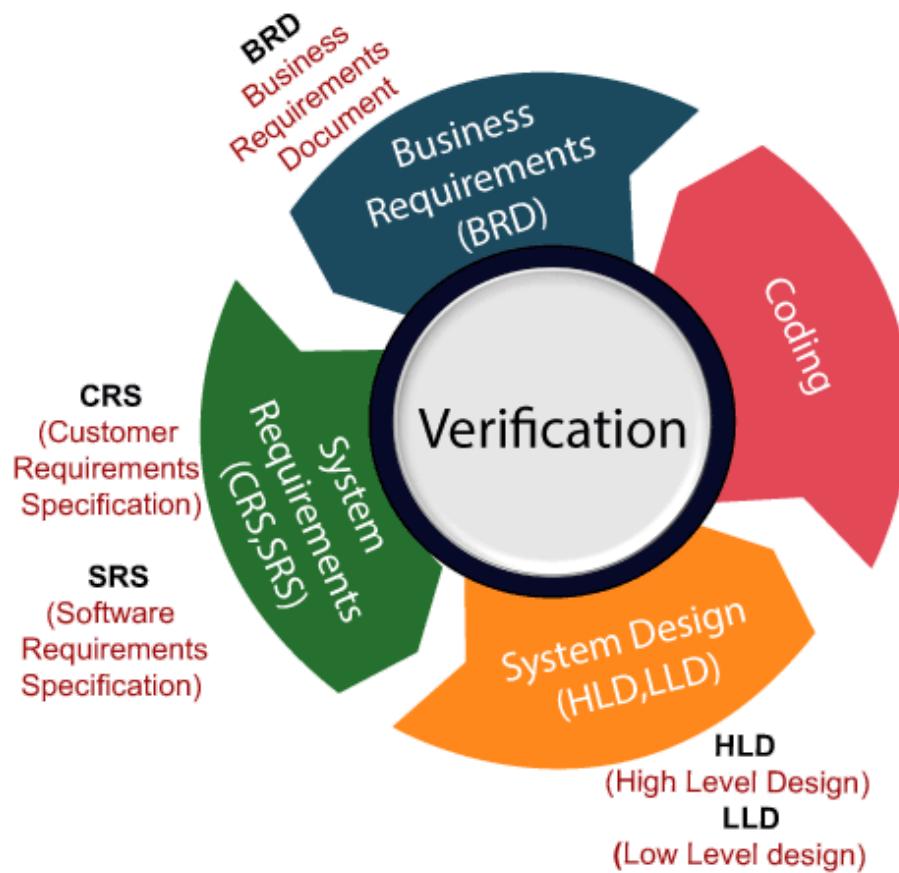
Verification and Validation Testing

In this section, we will learn about verification and validation testing and their major differences.

Verification testing

Verification testing includes different activities such as business requirements, system requirements, design review, and code walkthrough while developing a product.

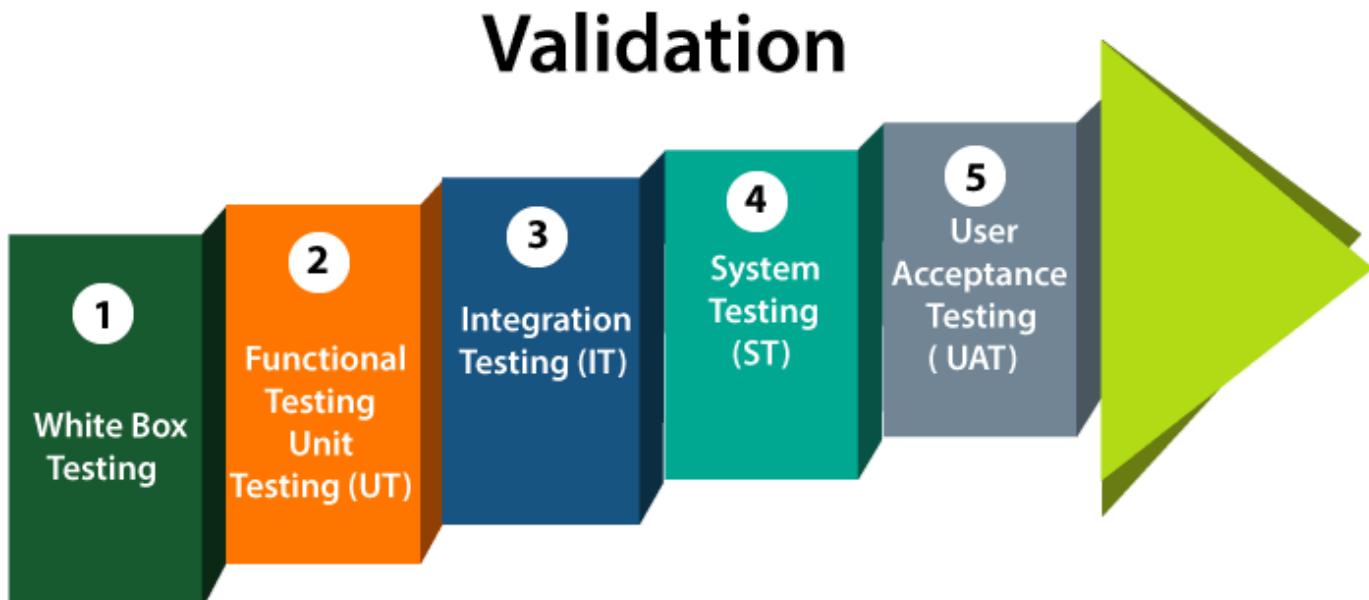
It is also known as static testing, where we are ensuring that "**we are developing the right product or not**". And it also checks that the developed application fulfilling all the requirements given by the client.



Validation testing

Validation testing is testing where tester performed functional and non-functional testing. Here **functional testing** includes **Unit Testing (UT)**, **Integration Testing (IT)** and **System Testing (ST)**, and **non-functional** testing includes User acceptance testing (UAT).

Validation testing is also known as dynamic testing, where we are ensuring that "**we have developed the product right**." And it also checks that the software meets the business needs of the client.



Note: Verification and Validation process are done under the V model of the software development life cycle.

Difference between verification and validation testing

Verification	Validation
We check whether we are developing the right product or not.	We check whether the developed product is right.
Verification is also known as static testing .	Validation is also known as dynamic testing .
Verification includes different methods like Inspections, Reviews, and Walkthroughs.	Validation includes testing like functional testing , system testing, integration , and User acceptance testing.
It is a process of checking the work-products (not the final product) of a development cycle to decide whether the product meets the specified requirements.	It is a process of checking the software during or at the end of the development cycle to decide whether the software follow the specified business requirements.
Quality assurance comes under verification testing.	Quality control comes under validation testing.
The execution of code does not happen in the verification testing.	In validation testing, the execution of code happens.
In verification testing, we can find the bugs early in the development phase of the product.	In the validation testing, we can find those bugs, which are not caught in the verification process.
Verification testing is executed by the Quality assurance team to make sure that the product is developed according to customers' requirements.	Validation testing is executed by the testing team to test the application.
Verification is done before the validation testing.	After verification testing, validation testing takes place.
In this type of testing, we can verify that the inputs follow the outputs or not.	In this type of testing, we can validate that the user accepts the product or not.

What are the differences between Alpha Testing and Beta Testing?

Alpha testing is a type of acceptance testing, which is performed to identify all possible bugs/issues before releasing the product to the end-user. Alpha test is a preliminary software field test carried out by a team of users to find out the bugs that were not found previously by other tests. Alpha testing is to simulate a real user environment by carrying out tasks and operations that actual user might perform. Alpha testing implies a meeting with a software vendor and client to ensure that the developers appropriately meet the client's requirements in terms of the performance, functionality, and durability of the software.

Alpha testing needs lab environment, and usually, the testers are an internal employee of the organization. This testing is called alpha because it is done early on, near the end of the software development, but before beta testing.

Beta Testing is a type of acceptance testing; it is the final test before shipping a product to the customers. Beta testing of a product is implemented by "real users" of the software application in a "real environment." In this phase of testing, the software is released to a limited number of end-users of the product to obtain feedback on the product quality. It allows the real customers an opportunity to provide inputs into the design, functionality, and usability of the product. These inputs are essential for the success of the product. Beta testing reduces product failure risks and increases the quality of the product through customer validation. Direct feedback from customers is a significant advantage of beta testing. This testing helps to test the software in a real environment. The experiences of the previous users are forwarded back to the developers who make final changes before releasing the software product.

Differences between the Alpha testing and Beta testing are:

Sr. No.	Alpha Testing	Beta Testing
1.	Alpha testing performed by a team of highly skilled testers who are usually the internal employee of the organization.	Beta testing performed by clients or end-users in a real-time environment, who is not an employee of the organization.
2.	Alpha testing performed at the developer's site; it always needs a testing environment or lab environment.	Beta testing doesn't need any lab environment or the testing environment; it is performed at a client's location or end-user of the product.
3.	Reliability or security testing not performed in-depth in alpha testing.	Reliability, security, and robustness checked during beta testing.
4.	Alpha testing involves both white box and black-box techniques.	Beta testing uses only black-box testing.
5.	Long execution cycles maybe require for alpha testing.	Only a few weeks are required for the execution of beta testing.
6.	Critical issues or fixes can be identified by developers immediately in alpha testing.	Most of the issues or feedback is collected from the beta testing will be implemented for the future versions of the product.
7.	Alpha testing performed before the launch of the product into the market.	At the time of software product marketing.
8.	Alpha testing focuses on the product's quality before going to beta testing.	Beta testing concentrates on the quality of the product, but gathers user input on the product and ensures that the product is ready for real-time users.
9.	Alpha testing performed nearly the end of the software development.	Beta testing is a final test before shipping a product to the customers.

10.

Alpha testing is conducted in the presence of developers and the absence of end-users.

Beta testing is the reverse of alpha testing.

Black Box Testing vs. White Box Testing vs. Grey Box Testing

Index	Black Box Testing	White Box Testing	Grey Box Testing
1	Knowledge of internal working structure (Code) is not required for this type of testing. Only GUI (Graphical User Interface) is required for test cases.	Knowledge of internal working structure (Coding of software) is necessarily required for this type of testing.	Partially Knowledge of the internal working structure is required.
2	Black Box Testing is also known as functional testing, data-driven testing, and closed box testing.	White Box Testing is also known as structural testing, clear box testing, code-based testing, and transparent testing.	Grey Box Testing is also known as translucent testing as the tester has limited knowledge of coding.
3	The approach towards testing includes trial techniques and error guessing method because tester does not need knowledge of internal coding of the software.	White Box Testing is proceeded by verifying the system boundaries and data domains inherent in the software as there is no lack of internal coding knowledge.	If the tester has knowledge of coding, then it is proceeded by validating data domains and internal system boundaries of the software.
4	The testing space of tables for inputs (inputs to be used for creating test cases) is pretty huge and largest among all testing spaces.	The testing space of tables for inputs (inputs to be used for creating test cases) is less as compared to Black Box testing.	The testing space of tables for inputs (inputs to be used for creating test cases) is smaller than Black Box and White Box testing.
5	It is very difficult to discover hidden errors of the software because errors can be due to internal working which is unknown for Black Box testing.	It is simple to discover hidden errors because it can be due to internal working which is deeply explored in White Box testing.	Difficult to discover the hidden error. Might be found in user level testing.
6	It is not considered for algorithm testing.	It is well suitable and recommended for algorithm testing.	It is not considered for algorithm testing.
7	Time consumption in Black Box testing depends upon the availability of the functional specifications.	White Box testing takes a long time to design test cases due to lengthy code.	Test cases designing can be done in a short time period.
8	Tester, developer and the end user can be the part of testing.	Only tester and developer can be a part of testing; the end user can not involve.	Tester, developer and the end user can be the part of testing.
9	It is the least time-consuming process among all the testing processes.	The entire testing process is the most time consuming among all the testing processes.	less time consuming than White Box testing.
10	Resilience and security against viral attacks are covered under Black Box testing.	Resilience and security against viral attacks are not covered under White Box testing.	Resilience and security against viral attacks are not covered under Grey Box testing.

11	The base of this testing is external expectations internal behavior is unknown.	The base of this testing is coding which is responsible for internal working.	Testing based on high-level database diagrams and dataflow diagrams.
12	It is less exhaustive than White Box and Grey Box testing methods.	It is most exhaustive between Black Box and Grey Box testing methods.	Partly exhaustive; depends upon the type of test cases are coding based or GUI based.

Difference Between Globalization Testing and Localization Testing

In this section, we are going to understand the major differences between globalization testing and localization testing.

Globalization testing and Localization testing are a widely used type of software testing by many small and large enterprises.

What is Globalization testing?

It is another type of software testing used to test the software developed for multiple languages, is called **globalization testing**, and improving the application or software for various languages is known as **globalization**.

Globalization testing ensures that the application will support multiple languages and multiple features because, in current scenarios, we can see the enhancement in several technologies as the applications are planned to be used globally.

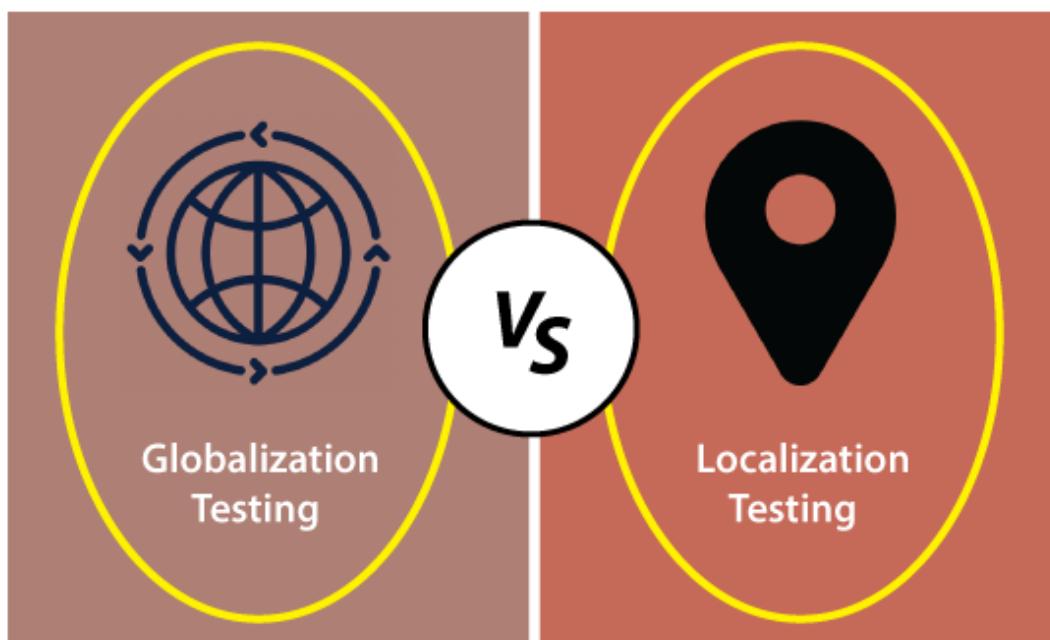
What is Localization testing?

It is nothing but a format of **software testing**. We test the particular application based on the country, region, etc. As we know, the Localized product only supports the precise kind of languages that are usable only in a specific region.

It is also known as **L10N** testing, and here **10** is the number between **L** and **N** in the **Localization** word.

Globalization Testing vs. Localization Testing

Let us see some of the essential differences between Globalization Testing and Localization Testing.



S.NO	Globalization Testing	Localization Testing
1.	Globalization testing is performed to make sure that the software application supports various languages along with multiple features.	The localization testing is performed to verify that the software application supports a precise kind of language which can be used only in a specific region.
2.	In a globalized product, a code is separated from the messages or information. By using globalization testing, we can enable the software to be used with different languages without redesigning the complete software.	The localized product is not required in localization testing.
3.	It mainly emphasizes users as the generic user base.	It mainly emphasizes a small group of users in a given culture or locale.

4.	The globalization testing will help us identify the potential issues and bugs in the software, preventing its expected performance.	The localization testing will help us to detect typographical errors.
5.	It verifies all the functionality of the software product with the help of all the possible international inputs.	The localization testing will help us to verify all application resources.
6.	While executing the globalization testing, the test engineer assumes that the software product is being tested and used across the world.	While executing the localization testing, the test engineer assumes that the software product is being tested and going to be used by a certain group of users in a particular locality.
7.	The globalization testing validates the different currency formats such as an address, mobile number formats are maintained by the software application.	The localization testing validates that the particular address format, currency format, and mobile number format are working fine or not.
8.	It helps to separate the test engineer from translators and engineers and ensure a detailed and independent approach.	It helps to diminish the time for testing; subsequently, it's done just on locale.
9.	Globalization testing takes comparatively time to implement the tests.	Localization testing will take less time to implement tests.
10.	In globalization testing, we have formalized bug reporting.	The localization testing helps us to decreases overall testing and support costs.
11.	For example, www.google.com supports many languages, and people from different countries can access it; therefore, it is a globalized product.	For example, QQ.com supports only the Chinese language, which can be accessed only by a few countries.

Conclusion

Using **globalization testing and localization testing**, we can test some of the most important software features, such as **linguistic relevance, Cultural sensitivity, and the software's global and locale appeal**. With the help of these components, the software can be used by the global and local audience and increase the popularity of the software.

The **globalization testing and localization testing** will help the testing team takes required procedures to enhance the **performance, quality, functionality**, and other significant fundamentals of the software product.

Lastly, we can say that if we do not perform **globalization testing and localization testing** techniques, the software engineers cannot develop a good quality software product, which meets the necessities of the end-user and market.

Test Case Vs. Test Scenarios

In this section, we are going to discuss the difference between the **Test Case** and **Test Scenario** as we understood that both the **test case** and **test scenario** are part of Testing documentation.

As test engineers, we must be aware of these two testing terms as they play a major role in the [Software Test Life Cycle's Test Design and Test Execution](#) phases.

But before we see the difference between **Test Case** and **Test Scenarios**, first, **we will understand the definitions of Test Case and Test Scenario.**

What is a Test Case?

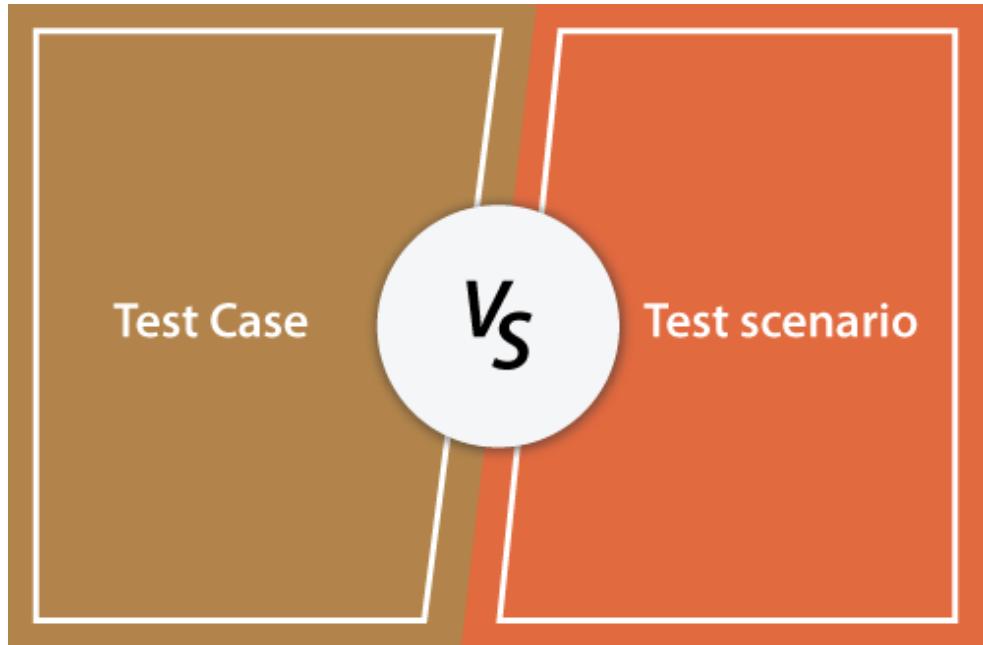
The **test case** is an in-detailed document that includes **all possible inputs** such as **positive and negative**, and **the navigation steps**. These are implemented during the testing process to check whether the software application is performing the task for that it was developed or not.

What is Test Scenarios?

It is a detailed document of test cases that cover **end to end functionality** of a software application in liner statements. The **test scenario** is a high-level classification of testable requirements. Before performing the test scenario, the test engineer needs to consider the test cases for each scenario.

Difference between Test case and Test Scenarios

In the below table, we have listed some of the important difference between **test case** and **test scenarios**:



S.NO	Test Case	Test Scenarios
1.	The test case is a detailed document, which provides information about the testing strategy , testing process , preconditions , and expected output .	The test scenarios are those derived from the use case and give the one-line information about what to test .
2.	It includes all the positive and negative inputs , navigation steps , Expected results , pre and post condition , etc.	Test scenarios are one-liner statement, but it is connected with several test cases.
3.	These are low-level actions .	These are high-level actions .
4.	The main objective of writing the test case is to verify the test scenario by implementing steps.	Writing the test scenario's main objective is a cover end to end functionality of a software application .
5.	It takes more time as compared to test scenarios.	It takes less time as compared to test cases.

6.	The test cases are hard to maintain .	Test scenarios are easy to maintain due to their high-level design.
7.	The test case will help our in-depth testing of the application.	The test scenario will help us in an agile way of testing throughout the functionality.
8.	The test case is work on the basics of " How to be tested ".	The test scenarios are work on the basic to " What to be tested ".
9.	To write the test, we need additional resources to create and perform test cases.	Fewer resources are sufficient to write test scenarios as compared to the test cases.
10.	In a test case, we mainly focus on the document details .	In test scenarios, we will focus on thinking and discussing details .
11.	It can obtain from test scenarios .	It can obtain from the use cases .
12.	If the test engineer is working offside, then the creation of test cases is significant.	If the tester is particularly working in Agile methodology, then creating test scenarios helps us in a time-sensitive situation.
13.	Writing test cases is a one-time attempt that can be used in the future at the time of regression testing.	The test scenarios are a time saving process; that's why it is preferred by the new generation of software testing community.

Conclusion

In this section, we have understood the essential differences and impotence of both **test case and test scenarios**.

Using both the **test case and test scenario** together ensures **robustness and high coverage testing creativity**.

It's the best exercise if we write **Test Scenarios** first and then write the **Test Cases** because most IT companies prefer Test Scenarios in today's Agile time. In the agile era, the test case is being replaced with test scenarios to save time.

Test Plan VS. Test Strategy

In this section, we are going to discuss the major difference between the **Test Plan and Test Strategies** as we already know that both are an important part of Testing documentation.

As test engineers, we must be aware of these two testing terms as they play a major role in the **Software Test Life Cycle (STLC)**. And for the interview purpose, it is the most commonly asked question.

But before we get into the difference between **Test Plan and Test Strategy** first, **we will understand the concepts of Test Plan and Test Strategy in brief.**

What is Test Plan?

The **test plan** is a base of **software testing**. It is a detailed document, which includes several testing attributes such as **test objectives, scope, test schedule, template, required resources (human resources, software, and hardware), test estimation and test deliverables, risk, mitigation plan, defect tracking, entry and exit criteria, test environment**, etc., which defines software testing areas and activities.

The test plans play a major role in testing and help us deliver a quality product.

What is a Test Strategy?

The test strategy is a high-level document used to validate the test levels to be executed for the product. And it also describes what kind of technique has to be used and which module will be tested.

It contains various components like **documentation formats, objectives, test processes, scope, customer communication strategy**, etc.

The Test Strategy's main purpose is to deliver a systematic approach to the software testing process to ensure **reliability, quality, traceability, and better planning**.

Difference between Test Plan and Test Strategy

In the below table, we have listed some of the important difference between Test Plan and Test Strategy:



S.NO	Test Plan	Test Strategy
1.	It is a formal document used to define the scope of testing and different testing activities.	It is a high-level document that involves planning for all the testing activities and delivering a quality product.
2.	It is derived with the help of Use Case documents, SRS (Software Requirement Specification), and Product Description.	While the Test Strategy can be derived with the help of the BRS (Business Requirement Specification) document.
3.	A test plan is developed by Test Lead or test/Project manager.	Generally, the test strategy is developed by the Business Analyst and approved by the Project Manager.

4.	A Test Plan is a dynamic document that can be updated frequently when new requirements or modifications have occurred.	It is a static document, which implies that it cannot be changed or modified.
5.	A test plan defines the whole testing activities thoroughly.	The test strategy defines high-level test design methods.
6.	A test plan is specified at the project level.	A test strategy is specified at the organization level, which can be used by multiple projects.
7.	<p>The important testing attributes included in the test plan are as follows:</p> <ul style="list-style-type: none"> ○ Objective ○ Scope ○ Test methodology ○ Approach ○ Assumption ○ Risk ○ Mitigation plan, ○ Roles and responsibility ○ Schedule ○ Bug tracking ○ Test environments ○ Entry and exit criteria ○ Test automation ○ Effort estimation, ○ Test deliverables ○ Templates 	<p>The significant testing components include in the test strategy are as follows:</p> <ul style="list-style-type: none"> ○ Scope and overview ○ Testing tools ○ Testing metrics ○ Requirement Traceability Matrix ○ Training plan, ○ Business issues ○ Reporting tool ○ Change and configuration management ○ Test summary

Conclusion

Fundamentally, in a **Test plan**, all the names of the test engineers and test cycle numbers have been mentioned who tested a particular script. Therefore, if some feature fails in the particular cycle, we can easily refer to the previous cycle to check if that specified module was passed or failed.

Test strategy can't frequently change as it sets some standards for the test plan. And it becomes difficult to stick to a precise plan and modified strategy because if the requirement changes happen repeatedly, it will affect the quality of the testing, and we may not be able to deliver a quality product.

To get the complete information on the **test plan and test strategy documents**, check our tutorials on Testing documentation and **Test Plan**.

Difference between Boundary value analysis & Equivalence partitioning

In this section, we are going to discuss some of the important differences between **Boundary value analysis** and **Equivalence Partitioning**.

As we already know that both **Boundary value analysis** and **Equivalence Partitioning technique** are part of **test case design techniques** in black-box testing.

But before we see the difference between **Boundary value analysis** and **Equivalence Partitioning**, first, we will understand both the terms in brief.

What is Boundary value analysis?

It is one of the extensively used test case design techniques for **black-box testing**. In this, we will test the boundary values as the input values near the boundary have higher chances of error. It is applicable at all levels of the testing process.

Whenever we perform the boundary value analysis technique, the test engineer focuses on entering boundary value whether the software is creating correct output or not.

What is Equivalence partitioning?

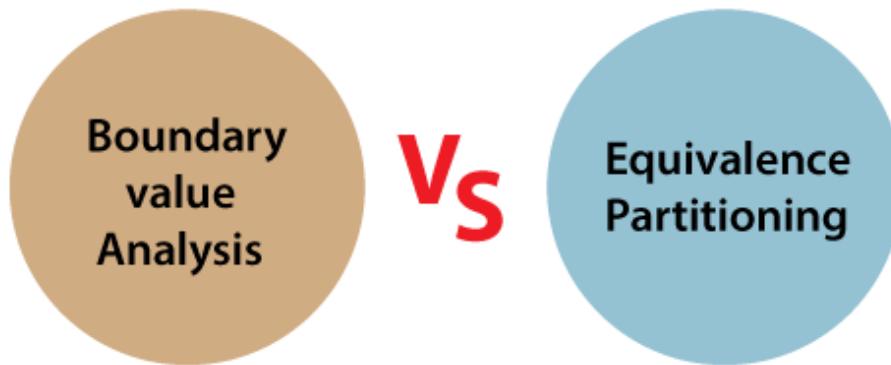
Another **test case** design technique is Equivalence partitioning, which is derived from the software's requirements and specifications. To cover maximum requirements Equivalence Partitioning uses the minimum test cases.

In this, the **test cases** should be designed to cover each partition at least once. And each value of every equal partition must display the same behavior as the other.

Boundary value analysis Vs. Equivalence partitioning

In the below table, we have listed some of the important difference between

Boundary value analysis and Equivalence partitioning:



S.NO.	Boundary value analysis	Equivalence partitioning
1.	It is a technique where we identify the errors at the boundaries of input data to discover those errors in the input center.	It is a technique where the input data is divided into partitions of valid and invalid values.
2.	Boundary values are those that contain the upper and lower limit of a variable.	In this, the inputs to the software or the application are separated into groups expected to show similar behavior.
3.	Boundary value analysis is testing the boundaries between partitions.	It allows us to divide a set of test conditions into a partition that should be considered the same.
4.	It will help decrease testing time due to a lesser number of test cases from infinite to finite.	The Equivalence partitioning will reduce the number of test cases to a finite list of testable test cases covering maximum possibilities.

5.	The Boundary Value Analysis is often called a part of the Stress and Negative Testing.	The Equivalence partitioning can be suitable for all the software testing levels such as unit, integration, system.
6.	Sometimes the boundary value analysis is also known as Range Checking.	Equivalence partitioning is also known as Equivalence class partitioning.

Conclusion

After seeing all the major differences between **Boundary Value Analysis** and **Equivalence Partitioning**, we must conclude that the boundary value analysis is a better approach than Equivalence Partitioning.

Suppose testing values are repeated while comparing Equivalence Partitioning and Boundary Value Analysis. In that case, we can neglect the Equivalence Partitioning and perform only Boundary Value analysis as it covers all the values.

Therefore, the **Boundary Value Analysis** proves to be a good option in assuring the quality after the **Equivalence Partitioning** technique.

SDLC VS. STLC

In this section, we are going to discuss the difference between the **SDLC** and **STLC** as we understood that both are an integral part of Software testing.

But before we see the difference between **SDLC** and **STLC**, first, we will understand the **SDLC** and **STLC** in brief.

What is SDLC?

Software Development Life Cycle [SDLC] is a classification of individual activities executed throughout the software development process.

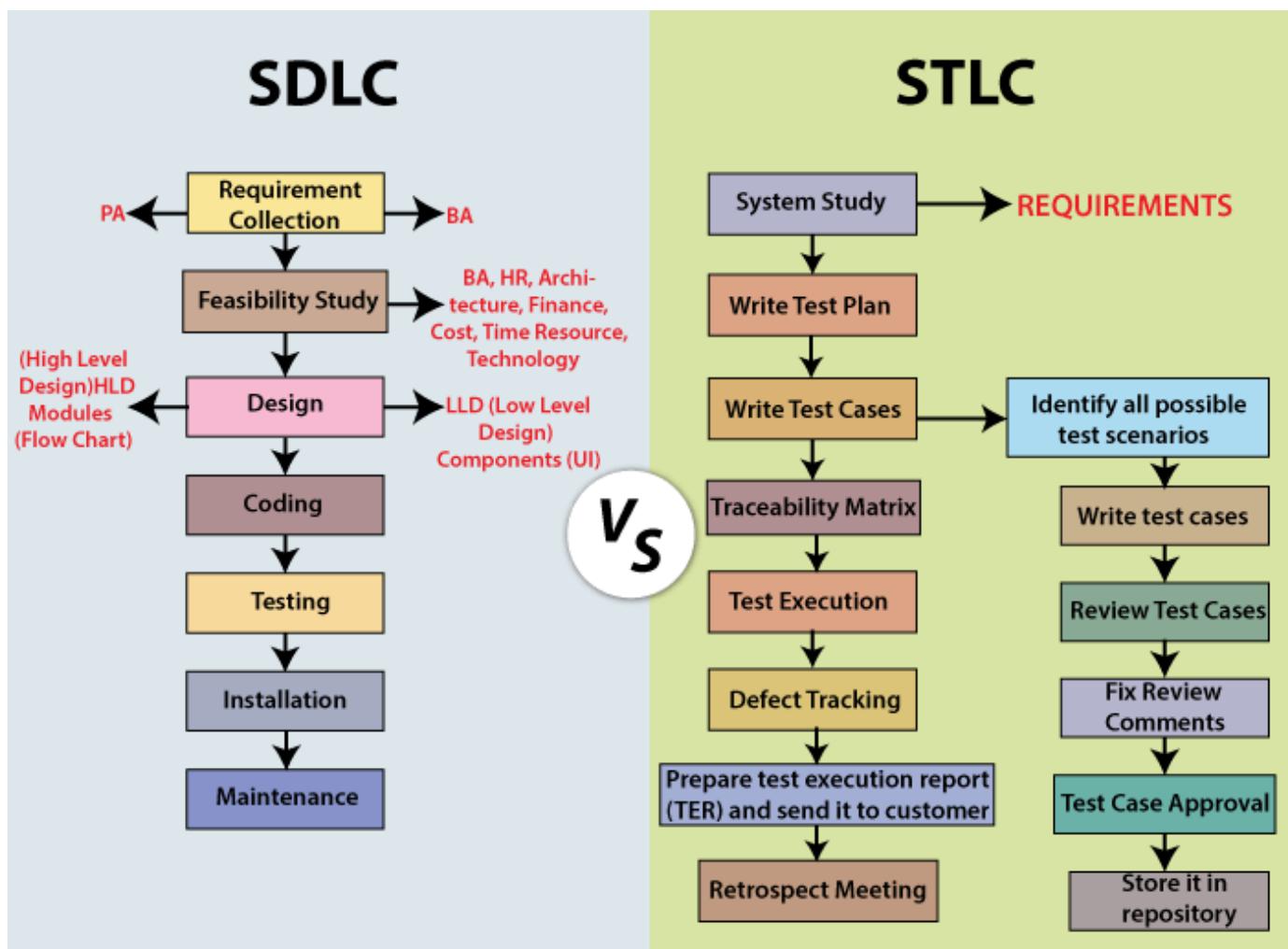
The SDLC includes various phases, and each phase has several activities, which help the development team design, create, and deliver a high-quality product.

What is STLC?

Software Testing Life Cycle [STLC] is the order of different activities executed throughout the software testing process. Testing itself has many phases called STLC, and each activity is done to improve the quality of the software product.

Difference between SDLC and STLC

In the below table, we have listed some of the important difference between the **Software Development Life Cycle** and **Software Testing Life Cycle**:



S.NO	Comparison basis	SDLC	STLC
1.	Explanations	It is primarily connected to software development, which means that it is the procedure of developing a software application.	It is mainly linked to software testing, which means that it is a software testing process that contains various phases of the testing process.

2.	Representation	SDLC stands for Software Development Life Cycle .	STLC stands for Software Testing Life cycle .
3.	Resources	While performing the SDLC process, we needed a greater number of developers to complete the development process.	The STLC process needed a smaller number of testers to complete the testing process.
4.	Focuses on	Besides the development phase, other phases like testing are also included.	The STLC concentrate only on testing the software.
5.	Objective	The objective of the Software development life cycle is to complete the development of software successfully.	The objective of the Software testing life cycle is to complete the testing of software successfully.
6.	Help in	The SDLC will help us to develop a good quality software product.	The STLC will helps to create the software bug-free.
7.	Different phases	<p>The various phase includes in Software Development Life Cycle are as follows:</p> <ul style="list-style-type: none"> ◦ Requirements Collection ◦ Feasibility Study ◦ Design ◦ Programming or Coding ◦ Testing ◦ Installation ◦ Maintenance 	<p>The various phase includes in Software Testing Life Cycle are as follows:</p> <ul style="list-style-type: none"> ◦ Requirement collection or System study ◦ Test Plan ◦ Write test case ◦ Traceability Matrix ◦ Defect Tracking ◦ Test Execution Report ◦ Retrospect meeting
8.	Requirement collection phase	In the SDLC Requirement collection phase, the BA [Business Analyst] and PA [Product Analyst] will collect the requirements and interpret business language into software language.	In the Requirement Analysis phase of the STLC, the QA [Quality Assurance] team will study requirement documents and prepare the System Test Plan.
9.	Designing phase	Based on the requirement understanding, the development team will develop the HLD [High-Level Design] and LLD [Low-Level Design] of the software.	Generally, in STLC, the Test Architect or a Test Lead plan the test strategy. And also finds the testing points.
10.	Coding phase	In the SDLC coding phase, the developer will start writing the code as per the designed document and beginning of building the software.	In STLC, the QA team writes the test scenarios to authenticate the quality of the product.
11.	Environment Set up	After writing the code, the development team sets up a test environment with the developed product to validate the code.	Based on the prerequisites, the Test team confirms the environment set up. And do one round of smoke testing to ensure that the environment is stable for the product and ready for testing.

12.	Testing Phase	Once the environment has been set, the test engineer will perform various types of testing, such as Unit, Integration, System, Retesting , Regression testing, and so on. And the development team is also involving to fixing the bugs and report back to the tester.	Based on the test cases, the tester will do one round of integration and system testing. While performing the testing, if they encounter with any bugs, it will be reported and fixed after the retesting.
13.	Deployment/ Product Release phase	In the SDLC deployment phase, when we received sign-off from various testing teams, the application is deployed or installed in a production environment for real end-users.	In STLC, the Smoke and sanity testing are performed in the production environment as soon as the product is deployed. And the testing team will prepare the test reports and matrix to analyze the product.
14.	Maintenance Phase	Once the product has been deployed, the development team includes support and release updates.	To check maintenance code deployed, the QA team performs the regression suites.
15.	Performed	The SDLC phases are done before the STLC phases.	The STLC phases are completed after SDLC phases.

Conclusion

The **SDLC** and **STLC** provide a structure to the **development** and **testing** of software.

Generally, the test engineer may feel that the **Software development life cycle** is appropriate to developers only. Still, after understood the complete process of the **development and testing life cycle**, we can say that both life cycles are dependent and time to time performed in parallel.

Hence, it is beneficial even to test engineers if they understand the **SDLC phase** with **STLC**.

Difference between Bug, Defect, Error, Fault & Failure

In this section, we are going to discuss the difference between the **Bug, Defect, Error, Fault & Failure** as we understood that all the terms are used whenever the system or an application act abnormally.

Sometimes we call it an **error** and sometimes a bug or a **defect** and so on. In software testing, many of the new test engineers have confusion in using these terminologies.

Generally, we used these terms in the **Software Development Life Cycle (SDLC)** based on the phases. But there is a conflict in the usage of these terms.

In other words, we can say that in the era of **software testing**, the terms **bugs, defects, error, fault, and failure** come across every second of the day.

But for a beginner or the inexperienced in this field, all these terminologies may seem synonyms. It became essential to understand each of these terms independently if the software doesn't work as expected.

What is a bug?

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

If a **QA (Quality Analyst)** detect a bug, they can reproduce the bug and record it with the help of the **bug report template**.

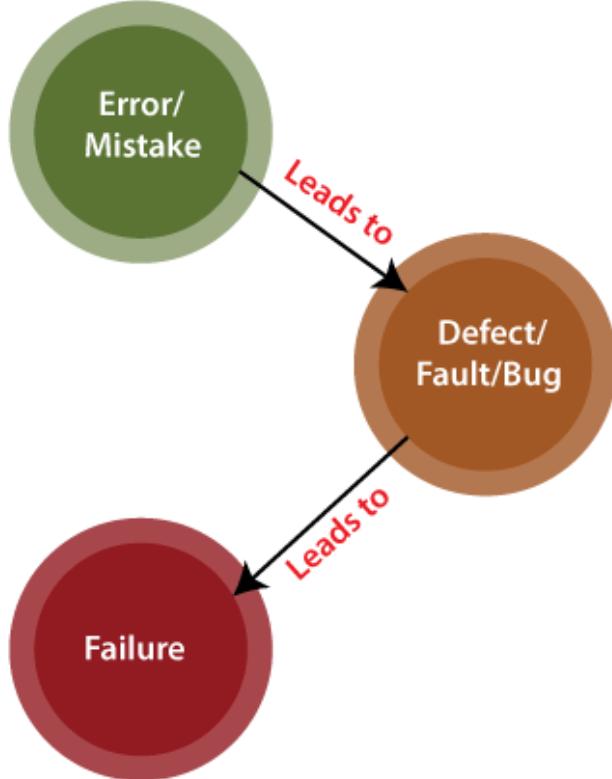
What is a Defect?

When the application is not working as per the requirement is known as **defects**. It is specified as the aberration from the **actual and expected result** of the application or software.

In other words, we can say that the bug announced by the **programmer** and inside the code is called a **Defect**.

What is Error?

The Problem in code leads to errors, which means that a mistake can occur due to the developer's coding error as the developer misunderstood the requirement or the requirement was not defined correctly. The **developers** use the term **error**.



What is Fault?

The fault may occur in software because it has not added the code for fault tolerance, making an application act up.

A fault may happen in a program because of the following reasons:

- Lack of resources
- An invalid step
- Inappropriate data definition

What is Failure?

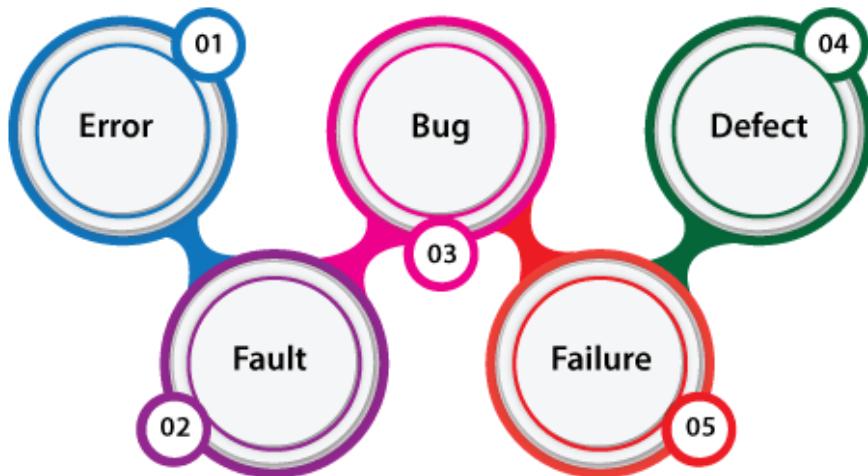
Many defects lead to the **software's failure**, which means that a loss specifies a fatal issue in software/ application or in its module, which makes the system unresponsive or broken.

In other words, we can say that if an end-user detects an issue in the product, then that particular issue is called a **failure**.

Possibilities are there one defect that might lead to one failure or several failures.

For example, in a bank application if the **Amount Transfer** module is not working for end-users when the end-user tries to **transfer money**, submit button is not working. Hence, this is a **failure**.

The flow of the above terminologies are shown in the following image:



Bug Vs. Defect Vs. Error Vs. Fault Vs. Failure

We have listed some of the vital differences between **bug**, **defect**, **error**, **fault**, and **failure** in the below table.

Comparison basis	Bug	Defect	Error	Fault	Failure
Definition	It is an informal name specified to the defect.	The Defect is the difference between the actual outcomes and expected outputs.	An Error is a mistake made in the code; that's why we cannot execute or compile code.	The Fault is a state that causes the software to fail to accomplish its essential function.	If the software has lots of defects, it leads to failure or causes failure.
Raised by	The Test Engineers submit the bug.	The Testers identify the defect. And it was also solved by the developer in the development phase or stage.	The Developers and automation test engineers raise the error.	Human mistakes cause fault.	The failure finds by the manual test engineer through the development cycle .

Different types	Different type of bugs are as follows: <ul style="list-style-type: none"> ◦ Logic bugs ◦ Algorithmic bugs ◦ Resource bugs 	Different type of Defects are as follows: Based on priority: <ul style="list-style-type: none"> ◦ High ◦ Medium ◦ Low And based on the severity: <ul style="list-style-type: none"> ◦ Critical ◦ Major ◦ Minor ◦ Trivial 	Different type of Error is as below: <ul style="list-style-type: none"> ◦ Syntactic Error ◦ User interface error ◦ Flow control error ◦ Error handling error ◦ Calculation error ◦ Hardware error ◦ Testing Error 	Different type of Fault are as follows: <ul style="list-style-type: none"> ◦ Business Logic Faults ◦ Functional and Logical Faults ◦ Faulty GUI ◦ Performance Faults ◦ Security Faults ◦ Software/hardware fault 	-----
Reasons behind	Following are reasons which may cause the bugs: Missing coding Wrong coding Extra coding	The below reason leads to the defects: Giving incorrect and wrong inputs. Dilemmas and errors in the outside behavior and inside structure and design. An error in coding or logic affects the software and causes it to breakdown or the failure.	The reasons for having an error are as follows: Errors in the code. The mistake of some values. If a developer is unable to compile or run a program successfully. Confusions and issues in programming. Invalid login, loop, and syntax. Inconsistency between actual and expected outcomes. Blunders in design or requirement actions. Misperception in understanding the requirements of the application.	The reasons behind the fault are as follows: A Fault may occur by an improper step in the initial stage, process, or data definition. Inconsistency or issue in the program. An irregularity or loophole in the software that leads the software to perform improperly.	Following are some of the most important reasons behind the failure: Environmental condition System usage Users Human error

Way to prevent the reasons	<p>Following are the way to stop the bugs: Test-driven development.</p> <p>Offer programming language support.</p> <p>Adjusting, advanced, and operative development procedures.</p> <p>Evaluating the code systematically.</p>	<p>With the help of the following, we can prevent the Defects:</p> <ul style="list-style-type: none"> Implementing several innovative programming methods. Use of primary and correct software development techniques. Peer review It is executing consistent code reviews to evaluate its quality and correctness. 	<p>Below are ways to prevent the Errors:</p> <ul style="list-style-type: none"> Enhance the software quality with system review and programming. Detect the issues and prepare a suitable mitigation plan. Validate the fixes and verify their quality and precision. 	<p>The fault can be prevented with the help of the following:</p> <ul style="list-style-type: none"> Peer review. Assess the functional necessities of the software. Execute the detailed code analysis. Verify the correctness of software design and programming. 	<p>The way to prevent failure are as follows:</p> <ul style="list-style-type: none"> Confirm re-testing. Review the requirements and revisit the specifications. Implement current protective techniques. Categorize and evaluate errors and issues.
-----------------------------------	--	--	---	---	---

Conclusion

After seeing all the significant differences between **bug, defect, error, fault, and failure**, we can say that the several issues and inconsistencies found throughout software are linked and dependent on each other.

All the above terminology affects and change different parts of the software and differ from one another massively. However, all these differences between **bug, defect, errors, faults, and failures** slow down the software's excellence and performance.

Difference Between Testing and Debugging

In this section, we are going to understand the difference between **Testing** and **Debugging**. Both the terminologies are the integral parts of SDLC as both are used at the different phase of **Software Development Life Cycle** and gives distinct types of outcomes.

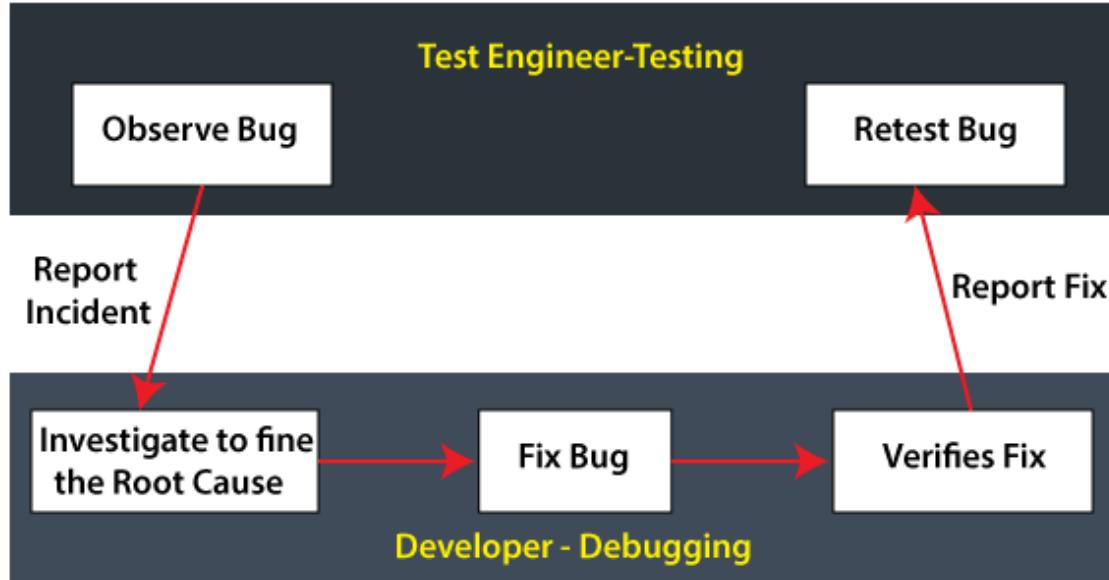
At the time of development and after the outcome of any application or the software product established in any programming language, both **Testing and Debugging** play a vital role in finding and removing mistakes.

Note: Both Testing and Debugging are two words that seem to share a similar meaning but intensively different from one another.

They have quite an equivalent function, but they are diverse in terms of **designs, requirements, benefits, and performance**.

Therefore, it is required for us to understand the **differences between testing and debugging** properly that will support us in receiving better software development outcomes.

Before we see the difference between **testing and debugging**, we will discuss the **in-detail** evaluation of **testing and debugging**, which will help us distinguish both of them appropriately.



What is Software Testing?

Software testing is a process of identifying defects in the software product. It is performed to validate the behavior of the software or the application compared to requirements.

In other words, we can say that the testing is a collection of techniques to determine the accuracy of the application under the predefined specification but, it cannot identify all the defects of the software.

Each software or application needs to be tested before delivering to the clients and checks whether the particular software or the application is working fine as per the given requirements.

What is Debugging?

As opposed to Testing, Debugging is the action where the development team or a developer implements after receiving the test report related to the bugs in the software from the testing team.

In the software development process, debugging includes detecting and modifying code errors in a software program.

In debugging process, the developer needs to identify the reason behind the particular **bug** or defect, which is carried out by analyzing the coding rigorously.

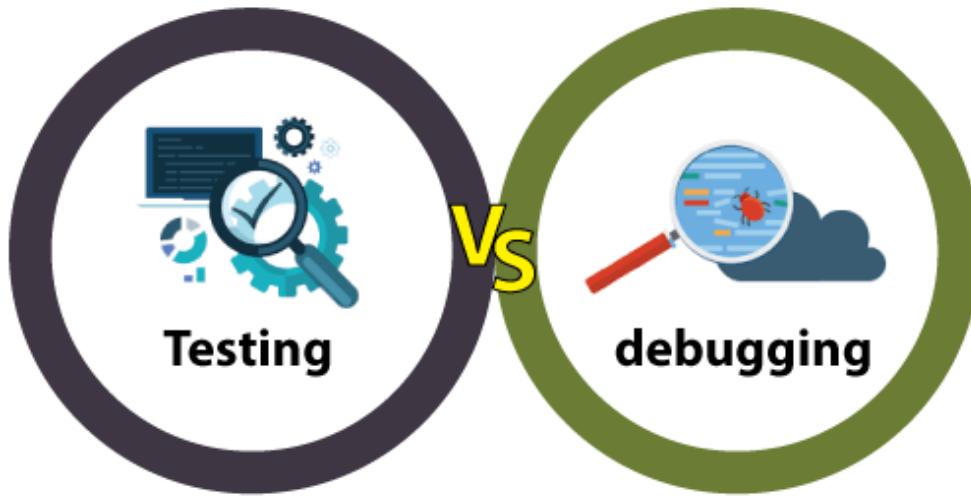
The developer changes the code and then rechecks whether the defect has been deleted whenever the bug or error is found.

Once the debugging is successfully finished, the application is again sent back to the test engineers, who remain in the process of testing.

The debugging process allows us an earlier finding of an error and makes software development stress-free and easy.

Now, based on features and technique of practice, we can distinguish between **Testing and Debugging**.

Testing Vs. Debugging



In the below table, we have listed some of the significant difference between testing and debugging:

S.NO	Testing	Debugging
1.	It is the implementation of the software with the intent of identifying the defects	The process of fixing and resolving the defects is known as debugging.
2.	Testing can be performed either manually or with the help of some automation tools.	The debugging process cannot be automated.
3.	A group of test engineers executes testing, and sometimes it can be performed by the developers.	Debugging is done by the developer or the programmer.
4.	The test engineers perform manual and automated test cases on the application, and if they detect any bug or error, they can report back to the development team for fixing.	The developers will find, evaluate, and remove the software errors.
5.	Programming knowledge is not required to perform the testing process.	Without having an understanding of the programming language, we cannot proceed with the debugging process.
6.	Once the coding phase is done, we proceed with the testing process.	After the implementation of the test case, we can start the Debugging process.
7.	Software Testing includes two or more activities such as validation and verification of the software.	Debugging tries to match indication with cause, hence leading to the error correction.
8.	It is built on different testing levels such as Unit Testing, Integration Testing, System Testing, etc.	It is built on different kinds of bugs because there is no such level of debugging is possible.
9.	Software testing is the presentation of defects.	It is a logical procedure.
10.	Software testing is the vital phase of SDLC (Software Development Life Cycle).	It is not a part of SDLC because it occurs as a subset of testing.

11.	<p>Some advantages of software testing are as below:</p> <ul style="list-style-type: none"> ◦ It can easily understand by the new test engineers or the beginner. ◦ The test engineer can interact with software as a real end-user to check the usability and user interface issues. ◦ It is used to test dynamically altering GUI designs. ◦ Testing is a cost-effective and time-saving process. ◦ Software testing delivers a consistence software. ◦ It will help us to execute the root cause analysis that will enhance the software's productivity. ◦ The testing process also helps detect and fixing the bugs before the software becomes active, which significantly reduces the risk of failure. 	<p>Some advantages of debugging process are as follows:</p> <ul style="list-style-type: none"> ◦ It supports the developer in minimizing the data. ◦ If we perform the debugging, we can report the error condition directly. ◦ During the debugging process, the developer can avoid complex one-use testing code that helps the developer save time and energy. ◦ Debugging delivers maximum useful information of data structures and allows its informal understanding.
12.	<p>Software testing contains various type of testing methods, which are as follow:</p> <ul style="list-style-type: none"> ◦ Black-box testing ◦ White-box testing ◦ Grey-box testing <p>And some other type of testing types is as below:</p> <ul style="list-style-type: none"> ◦ Unit testing ◦ Integration Testing ◦ System Testing ◦ Stress Testing ◦ Performance Testing ◦ Compatibility Testing ◦ Beta Testing ◦ Alpha Testing ◦ Smoke Testing ◦ Regression Testing ◦ User Acceptance Testing and so on. 	<p>Debugging involves a various type of approaches, which are as follows:</p> <ul style="list-style-type: none"> ◦ Induction ◦ Brute Force ◦ Deduction
13.	<p>The testing team can subcontract to the outside team as well.</p>	<p>Debugging cannot be subcontracted to an outside team because the inside development team only does it.</p>
14.	<p>We can plan, design, and implement the testing process.</p>	<p>As compared to the testing process, the debugging process cannot be forced.</p>

Conclusion

In this article, we understood that **testing and debugging** are the essential parts of the **software testing life cycle**, and both of them play a crucial role in detecting bugs and errors in the software.

After seeing the critical differences between testing and debugging, we can say that both terminologies are dependent on each other, which means that one cannot be implemented without the other one.

The outcome of implementing the debugging process is that the issue was fixed and available for re-testing. The test engineer does not fix the defects; relatively, they verify those bugs, which is resolved by the developers.

Finally, we can conclude that the developers execute the debugging, and they fix the issues reported by the test engineers in the debugging phase.

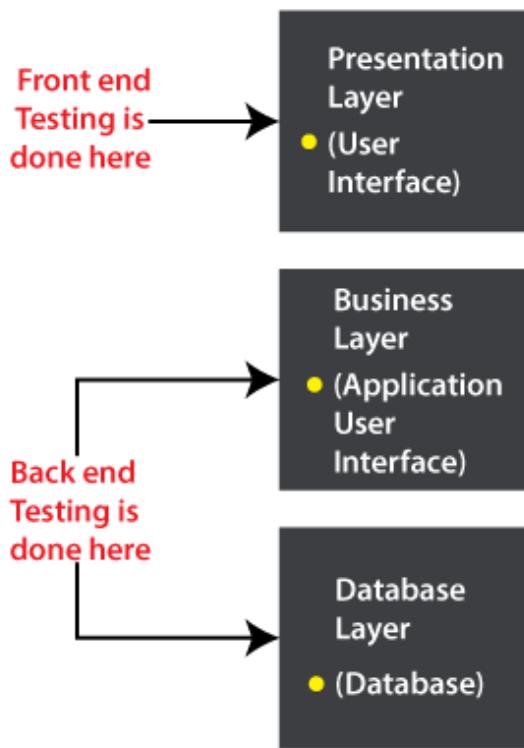
The testing and the debugging process are performed to make the particular software product better and improved way.

And it doesn't depend on which team (testing and debugging team) we belong to.

Frontend Testing VS. Backend Testing

In this section, we are going to see the difference between **Frontend Testing** and **Backend Testing**. The **frontend** and **backend** are mostly used technical teams in the computer industry.

Generally, a web-based application is **three-tier architecture** application. The **first layer** is the **presentation layer** known as **Front-end**, and **the third layer** is the **database layer** known as **backend**.



What is Frontend Testing?

It is a type of **software testing** used to evaluate the **presentation layer** of a **three-tier architecture** in a web application.

In a web application, frontend testing will include the analysis of multiple components such as:

- **Menus**
- **Graphs**
- **Forms**
- **Reports and related JavaScript.**

The word **Frontend testing** is used to covers a diversity of testing approaches.

To execute the **frontend testing**, a test engineer required a good knowledge of business necessities. It can be executed either manually or with the help of some automation tools.

Fundamentally, it is performed on the **UI (user interface)**, which is also known as the **presentation layer** in a **three-tier architecture**.

What is Backend Testing?

Another part of software testing is **backend testing**, which is used to test the **application and Database layer** of a **3-Tier Architecture** in a web application.

Essentially, in order to perform the back-end testing, we do not require any **GUI (Graphical User Interface)** as it is achieved on the **AUI (Application User Interface)** and database.

That's why we can pass the data directly with browsers' help and the parameters needed for the specific function. And to retrieve the response in a nearly predefined format, such as **XML** or **JSON**.

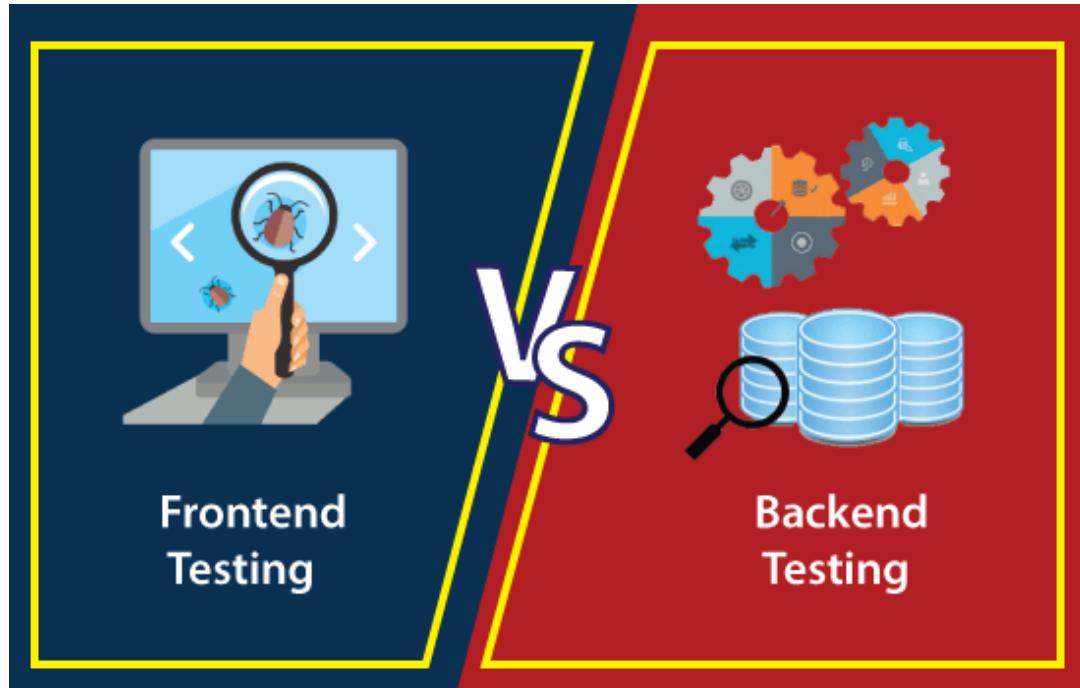
We can also connect to the database directly and validate the data with the help of **SQL** commands.

To test the backend and application layers, we can go for the **backend testing**.

After seeing the brief introduction of frontend and backend testing, we will understand the comparison between them.

Difference between Frontend Testing and Backend Testing

In the below table, we have listed some of the important differences between **frontend and backend** testing.



S.NO	Frontend testing	Backend testing
1.	It is executed on the presentation layer of the 3-tier architecture.	It is performed on the Application and Database layer of the 3-tier architecture.
2.	It is always performed on the Graphical user interface (GUI).	It is always implemented on the Application User Interface (AUI).
3.	While performing the frontend testing, we do not require to store any information in a database.	While performing the backend testing, we need to store the data in the database.
4.	The understanding of requirements is necessary in order to execute the frontend testing.	The understanding of the database is essential to execute the backend testing.
5.	It will analyze the overall capabilities of the application.	It will analyze the deadlock, data corruption, or data loss.
6.	In GUI-based frontend testing, the resources are centrally achieved in cloud computing.	In AUI based backend testing, the resources are executed on a collaboration pattern in Grid Computing.
7.	Knowledge about the automation frameworks tools like QTP, Selenium is mandatory to perform the Frontend testing.	Knowledge about SQL (Structured Query Language) language concepts is compulsory to implement the backend testing.
8.	Frontend testing includes the verification of the application and checks the performance of application whether it is working according to the requirement.	Backend testing execution makes sure that the data is continuing as there is no performance hit.
9.	System testing and Acceptance Testing, unit testing, accessibility testing, and regression testing are performed under frontend testing.	The database testing (API testing and SQL testing) are performed under backend testing.

10.	<p>Just like other types of testing frontend testing also contains some tools, which are as follows:</p> <ul style="list-style-type: none"> ◦ LiveReload ◦ Karma ◦ Grunt 	<p>To execute the backend testing, we have some tools available in the market, which are as follows:</p> <ul style="list-style-type: none"> ◦ DTM Data Generator ◦ TurboData ◦ Data Factory
-----	---	--

Conclusion

In this tutorial, we have made the difference between the **frontend and backend testing**. And we can conclude that both testings' play a significant role in software testing.

The frontend testing contains **HTML, JS, CSS**, and images, whereas the backend testing contains the business logic testing and database testing.

Frontend and backend testing has different activities implemented by the developers that work on both ends of the line.

Finally, we can say that the frontend means the **browser** and the **backend**, the server, or, more recently, the cloud.

Difference Between HLD and LLD

In this section, we are going to discuss the difference between **HLD and LLD**; and see the brief introduction between them.

What is HLD?

The HLD stands for **High-Level Design**, where the designer will only focus on the various models, like:

- **Decision Tables**
- **Decision Trees**
- **Flow Diagrams**
- **Flow Charts**
- **Data Dictionary**

The solution architect develops the High-level design, which is used to specifies the complete description or architecture of the application.

The HLD involves **system architecture, database design, a brief description of systems, services, platforms, and relationships** among modules.

The HLD is also known as **macro-level or system design**. It changes the business or client requirement into a **High-Level Solution**.

The High-level design is created before the Low-Level Design.

What is LLD?

The **LLD** stands for **Low-Level Design**, in which the designer will focus on the components like a **User interface (UI)**.

The Low-level design is created by the **developer manager and designers**.

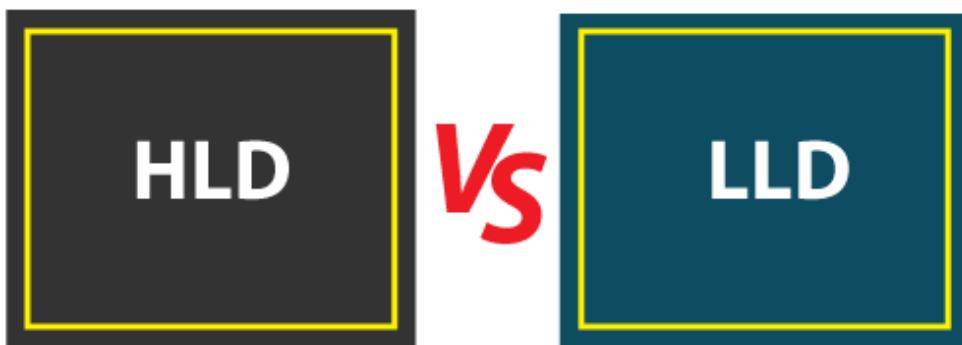
It is also known as **micro-level or detailed design**. The LLD can change the **High-Level Solution** into a **detailed solution**.

The Low-level design specifics the detailed description of all modules, which implies that the LLD involves all the system component's actual logic. It goes deep into each module's specification.

The Low-level design is created after the High-Level Design.

HLD vs LLD

In the below table, we have discussed some significant comparisons between high-level design and low-level design.



S.NO.	Comparison Basis	HLD	LLD
1.	Stands for	It stands for High-Level Design.	It stands for Low-Level Design.

2.	Definition	It is the general system design, which means it signifies the overall system design.	It is like describing high-level design, which means it signifies the procedure of the component-level design.
3.	Purpose	The HLD states the concise functionality of each component.	LLD states the particular efficient logic of the component.
4.	Also, known as	HLD is also called a System or macro-level design.	LLD is also called details or micro-level design.
5.	Developed by	Solution Architect prepares the High-Level Design.	Designer and developer prepare the Low-Level Design.
6.	Sequential order in the design phase	It is developed first in sequential order, which implies that the HLD is created before the LLD.	It is developed after High-level design.
7.	Target Audience	It is used by management, program, and solution teams.	It is used by designers, operation teams, and implementers.
8.	Conversion	The HLD changes the client or business requirement into a high-level solution.	The LLD changes the high-level solution to a comprehensive solution.
9.	Probable output	The high-level design is necessary to understand the flow across several system objects.	A low-level design is required for creating the configurations and troubleshooting inputs.
10.	Input Criteria	The input measure in high-level design is SRS (Software Requirement Specification).	The input measure in low-level design is the reviewed HLD (High- Level Design).
11.	Output Criteria	The output measures in the HLD are functional design, database design, and review record.	The output bases in the low-level design are the unit test plan and program specification.

Conclusion

In this section, we have understood the major difference between **high-level and low-level designs**.

And we can conclude that the **high-level design** specifies the complete report and planning of the particular software product or application. On the other hand, the **low-level design** specifies the in-detail report of all the modules.

In **Software Development Life Cycle's design phase**, the members of design team, client team, and review teams are included in high-level designing. On the other hand, the design team and operation teams will prepare the low-level Design.

Finally, we can say that both **HLD and LLD** are essential part of design phase in the SDLC process for any software product.

BRS vs SRS

In this section, we are going to discuss the difference between BRS and SRS and see a brief introduction between them.

The **BRS and SRS** are the most important documents to develop any project or software. These types of the document contain the in-depth details of the particular software.

In [software testing](#), BRS and SRS types of document requirements depend upon business type, their standards, how company processes, and what class of software is to be developed.

Before we understood the difference between BRS and SRS, we will look into the difference between **requirement and specification**.

Requirement vs Specification

In the below table, we have made a comparison between requirement and specification.



Requirement	Specification
They plan the software from the end-user, business, and stakeholder perspectives.	They prepare the software from the technical team's point of view.
The requirements define what the software must do.	The specification defines how the software will be developed.
Some of the common terminologies used for requirement document are as follows: <ul style="list-style-type: none">◦ SRD: System Requirement Document◦ BRD: Business Requirement Document	Some of the common terminologies used for specification document are as follows: <ul style="list-style-type: none">◦ FRS: Functional Requirement Specifications◦ SRS: System Requirement Specifications◦ CRS: Configurations Requirements Specification◦ PRS: Performance Requirements Specifications◦ RRS: Reliability Requirements Specifications◦ CRS: Compatibility Requirements Specifications

Now, let's see a brief introduction to **BRS and SRS** documents.

What is BRS?

The BRS document stands for **Business Requirement Specification**. To create the BRS document, the **Business analyst** will interrelate with the customers. The BRS document includes the business rules, the project's scope, and in-detail client's requirements.

In this type of document, the client describes how their business works or the software they need.

For the CRS, the details will be written in the simple business (English) language by the BA (business analyst), which developers and the test engineers cannot understand.

What is SRS?

The SRS document stands for **Software Requirement Specification**.

In this document, the Business Analyst will collect the Customer Requirement Specifications (CRS) from the client and translate them into Software Requirement Specification (SRS).

The SRS contains how the software should be developed and given by the Business Analyst (BA).

In other words, we can say that the SRS document is used to convert the customer information into a detailed document, which can easily be understood by the developers and the test engineers.

Characteristic of Software Requirement Specification

Some important features of the SRS document are as below:

- The SRS document is used to determine the early cost of the software product.
- It is helpful to link the gap between the developer and the user.
- Software requirement specification works as an agreement between communicating parties.

The Key difference between BRS and SRS documents

The below facts explain the vital differences between **BRS** and **SRS** documents:

- **SRS** represented as **System Requirement Specification** while **BRS** represented as **Business Requirement Specification**.
- SRS defines the **functional and non-functional** needs of the software; on the other hand, **BRS** is a formal document, which specifies the needs given by the customer.
- The **SRS** document is developed by the **SA (System Architect)**; on the other hand, the **BRS** is generally developed by the **BA (Business Analyst)**.
- **SRS** is obtained from the **BRS**, whereas BRS is acquired from customer statements and their business needs.

Difference between BRS vs SRS



The below comparison table enlightens their significant differences between SRS and BRS in a quick manner:

S.NO.	BRS (Business Requirement Specification)	SRS (Software Requirement Specification)
1.	It is a document that describes the requirements of the client using the non-technical expression.	It determines the specifications of a software product more formally.
2.	It prepares the report of the user connections.	It specifies how the clients communicate with the system with the help of use cases.

3.	In the BRS document, it is not important to contain the references of figures and tables.	It always includes references to illustrations and tables.
4.	The BRS document is obtained from relating to the client's requirements and taking responsibility for them.	The Software Requirement Specification obtain from the Business Requirement specification.
5.	The BRS document involved the product's future scope, keeping in mind the organization's strategies for development plans.	The SRS document does not involve the scope of the product.
6.	In the BRS document, all the client's requirements complete in all the details are together and accessible.	SRS document describes the stepwise sequence of all the operations characteristics for each module and sub-modules.
7.	The BRS Document is prepared by a team of Business Analysts (BA) who interacts with the clients.	The SRS document is created by a team of System Analysts (SA), a technical expert.
8.	It defines, at a very high level, which includes the functional conditions of the application.	It specifies at a high level where the functional and technical requirements of the application are included.
9.	The BRS documents cover all types of requirements.	The SRS documents cover all functional and non-functional requirements.
10.	The BRS document lists the user base along with similar stakeholders from the client-side.	The SRS document doesn't list anyone from the client or user base.

Conclusion

In this section, we have made a comparison between BRS and SRS documents.

Finally, we can conclude that the programmers used both the documents in the development process and for the testing process.

The **Business Requirement Specification (BRS)** is a formal document that specifics the customer's requirement, written or verbal.

Simultaneously, the **Software Requirement Specification (SRS)** document defines the functional and non-functional needs of the software to be established.

The difference between Positive Testing and Negative Testing

The primary objective of performing the software testing is to identify the **bugs** in the code and enhance the quality of software or an application.

To achieve this, we have two different but unique software testing approaches, such as positive testing and negative testing.

Both of these testing strategies have their prime features and functionality, and both testings play a significant role to check the software or an application.

To test any kind of application, we need to provide some input value and validate if the results are based on the given requirements or not.

Before understanding the significant difference between **positive and negative testing**, we will briefly introduce positive and negative testings with some examples.

What is Positive Testing?

It is used to check whether our application works as expected or not. And if an error is detected at the time of positive testing, the test is considered as fail. A positive testing is a technique whenever a test engineer writes the test cases for a set of respective outputs.

In **positive testing**, the test engineer will always check for only a good set of values.

In other words, we can say that **positive testing** is a process where the system or an application is tested against the valid input data.

And the primary purpose of performing the positive testing is to validate whether the software does what it is supposed to do.

In simple terms, we can say that positive testing is implemented by providing a **positive point of view**.

For example, **Numbers like 9999999**.

Enter Only Numbers

9999999

Positive Testing

Example of Positive Testing

In the following example, we are trying to understand the working of positive testing.

- Suppose, we have one test scenario, where we want to test an application that includes a simple text box.
- To enter **Phone Number** according to the business needs it accepts only numerical values.
- Hence, in positive testing, we will only give the positive numerical values in order to test whether it is working as per the requirement or not.

Note: Generally, most application developers execute the Positive scenarios where the test engineer gets less bug count around positive testing.

To get in-detail information about positive testing, refers to the following link:

What is Negative Testing?

It is implemented to check how the application can gracefully handle invalid input or unpredicted user performance.

The fundamental purpose of executing the negative testing is to ensure the application's stability against the effects of different variations of improper validation data set.

Negative testing is also known as **error path testing or failure**. And it helps us to identify more bugs and enhance the quality of the software application under test.

Once the positive testing is complete, then only we can execute the negative testing, which helps to identify more bugs and enhance the quality of the software application under test.

We can say that the negative testing is executing by keeping the **negative point of view** in simple terms. For example, **99999abcde**

Enter Only Numbers

99999abcde

Negative Testing

Example of Negative Testing?

Let see one example which will help us to understand the negative testing in an effective way.

- Suppose we have one sample form where the **Mobile Number** field can only accept the numbers' value and does not accept the unique characters and alphabets values.
- However, let's entered the unique characters and alphabets values on **the Mobile number** field to test that it accepts the individual characters and alphabets values or not.
- We expect that the textbox field will not accept invalid values and show an error message for the incorrect entry.

To get in-detail information about negative testing refers to the following link:

Example for Positive and Negative testing

The positive method is the same to the negative testing. But the significant difference between them is, in positive testing, we will enter the valid data instead of false data, and the expected result is the system accepting the code with no further issues.

Let's see another example where we will see the Positive and negative test scenarios:

Suppose we have one requirement, where the **Password text box field** should accept only 8-15 alphanumeric characters.

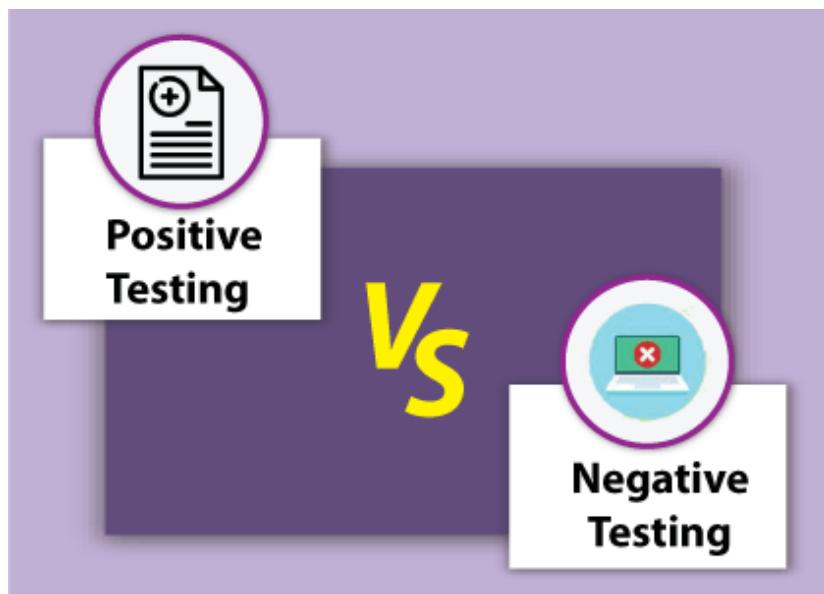
Let see the below table where we discussed some of the positive and negative test scenarios.

Positive Test Scenarios	Negative Test Scenarios
The Password text box should accept 8 characters.	The Password text box should not accept less than 8 characters.
The Password text box should accept 15 characters.	The Password text box should not go beyond more than 15 characters.
The Password text box should accept any value between 8-15 characters long.	The special characters should not accept by the password text box.

The Password text box should accept all numeric and alphabet values.

Positive Testing VS Negative Testing

Some of the significant difference between positive and negative testing is discussed in the following table:



S.No.	Positive Testing	Negative Testing
1.	Checking the application response with the help of valid input data is known as positive testing.	Checking the application response by using the invalid input data set is known as negative testing.
2.	Positive testing is implemented only for the expected conditions.	Negative testing is implemented only for unexpected conditions.
3.	Positive testing doesn't guarantee a good quality of software product.	Negative testing guarantees to deliver a good quality of software product.
4.	The execution of positive testing takes less time as compared to negative testing.	The execution of positive testing takes more time as compared to negative testing.
5.	To validate the available set of test conditions, we will consistently implement the Positive testing.	To break the project and product with an unidentified set of test conditions, we will consistently implement Negative testing.
6.	The primary purpose of executing Positive testing is to guarantee that the software application always meets the developer's requirements and specifications.	The primary purpose of executing the negative testing is to test a web application's constancy in contradiction to inaccurate validation data sets.
7.	Positive testing doesn't encompass all the possible cases.	Negative testing encompasses all the possible cases.
8.	It is a process where the system is validated in contradiction of the valid input data.	It is a testing process which contains the validation in contrast to invalid input data.
9.	Positive testing is less significant than Negative testing.	Negative testing is more vital than Positive testing.
10.	Positive testing can be implemented on every application.	Negative testing can be implemented when the possibilities of unpredicted conditions.
11.	The people having less knowledge can execute the positive testing.	The testing professionals can execute the negative testing.

12.	It makes sure that the software is standard.	Negative testing makes sure to deliver 100 percent bug-free software.
-----	--	---

Conclusion

After seeing the **significant comparison between positive and negative testing**, we can conclude that testing helps us to increase the software application quality and ensures the software is bug-free before the software is launched.

As the name recommends, they are two opposing techniques but highly effective in getting quality software at the end of the day.

The fundamental purpose of implementing the Positive and Negative Testing is to verify that the application works as per the given business needs and specifications and provides us enough assurance in the software's quality.

And finally, we can say that Positive testing and Negative testing are like two sides of a coin, which implies that we cannot overlook anyone's significance in any case.

If both are implemented, the output is overwhelming; if we ignored one of them, the final product would be unpleasant.

Difference between Top-Down and Bottom-Up Integration Testing

In this section, we are going to discuss the difference between **top-down and bottom-up integration testing**; and see a brief introduction of them.

As we understood in the earlier section of software testing, every software or application encompasses various modules that are contrary to each other through an interface.

When each component or module works independently of an application, we need to check the dependent modules' data flow, known as **integration testing**. It is a significant part of Functional testing.

Before we see the top-down and bottom-up integration testing approaches in detail, we need to understand **incremental integration testing** as top-down and bottom-up integration testing is an integral part of it.

Incremental Integration Testing Approach

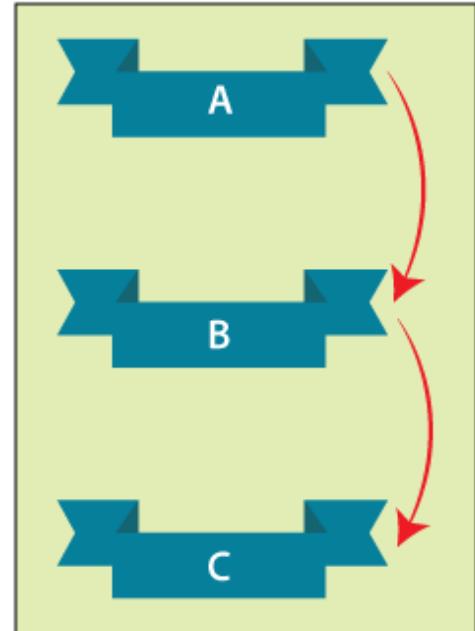
- **Incremental testing** is the most vital part of integration testing. The modules are added in ascending order one by one as per the customer's need. And the selected modules need to be related logically to each other.
- Usually, two or more modules are added and tested to fix the precision of functions. And, the process will continue until all the modules or components are tested successfully.
- In simple words, we can say that when there is a strong relationship between the dependent modules, we will perform the **incremental integration testing**.

Now, let look into the definition and basic working of **top-down and bottom-up incremental integration testing**.

What is Top-down Integration Testing?

- In top-down incremental integration testing, we will add the modules incrementally or one by one and test

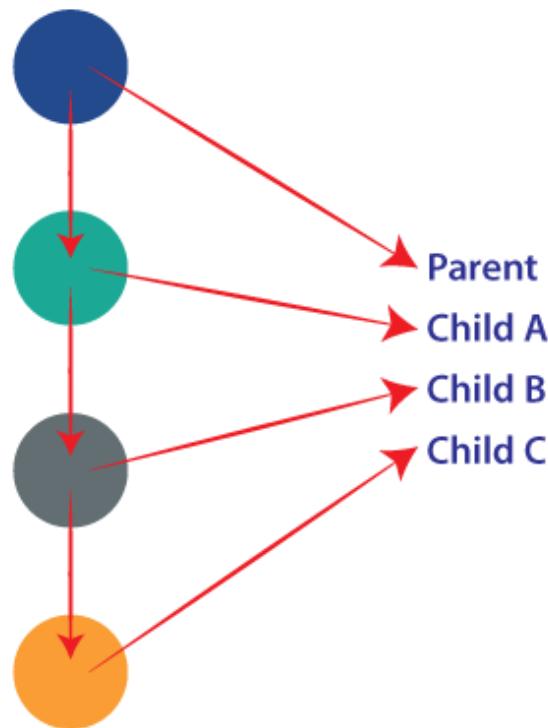
Top-Down Approach



the data flow in similar order as we can see in the below diagram:

- This testing technique deals with how higher-level modules are tested with lower-level modules until all the modules have been tested successfully.

- In the top-down method, we will also make sure that the module we are adding is the **child of the previous one**, like Child C, is a child of Child B.



one, like Child C, is a child of Child B.

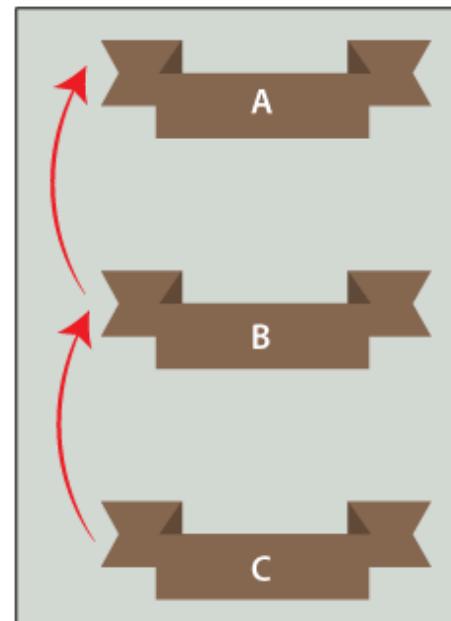
- The purpose of executing top-down integration testing is to detect the significant design flaws and fix them early because required modules are tested first.

What is Bottom Up Integration Testing?

The next testing approach we are talking about is **bottom-up integration testing**.

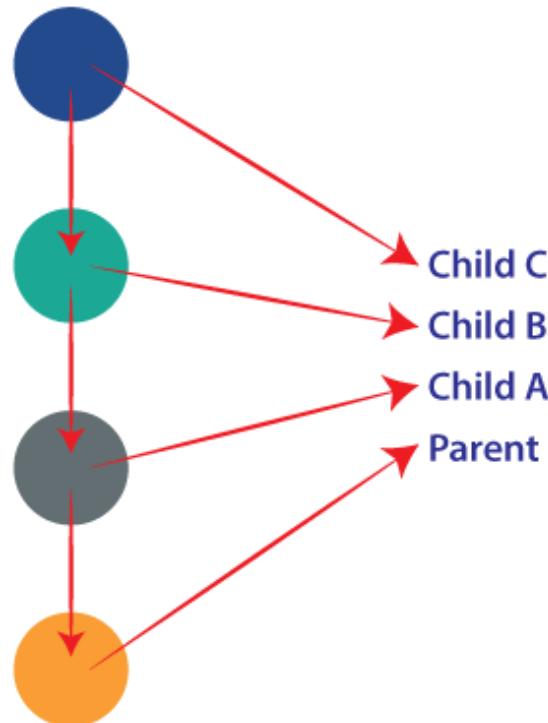
- This type of testing method deals with how lower-level modules are tested with higher-level modules until all the modules have been tested successfully.
- In bottom-up testing, the top-level critical modules are tested, at last. Hence it may cause a defect.
- In simple words, we can say that we will be adding the modules from **the bottom to the top** and test the

Bottom-up Approach



data flow in similar order as we can see in the below image:

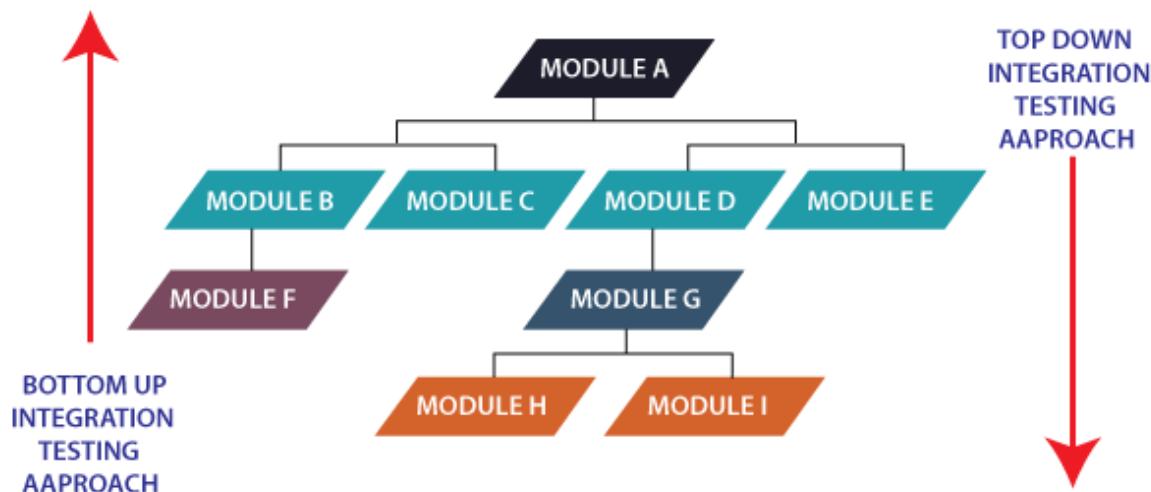
- In the bottom-up method, we will ensure that the modules we are adding **are the parent of the previous one**



as we observe in the following image:

Key difference between top-down and bottom-up incremental integration testing

The below facts explain the critical differences between **top-down and bottom-up integration testing**, which will allow test engineers to create an informed decision regarding which type of integration testing approach they want to select for the different testing processes.



1. The **top-down integration testing** approach is simple and not data-intensive; on the other hand, the **bottom-up integration testing** approach is complex and data-intensive.
2. The process of top-down integration testing is much simpler as compared to bottom-up integration testing.
3. Top-down approaches are **backward-looking**; on the other hand, the bottom-up approaches are **forward-looking**.
4. The top-down integration testing works through **significant to minor components**, whereas the bottom-up approach works through **small to essential components** or modules.
5. Top-down approach analyses the risk by collecting the impact of internal operational failures, whereas the bottom-up approach analyses the risks in individual processes with models' help.
6. In the top-down approach, the stubs are used to simulate the submodule, which implies that the Stub works as a momentary replacement. On the other hand, in the bottom-up testing approach, the drivers simulate the main module, which means that the Driver works as a momentary replacement.

Top-Down Integration Testing VS Bottom-Up Integration Testing

We have discussed some significant comparisons between **Top-Down Integration Testing** and **Bottom-Up Integration Testing** in the below table:

Top-Down Integration Testing

Bottom-Up Integration Testing

VS

S.NO.	Comparison Basis	Top-Down Integration Testing	Bottom-up Integration Testing
1.	Definition	We will add the modules incrementally or one by one and test the data flow in similar order.	The lower-level modules are tested with higher-level modules until all the modules have been tested successfully.
2.	Executed on	The top-down integration testing approach will be executed on the Structure or procedure-oriented programming languages.	The bottom-up integration testing approach will be executed on Object-oriented programming languages.
3.	Observation	In the top-down approach, the observation of test output is more complicated.	In the bottom-up approach, the observation of test output is more accessible.
4.	Risk analysis	We are collaborating on the impact of internal operational failures.	To analyze the individual process, we can use the Models.
5.	Work on	The top-down integration testing approach will work on major to minor components.	The bottom-down integration testing approach will work on the minor to significant components.
6.	Complexity	The complexity of the top-down approach is simple.	The complexity of the bottom-up approach is complex and highly data intensive.
7.	Stub/Driver creation	Stub modules must be created in the top-down testing approach.	The driver modules must be created in the bottom-up testing approach.
8.	Managed from	It is performed from main-module to sub-module.	It is performed from the sub-module to the main module.
9.	Advantage	<p>Following are some of the significant benefits of using top-down integration testing:</p> <ul style="list-style-type: none">◦ In this, the early prototype is possible.◦ Fault Localization is easier.	<p>Below are some of the essential advantages of using bottom-up integration testing:</p> <ul style="list-style-type: none">◦ We do not need to wait for the development of all the modules as it saves time.◦ Identification of defects is easy.

10.	Disadvantage	<p>Some of the most common drawbacks of the top-down approach are as follows:</p> <ul style="list-style-type: none"> ◦ Lower-level modules are tested ineffectively. ◦ Due to the high number of stubs, it gets pretty complicated. ◦ Critical Modules are tested first so that fewer chances of defects. 	<p>The disadvantage of the bottom-up approach is as follows:</p> <ul style="list-style-type: none"> ◦ Compulsory modules are tested last, due to which the defects can occur. ◦ There is no possibility of an early prototype.
-----	---------------------	--	--

Conclusion

In this tutorial, we have made a comparison between **top-down and bottom-up incremental integration testing**.

Here, we have concluded that the execution of top-down and bottom-up approaches is required to test the software or the application.

The top-down testing technique is the most commonly used **Integration Testing Type**. It is an integration testing technique used to imitate the lower-level modules that are not yet integrated.

The bottom-Up testing technique is a type of incremental integration testing approach implemented to evaluate the risks in the software. The most significant advantage of executing this approach is its user-friendliness which provides high deployment coverage in software development.

In both approaches, **top-down and bottom-up integration testing**, the top-down generates more redundant results and leads to additional efforts in the form of overheads. Equally, the bottom-up approach is challenging but more efficient than the top-down approach.

Difference Between Use Case and Test Case

In this section, we are going to discuss the difference between **Use case and Test case**; and see a brief introduction of them.

In the **software testing** domain, the terms **Use case and Test case** are the most important and closely related but intensively different.

Now, let look into the definition and basic working of **Use case and test case**.

What is Use Case?

- In software testing, a use case is a graphic representation of business needs explaining how the end-user will cooperate with software or an application. And the use cases allow us all the possible techniques of how the end-user uses the application.
- In simple words, we can express that with the help of use cases; we can define how to use the system for executing a precise task.
- The use case is not a part of execution which means that it is only a graphic demonstration of a document that explains how to implement a particular task.
- With the help of the use case, we get to know how the product should work.

For more information about the use case technique, refers to the below link: <https://www.javatpoint.com/use-case-technique-in-black-box-testing>.

What is Test Case?

- A **test case** is defined as a group of conditions under a test engineer concludes whether a software application is working as per the customer's requirements or not.
- The test case designing includes **preconditions, case name, input conditions, and expected result**.
- These are derived from the test scenarios, and it is a first-level action.
- Mainly, test cases are used to validate the developed software by test engineers that the particular software is working as per requirement or not.
- A **test Case** is described as a group of various testing activities like **test inputs, execution conditions, and expected results** that additionally lead to evolving a particular test objective.
- Writing test cases is a one-time attempt that can be used in the future at the time of regression testing.

For more information about the test case technique, refers to the below link: <https://www.javatpoint.com/test-case>.

The key difference between Use case and Test Case

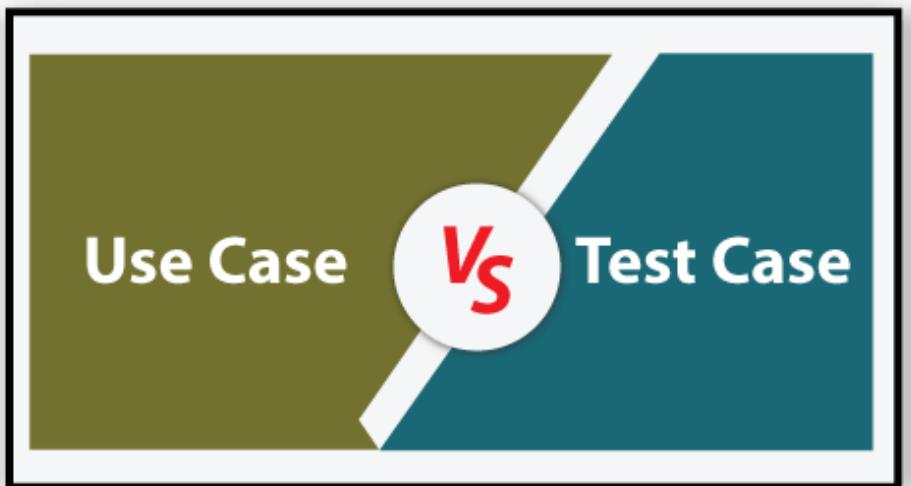
The below facts explain the key differences between **Use case and the test case**.

- The **use case** cannot be implemented, which implies **that it is only designed**. And on the other hand, the **test case is designed, and later we implemented them**.
- The **use case** is obtained from the **BRS (Business requirement specification)** whereas the **test case** is derived from the **use case**.
- The **use case** is a **graphic representation of client requirements**, whereas the **test case** is not **represented diagrammatically**; it is only documented in an excel sheet.
- The **use case** is a document that always describes **the flow of events of an application**. In contrast, the **test case** is a document that always **contains an action, event, and expected output of a particular feature of an application**.
- The **use case relies on the software requirements**; on the other hand, the **test case depends on the use case**.

- The **use case** collects the requirements, and on the other hand, the **test case** will analyze those requirements.
- In **use case**, the **results are not verified**. On the contrary, the **test case** results are verified, which means that the test case **checks if the result shows in use cases is functioning right**.

Use case VS Test Case

We have discussed some significant comparisons between **Use Case** and **Test Case** in the below table.



S.NO.	Comparison Basis	Use case	Test case
1.	Definition	It is a graphical representation of the software and its multiple features.	It is an in-details document containing all possible inputs (positive and negative) and the navigation steps used for the test execution process.
2.	Manged by	The diagrams manage the use case .	The function tests manage the test case .
3.	Source	These are prepared on requirements.	These are prepared on Use cases.
4.	Required	Documents and research are required in order to create a use case.	It requires preconditions, case names, input conditions, and expected results in order to create test cases.
5.	User	The Business user executes the use case.	The test engineers execute test cases.
6.	Purpose	The primary purpose of the use case is to reach the last operation that follow all sequential processes.	The purpose of the test case is to validate the software is working fine or not.
7.	Completion	The Use case will complete all steps at once.	In a test case, the testing is performed again and again in order to end the process.
8.	Output	In the use case, the result is important, and all steps are to be implemented together.	In a test case, all steps are significant and may have a different result.
9.	Iteration	The use case supports the different paths.	The test case supports a single test case.

10.	Designed by	The business analyst designs the use case.	The test engineers design the test case.
11.	Interaction with	The use case interacts with the users.	The test case is depending on the results.
12.	Works by	The use case is the procedure of following the step-by-step function capability of the software.	The use case steps can be followed while designing the test cases.
13.	Advantages	<p>Some of the significant advantages of using the use case technique while we are developing the product are as follows:</p> <ul style="list-style-type: none"> ○ The use case is the classification of steps that explain the connections between the user and its actions. ○ It is used to take the functional needs of the system. ○ It begins from an elementary view where the system is created first and primarily used for its users. ○ It is used to control the complete analyses. 	<p>If we properly designed a test case, we will get the following advantages:</p> <ul style="list-style-type: none"> ○ We can utilize the organizational resources more efficiently. ○ The creation of a test case makes sure to developed and delivered a higher quality project. ○ The identification of the bugs is easier and highly possible. ○ The testing schedule and project budgets can be strictly followed.

Conclusion

In this tutorial, we have seen the major difference between the **use case and the test case**.

In the software industry, both **Use cases and test cases** are used interchangeably and have different meanings.

And after seeing all the differences between use case and test case, we can conclude that both **use case and test case** are equally important in the various phases of the **Software development life cycle**.

The Use cases are mainly used in the **requirement and design phases** of the SDLC, which lead the development in the precise direction. On the other hand, the test cases are mainly used in the **testing phase** of the **SDLC** where the precise execution is displayed, and failures in the software are distinguished.

Monkey Testing VS Gorilla Testing

In this section, we are going to discuss the difference between **Monkey Testing** and **Gorilla Testing**; and see a brief introduction of them.

The two different types of **software testing** which is known as **Monkey Testing and Gorilla Testing**. These types of testing are conducted on a piece of software before released in the market.

Before getting into the difference between a monkey and gorilla testing, we will briefly introduce **Monkey** and **Gorilla** testing.

What is Monkey Testing?

- In software testing, we have some thorough testings, and Monkey testing is one of them.
- Checking the performance of the software or an application based on some random inputs without any test cases and verifies whether it crashes or not is known as **monkey testing**.
- It is also known as **Fuzz Testing, Random Testing, and Stochastic Testing**.
- We cannot use the test case while performing the monkey testing as it is a form of random testing.
- In this testing, an **end-user or a test engineer, or a developer** can test the system to verify whether it completes the preferred set of actions.

What is Gorilla Testing?

- Another special type of **software testing** is **Gorilla testing**.
- Testing all the software modules of an application to its edge is known as Gorilla testing.
- Gorilla testing is also known as **Frustrating testing, torture testing, and fault tolerance testing**.
- In simple words, we can say that each minor code of the software is tested until it begins to fall separately or fails to provide the preferred outcome.
- To test a module's functionalities constantly the test engineers and developers works together in Gorilla Testing.

Difference between Monkey Testing and Gorilla Testing

We have discussed some significant comparisons between Monkey Testing and Gorilla Testing in the below table.



S.NO.	Monkey Testing	Gorilla Testing
1.	In Monkey Testing, no test case is used to test the application as it is a part of random testing.	It is performed repeatedly as it is a part of manual testing.

2.	The Monkey Testing approach is primarily used in System Testing.	The Gorilla Testing approach is mainly used in Unit Testing.
3.	It is a particular type of Software Testing that is implemented based on specific random inputs without any test cases and tests the system's performance and validates whether it fails or not.	, tests the modules' functionalities, and validates that no bugs are present in the particular module.
4.	It further calls it Fuzz Testing, Random Testing, and Stochastic Testing.	It further calls it Fault Tolerance Testing, Torture Testing, and Frustrating Testing.
5.	Monkey testing is implemented on a whole system.	Gorilla testing is implemented on few selective components of the system.
6.	Monkey Testing can be executed by the end-user, test engineer, and developer in order to test the system and validate whether it execute the desired set of activities.	Generally, the Gorilla Testing is executed by the test engineers and developers either together or separately in order to check a component's features repeatedly.
7.	No software knowledge is required in order to execute the monkey testing.	It requires minimum software knowledge in order to execute the gorilla testing.
8.	The essential purpose of executing the monkey testing is to test whether the system crashes or not.	The fundamental purpose of executing the gorilla testing is to verify whether the component or module is working appropriately or not.
9.	It inhibits the system failure once the monkey testing is executed.	On the other hand, the execution of gorilla testing checks the ability of single module features.
10.	Monkey Testing is divided into three types of testing, namely, Dumb Monkey Testing, Smart Monkey Testing, and Brilliant Monkey Testing.	As compared to monkey testing, gorilla testing did not divide into different types of testing.
11.	The implementation of Monkey testing does not require any planning or preparation.	The Gorilla testing cannot implement without any preparation or planning.

Conclusion

In the above article, we have discussed the critical difference between **Monkey testing and Gorilla testing**.

And after seeing all the essential differences, we can conclude both the testing are the same because of the emphasis on randomly testing a given software under test.

Therefore, all the possible area is tested in contradiction of the requirement specifications.

Both the **Monkey and Gorilla** testing's approaches are action-centered software testing strategies in order to break the application or the software under test.

Further, we can say that both Monkey and Gorilla Testing are other strict and precise testing procedures.

Difference Between Stubs and Drivers

In this section, we are going to discuss the **difference between Stubs and Drivers**; and see a brief introduction of them.

Stubs and drivers are two types of test harness that are a collection of software and test, which means they are designed together to test a unit of a program by accelerating the change of situations at the same time as regularly checking its performances and results.

In the Software Testing process, the Stubs and Drivers are those components that are used as a short-term alternative for a module.

There are several elements in the Software Testing Life Cycle, which play an essential role in the testing process to be more precise and trouble-free.

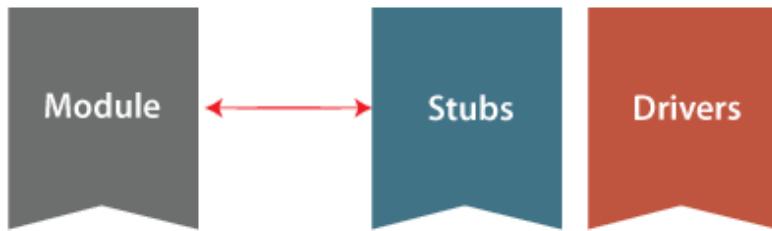
All the components connected to testing and try to enhance their quality, helps us to deliver an exact and predictable outcome, and facilities the fulfilment of the specified specifications.

Describe Stubs and Drivers in the Software Testing?

In **Software Testing**, the words **stub and drivers** described as a replica of the modules that operate as an alternative to the new or missing modules.

Stubs are mainly used in **top-down integration testing**; on the other hand, drivers are mainly used in **bottom-up integration testing** individually and designed to enhance the testing process.

To sustain the essential requirements of the inaccessible modules or components, we are precisely established the stubs and drivers. And extremely beneficial in getting the anticipated outcomes.



Both Stubs and drivers are the essential part of the basic software development and software testing process. Thus, to help us understand the substance of stubs and drivers in software testing, we will see an exhaustive discussion.

What are Stubs?

- **A stub** is a replica of a module that collects the data and develops many possible data. However, it executes like an actual module and is mainly used to test modules.
- Generally, stubs are created by software developers in order to use them instead of modules if the particular modules are miss or not yet developed.
- By using these test stubs, the test engineers can simulate the performance of the lower-level modules, which are not yet joined with the software. Furthermore, it helps us to accelerate the activity of the missing modules.

Types of Stubs

In the **top-down approach of incremental** integration testing, the stubs are divided into four essential parts, which are as follows:

- Demonstrate the trace message.
- Exhibits the parameter values.
- Returns the consistent values, which are handled by the modules or the components.

- Returns the values of the specific parameters, which were utilized by testing components or modules.

What are Drivers?

- The **Drivers** establish the test environments and takes care of the communication, estimates results, and also sends the reports.
- These are just like stubs and used by software test engineers in order to accomplish the missing or incomplete modules/ components requirements.
- The drivers are mainly developed in the **Bottom-up approach of incremental integration testing**.
- Generally, drivers are bit complex as compared to the stubs.
- These can test the lower levels of the code when the upper-level modules or codes are not developed or missing.
- In other words, we can say that the Drivers perform as **pseudo-codes**, which are mainly used when the stub modules are completed; however, the initial modules/components are not prepared.

Examples of Stubs and Drivers

Let us see an example of stubs and drivers, which help us to enhance our knowledge of stubs and drivers.

Suppose we have one web application that contains **four different modules**, such as:

- **Module-P**
- **Module-Q**
- **Module-R**
- **Module-S**

And all the modules, as mentioned earlier, are responsible for some individual activities or functionality, as we can observe in the following table:

Different Modules	Individual Activities
Module-P	Login page of the web application
Module-Q	Home-page of the web application
Module-R	Print Setup
Module-S	Log out page

Note: Modules P, Q, R, and S encompasses the dependencies of each module over the other.

It is always a better approach to implement the testing or development of all the modules equivalently. The minute each gets developed, they can be combined and tested according to their similar dependencies with a module.

Once **Module-P** is developed, it will go through the testing process. But, to perform and validate the testing methods regarding **Module-P**, they need **Module-Q**, which is not yet developed entirely and still in the developing process.

And it is not possible to test **Module-P** on the lack of **Module-Q**. Thus, in such scenarios, we will take the help of **Stubs and Drivers** in the **software testing process**.

The Stubs and drivers will replicate all the basic functionality and features displayed by the real **Module-Q**. And subsequently, it is being combined with **Module-P** in order to execute the testing process effectively.

Now, we can validate the estimated functionality of the **Login page**, which was in **Module-P**, only if it is going to the **Home Page**, which is the activity of **Module-Q** as per the correct and valid inputs.

In the same way, **stubs and drivers** are used to accomplish the requirements of other modules, such as the **Sign-out page**, which comes under the Module-S and needs to be aimed to the **Login page (Module-P)** after effectively logging out from the particular application.

Likewise, we may also use Stubs or Drivers instead of **Module-R** and **Module-S** if they are not available.

As the result of the inaccessibility of **Module-P**, stubs and drivers will work as an alternate for it to perform the testing of **Module-S**.

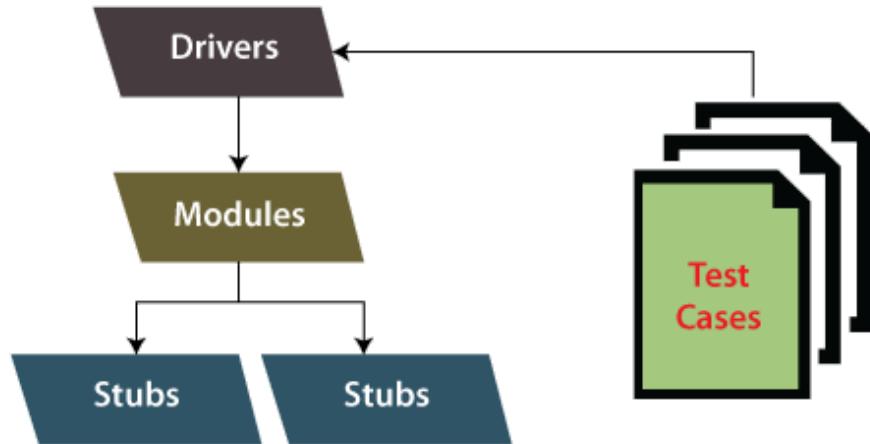
But here, the question arises that both drivers and Stubs serve the same functionality?

Let's find the answer to the above question:

Yes, we can say that both stubs and drivers carry out similar features and objectives. Both of them act as an alternative for the missing or absent module. But, the change between them can be pictured throughout the integration testing process.

The key difference between Stubs and Drivers

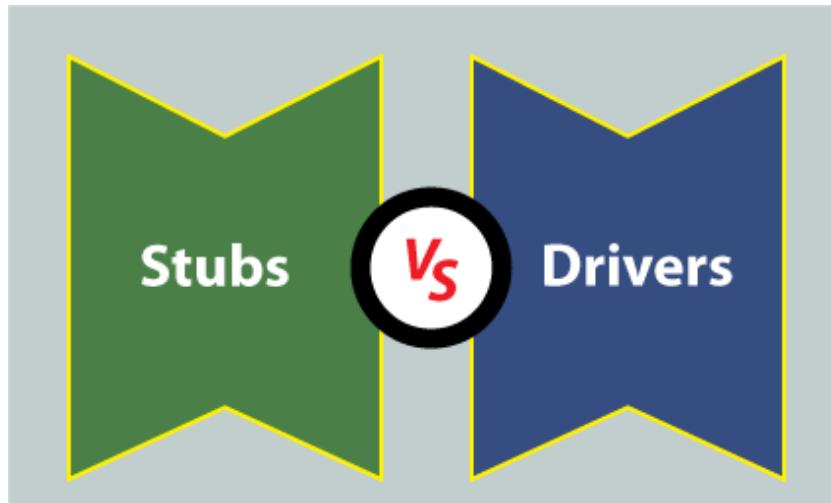
The below facts explain the critical differences between **stubs and drivers** during the integration testing process.



- **Stubs and Drivers** design as a dummy for the missing or inaccessible modules or the components.
- Most commonly, stubs and drivers are used in the **incremental integration**, where stubs are used in **top-bottom methodology** while drivers are used in a **bottom-up methodology**.
- Usually, developers and unit test engineers are included in the creation of stubs and drivers.
- Even though it provides an ease to perform the **separate components or the modules** without worrying about the accessibility of other modules, and leads them into a time-consuming process, as it involves the development of replica for all the missing modules.
- Precisely, stubs and drivers are developed for each module holding different purposes.

Stubs VS Drivers

Here, we are discussing some significant comparisons between **Stubs and Drivers** in the below table:



S.NO.	Stubs	Drivers
1.	A section of code that imitates the called function is known as Stubs.	A section of code that imitates the calling function is known as Drivers.
2.	It is used to test the functionality of modules and test modules and also replicate the performance of the lower-level module which are not yet merged, and the activity of the missing module/components.	When the main module is prepared or ready, we will take the help of drivers. Generally, drivers are a bit more complex as compared to the stubs.
3.	The stubs are developed during the Top-down approach of incremental integration testing.	The drivers are developed during the bottom-up approach of incremental integration testing.
4.	Stubs replicate the activity of not developed and missing modules or components.	Drivers authorizes test cases to another system and which refer the modules under testing.
5.	The stubs are created by the team of test engineers.	Mostly, the drivers are created by the developer and the unit Test engineers.
6.	Stubs are developed when high-level modules are tested, and lower-level modules are not formed.	Drivers are acquired when lower-level modules are tested, and higher-level modules are not yet developed.
7.	These are parallel to the modules of the software, which are under development process.	On the other hand, the drivers are used to reminding the component, which needs to be tested.
8.	The stubs signify the low-level modules.	The drivers signify the high-level modules.
9.	Fundamentally, the Stubs are also known as a called program and initially used in the Top-down integration testing.	The Drivers are also known as the calling program and are mainly used in bottom-up integration testing.
10.	These are reserved for testing the feature and functionality of the modules.	The drivers are used if the core module of the software isn't established for testing.

Conclusion

In this section, we have seen the significant difference between Stubs and Drivers.

And we conclude that the software testing process cannot be implemented with the unfinished and partial modules and components.

Hence, to ensure the correctness and efficiency of testing, it is necessary to develop the **stubs and drivers** that satisfy the requirements of unfinished modules.

And perform as simulated modules which are requested for testing the features of main modules or components.

Lastly, we can say that the stubs and drivers are a vital part of the software testing process.

They are computer programs that work as an alternative and replicate the functionalities of the other modules, which help us simplify the software testing activities.

The Difference between Component Testing and Unit Testing

In this section, we are going to discuss the difference between **component testing** and **Unit testing** based on the various parameters.

As we understood in the earlier section of **software testing**, all types of software testing have their features and functionality in order to test the application or software.

Before getting into the difference between **component testing** and **unit testing**, we will briefly introduce Component and Unit testing.

What is Component Testing?

It is one of the essential **types of software testing** in which the accessibility of each component is verified.

It can be implemented individually, that is, the separation from the remaining system. However, it has relied on the model of the preferred life cycle.

The execution of components testing makes sure that all the application component is working correctly and according to the requirements.

In order to execute the component testing, all the components or modules require to be in the individual and manageable state. And all the related components of the software should be user-understandable.

As we can see in the following image of the component testing process:



It is executed by the **developers**, where they should perform component testing before they move on to develop another component.

Once the identification of the defects happens in component testing, either the developer can fix all of them before moving to another component, or he/she can substitute between the fixing and development alternatively.

For more information about **Component Testing**, refers to the following link:

What is Unit Testing?

It is also a part of **software testing** and is used whenever the application is ready and given to the Test engineer. He/she will start checking every module of the application independently or one by one. This process is known as **Unit testing**.

04 Acceptance Testing

03 System Testing

02 Integration Testing

01 Unit Testing

In **unit testing**, the term **unit** is a single testable part of a software system and it is tested during the development stage of the application.

Unit testing is a testing method that tests all the independent modules to verify if there is an issue by the developer himself. It is correlated with the functional correctness of the independent modules.

For more information about Unit Testing, refers to the following link:

<https://www.javatpoint.com/unit-testing>.

The key differences between Component Testing and Unit Testing

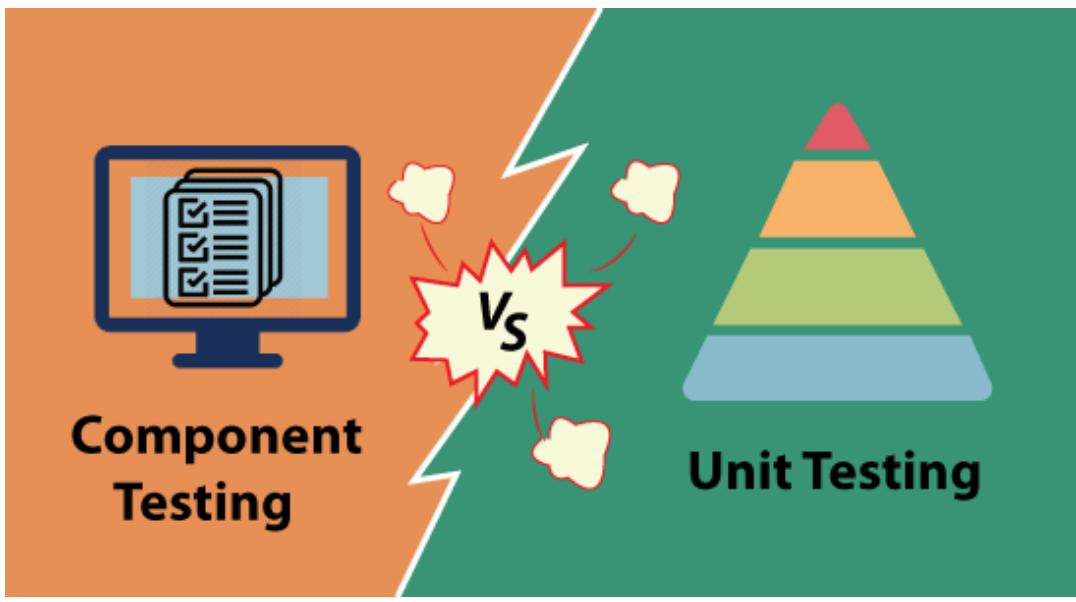
The critical difference between both testings is that the test engineers implement the Component Testing whereas the developers or SDET experts implement the Unit Testing.

Let's understand some other key differences between Component Testing and Unit testing:

- **Component Testing** is executed at the **application level**; on the other hand; **Unit Testing** is executed at a **granular level**.
- **Component Testing** is a type of **black-box testing**, whereas **Unit Testing** is part of the **white box testing**.
- In **Component testing**, all the modules/ components of the related software are checked individually with or without segregation with other objects or modules of the system. On the other hand, **Unit Testing** is tested if a separate program or code is getting performed according to the particular requirements.
- In **Component Testing**, testing is performed by validating use cases and **test requirements**, while in **Unit Testing**, we will test the application in contradiction of the design documents.
- When we performed **Component Testing**, **the test engineer does not become aware of the internal planning of the software**. Whereas, when we performed the **Unit Testing**, **the developers aware of the internal planning of the software**.

Component Testing VS Unit Testing

Let us see the following comparison table to understand the essential differences between **Component Testing** and Unit Testing.



S.NO.	Comparison Basis	Component Testing	Unit Testing
1.	Definition	It includes testing each component/module of the software individually.	It includes testing of separate modules for program implementation.
2.	Validates	Component testing tests the use cases and test specifications.	Unit testing is validating in contradiction of design documents.
3.	Executed by	Component testing executed by the team of test engineers.	The team of Developers executes unit testing.
4.	Performed	Once the Unit testing is successfully implemented, then only we will be performing the Component testing.	Unit testing is implemented before the execution of Component testing.
5.	Internal planning	In Component testing, the test engineer does not know about the internal planning of the application.	In Unit testing, the test engineer is aware of the internal planning or design of the application or the software.
6.	Execution level	Component testing is performed at the application level.	It is performed at a granular level.
7.	Type of	It is a part of black-box testing.	It is part of the white- box testing.
8.	Error Detection	In Component testing, the error detection is a bit complex as compared to unit testing.	In Unit testing, the detection of errors is easy as compared to component testing.
9.	Only implemented	Component testing is only executed when the entire software is developed.	Unit testing is only implemented after every development step.

Conclusion

After seeing all the essential differences between **Component Testing and Unit testing**, we can conclude that component testing is quite similar to unit testing. Still, it is performed at a higher level of integration and in the application setting, not just in the context of that unit/program as in unit testing.

Once the unit testing is implemented successfully, then only we can execute the Component testing.

In **Component Testing**, the **error identification is a bit challenging**; on the other hand, **Unit Testing** is performed after each development process. Therefore, in-unit testing error identification is a bit easy as compared to component testing.

If the Component testing is performed appropriately, then there are fewer bugs in the next stage; thus, it is conducted before unit testing, assessing the programs.

At last, we can say that component testing is essential for finding the errors and bugs to ensure that each component of an application works resourcefully. And it is always suggested to carry out the component test before proceeding with the unit testing.

The differences between Software Testing and Embedded Testing

In this article, we are going to discuss the differences between **Software testing** and **Embedded Testing** based on the various parameters.

Before getting into the comparison between **Software Testing and Embedded testing**, we will briefly introduce the **Software Testing** as well as **Embedded Testing**.

What is Software Testing?

Software testing is a procedure of finding the accuracy of the software by understanding its various qualities. Following are some of vital qualities of software testing:

- **Scalability**
- **Reliability**
- **Re-usability**
- **Portability**
- **Usability**

Testing is all about evaluating software modules/components' performance in order to identify the software bugs or errors, or defects.

It is a mandatory process to implement on any application or software as it will be a risky situation if the software collapses any time due to the absence of testing. Therefore, without performing the testing process on the software cannot be deployed to the end-user.

The software testing technique is also sharing information about the quality of the software with the clients. Furthermore, we can say that the execution of **Software testing** gives an impartial view of the software and also provides the security of the software ability.

The software testing process includes the analysis of all components or the modules under the vital services to validate whether it fulfils the particular requirements.

It also involves an examination of code in several environments as well as all the testing features of the code.

What is Embedded Testing?

To verify and validate both software and hardware performance, we will use the **Embedded testing** process. The execution of the embedded testing process makes sure that the specified entire system is bug/defect-free along with its software and hardware.

The added advantage of embedded testing is that it can be executed on the hardware in order to find the bugs/ defect.

With the help of embedded testing, we can document the system's progress, and it also helps us to guarantee that the specified system meets the client specification or the requirements.

Software Testing VS Embedded Testing

Let's see the key difference between **software testing** and **embedded testing**:



Software Testing

VS



Embedded Testing

S.NO	Comparison Basis	Software Testing	Embedded Testing
1.	Definition	The Software Testing is a process of verifying and validating the software as per the customer requirements.	The embedded testing is process of checking the functional and non-functional attributes of both software and hardware in an embedded system.
2.	Performed	It can be performed in both ways, either manually or automatically. While it is mainly performed in one way, that is, manually.	
3.	Implemented on	Software testing can only be implemented on the software.	As compared to software testing, embedded testing can be implemented on both software as well as hardware.
4.	Testing	It is mainly used to test software functionality.	On the other hand, it is mainly used to tests hardware behavior.
5.	Testing type	Usually, Software testing is based on black-box testing.	On the other hand, the embedded testing can be white-box and black-box testing.
6.	Database	The Database can be tested under software testing procedures.	The database cannot be tested in the embedded testing process.
7.	Target	Generally, it is executed on the client-server application.	On the other hand, the embedded testing is executed on the hardware.
8.	Application	In software testing, we will test the Web and Mobile applications.	In embedded testing, we will test the Embedded systems.
9.	Expensive	As compared to the embedded testing, Software testing is a bit costly and time-taking process.	On the other hand, Embedded testing is less costly and less time-consuming as compared to the software testing.
10.	Examples	Yahoo Mail, Google Mail Android applications are the examples of Software testing.	Microcontrollers used in computers and Systems s of healthcare domain are the examples of embedded testing.

Conclusion

After seeing all the essential differences between **Software testing and Embedded testing**, we can conclude that both the testing techniques have their features and capabilities, which helps the test engineers to achieve their respective goals.

Furthermore, we can say that using both the testing technique provides various **advantages** as well as **disadvantages**, which means we may see some complexity while performing the embedded testing as compared to software testing.

As a result, the embedded testing is more challenging to perform as opposed to the traditional software testing processes.

Differences between GUI Testing and Usability Testing

In this article, we are going to discuss the difference between **GUI testing and Usability testing** based on the various parameters.

Before getting into the difference between **GUI testing and Usability testing**, we will briefly introduce **GUI testing and Usability testing**.

What is GUI (Graphical User Interface) Testing?

It is a unique **type of software testing**, which is mainly used to test/check the Graphical user interface of the application or the software.

The primary goal of the GUI testing is to make sure the functionalities of software or the application operate according to given requirements/specifications.

In GUI testing, the Specification could be specified as checking screens and various controls such as windows, menus, click buttons, icons, scroll bar, and dialogue boxes, etc.

In other words, we can say that **graphical user interface testing** essentially allows consistent communication between a user and a software product.

What is Usability Testing?

Usability testing is also one of the special testing techniques which comes under the **non-functional testing** of **black-box testing**.

The usability testing is used to test the defect in the end-user communication of software or the product. It ensures that the developed software is straightforward while using the system without facing any problem and makes end-user life easier.

In short, we can say that checking the **user-friendliness, efficiency, and accuracy of the application** is known as **Usability Testing**.

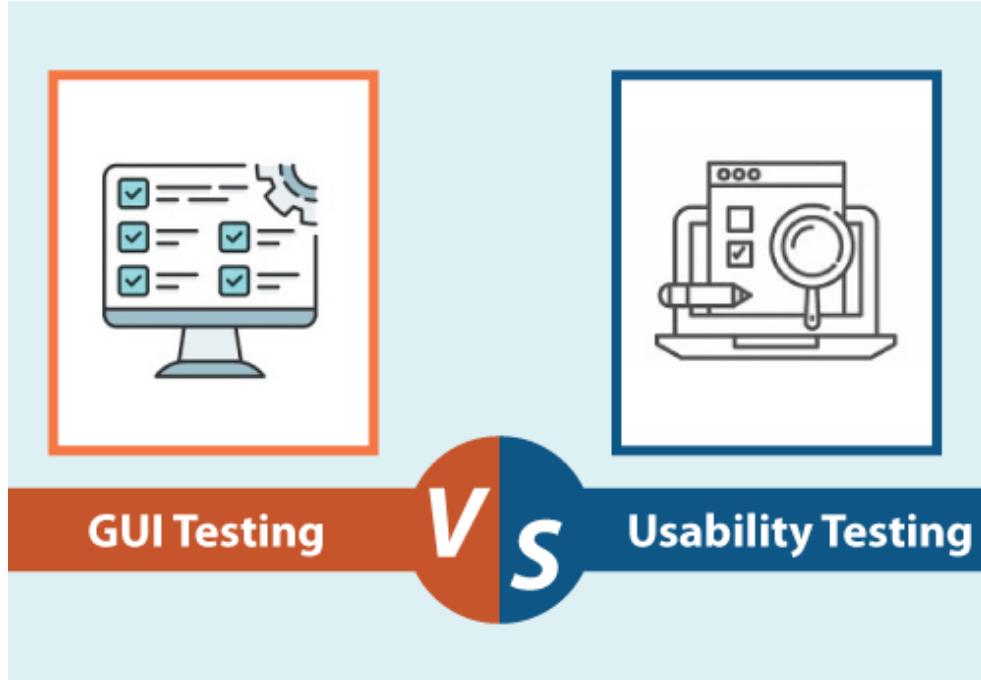
The main goal of executing the usability testing is to test that the application should be easy to use for the end-user who is meant to use it, whereas sustaining the client's specified functional and business requirements.

Basically, the usability testing evaluates the complete feasibility of the software from the user's perspective.

For more information regarding usability testing, refer to the following link: <https://www.javatpoint.com/usability-testing>.

GUI Testing VS Usability Testing

Let's understand the difference between GUI testing and Usability Testing on the following comparison basis table:



S.NO	Comparison Basis	GUI Testing	Usability Testing
1.	Definition	It is used to verify if the functional attributes creating the product design are working according to the user's prospects .	It is used to test the simplicity, availability of the product from the end-user's perspective.
2.	Emphasis on	The GUI testing emphasizes the look and feel of an application and involves the interface part of the software.	On the other hand, usability testing emphasizes the user-friendliness and product quality of an application.
3.	Test	Generally, in graphical user interface testing, the functionality of an application is not tested.	Whereas in usability testing, the functionality of an application is tested to verify whether it is user-friendly or not.
4.	Validation	In GUI testing, we validate the design specifications for the products.	The usability testing test if the creation of the user interface has been well created and is user-friendly.
5.	Target	The GUI testing is essential when the developers are targeting a precise user base like disabled people .	On the other hand, usability testing is not beneficial while developing an application for a unique user base.
6.	Ensures	The execution of GUI testing ensures the look and feel of an application by fulfilling the given user requirements and standards.	While the execution of usability testing ensures the user comfortless while using any application.
7.	Determine	Graphical user interface testing is used to determine the front-end portion of any application .	Usability testing determines the extent of the user-friendliness Interface as well as overall functioning of the software.
8.	Platform dependency	The GUI testing is implemented on several platforms to ensures its perfect look of the application.	The usability testing tests the application at its difficulty level.
9.	Importance	It is beneficial but less important as compared to usability testing.	Usability testing's importance is more as compared to the GUI testing.
10.	System Flow	The system flow of the product is left untouched by GUI testing.	In this, the systematic flow between various modules within the product gets tested.
11.	Advantage	The output we received after executing the Graphical user interface testing helps us engage more significant users due to its enhanced attractiveness.	On the other hand, getting an output by implementing the usability testing will also be beneficial for drawing a more significant part of users in the market.
12.	Coverage	The GUI testing gets covered the look and development of the application.	On the other hand, usability testing gets covered the features and capabilities of the product.

13.	Example	The graphical user interface testing tests all objects such as the size of the icon, setting the right combinations for font, the dialog box, contrast , attributes if they are appropriately exhibited throughout the screen.	The usability testing especially used to test the text input boxes for the users . And another example of usability testing is the scroll bars , which succeed for easy navigation of a website's pages.
-----	----------------	---	--

Conclusion

After seeing all the significant differences between the GUI testing and Usability testing, we can conclude that the execution of both Graphical user interface testing and usability testing can test some of the key software features, such as **Look and feel user-friendliness, efficiency, and accuracy of the application**.

The **Graphical user interface (GUI) testing and usability testing** will help the testing team takes essential procedures to increase the **performance, quality, user-friendliness, functionality**, and other vital essentials of the software product.

At last, we can say that if we do not execute the **GUI testing and usability testing** techniques, the development and testing may not be able to deliver a good quality software or the application.

The Difference between SDET and Tester

In this tutorial, we are going to understand the difference between SDET and Tester.

But before we are going through the difference between them, firstly, we will see the brief introduction of SDET and Tester.

SDET

SDET implies as **Software Development Engineer in Test** or **Software Design Engineer in Test**. SDET knows the internal design and execution of the software. Those people who can perform both the responsibility of development as well as testing process are known as SDET.

In other words, we can say that the tester who can write the code is known as **Software Development Engineer in Test (SDET)**. The Software Development Engineer in Test can also review the design and processes of the software product or an application.

The Software development engineer in the test role is initiated from **Microsoft**. At present, various organizations are demanding such SDET professionals who can participate in both developments of the application and testing of the developed software.

SDET expert's understanding is ultimately emphasized on **robustness, testability, and functioning** of **software testing** and **development process**.

Frequently, the SDET is involved in creating the quality, robust and high-performance code that is useful in designing the testing framework, automation of test cases, which can be used in place of a testing tool as well.

Tester

The **tester/ the test engineer/ manual tester/ quality analyst** does not take part in software development or write the codes. The test engineer is the person who implements the testing process on the software to identify the bugs or defects.

In other words, we can say that the manual tester test whether the software contains any defects or bugs or not.

The test engineer also analyses the various features of the specified software, and they don't understand or know the application's code or the development process of the software.

SDET VS Tester

Let's understand the difference between SDET and Tester on the following basis:



S.No.	Comparison Basis	SDET	Tester
-------	------------------	------	--------

1.	Performing	The Software Development Engineer in Test tests the specified application by using internal coding.	The Software Tester tests the application once the development process is done and the application is testing ready.
2.	Awareness	The SDET engineers understand the whole system from beginning to end.	On the other hand, the test engineer has a minimal understanding of the system.
3.	Skillset	The Software development engineer in the test has more abilities or a skill set as compared to the software tester.	The manual test engineer is less skilled as compared to the SDET.
4.	Programming Skill	The SDET has programming knowledge and can develop the code as well.	On the other hand, a Manual test engineer does not have any coding skills.
5.	Information about software requirements	The Software development engineer in test (SDET) knows all about software requirements as well as other features.	The test engineer has less information about software requirements as compared to the SDET.
6.	Responsibility	Separately from software testing, SDET can also be analyzing the performance of the tested application.	On the other hand, the manual test engineer is responsible in order to fulfill all the testing-related tasks.
7.	Salary	SDET gets more salary as compared to the manual test engineer as they have the automation skills.	The tester gets less salary as compared to the SDET because the manual test engineer does not have any automation skills.
8.	Roles	Generally, the SDET is part of software development and software testing.	On the other hand, the manual tester has more minor roles and responsibilities in software development.
9.	Internal knowledge	SDET engineers have the knowledge of designing, testing, and execution.	On the other hand, the manual test engineer doesn't have knowledge of the design and execution of the software application.

10.	Automation	<p>Generally, the Software development engineers in the test emphasize automating the monotonous scenarios to ensure that the manual test engineer can concentrate on more complex and edge situations and use their knowledge and skills more resourcefully.</p>	<p>For the automation process, the Manual test engineer has significantly less or no skills. But manual test engineers must be knowledgeable of using tools, which help in manual testing as well. For Example,</p> <ul style="list-style-type: none"> ◦ Working with cloud providers such as sauce labs for implementing the tests on several platform versions. ◦ Operating the Postman for implanting the API end-points, etc.
11.	Involvement	<p>The involvement of SDET is very significant as it can include in all the phases of the software development process, such as Designing, development, and testing.</p>	<p>Based on their knowledge and skills, the quality analyst/ manual tester can only be involved in the software development process's testing life cycle.</p>
12.	Testing Scope	<p>The SDET mainly concentrates on various testing types and procedures, such as Performance, Functional, security, and Non-functional, etc.</p>	<p>Usually, the manual tester emphasis on the performance perspective of the application under test. And the test engineer performs like a user/customer of the application under test and validation.</p>

Conclusion

After seeing all the significant comparisons between SDET and tester, we can say that the **Software development engineer in test (SDET)** is not just more than a tester.

They are the combination of both developers and a tester who has experience in end-user software requirements, project management, understanding of how-to code and create test automation tools, familiar with product or domain knowledge, and can also be taking part in product or software designing.

In the current software industry, all the abilities mentioned above make the **Software development engineer** very exclusive and highly in demand for the test role. Almost all good product organizations have this role in their respected teams and are highly valued.

The Differences Between Desktop Application testing, Client-Server Application Testing and Web Application Testing

In **software testing**, we have three different types of applications testing, such as:

- **Desktop application Testing**
- **Client-server application Testing**
- **Web applications Testing**

These software applications are being developed and created precisely to perform in different circumstances and areas.

All of the applications mentioned above relates to different environments where they can be tested individually and differently. In other words, we can say that the crucial aspect of this type of testing is the **environment** in which they are tested.

The user loses control over the environment entirely when the end-user continues from one testing type to another.

In this tutorial, we are going to understand the **difference between these (Desktop, Client-Server, and Web Application Testing) types of applications**.

But before we go through the difference between them, firstly, we will see the brief introduction of Desktop, Client-server, and Web applications testing.

What is Desktop Application Testing?

The first type of software application is the **Desktop application**. In a desktop application testing, the entire software is installed at the end-user machine, and only one user can access it at a time. The desktop application is further known as **a Stand-alone application testing**.

Basically, it is executed on personal computers, machines, and systems. In order to test the desktop application, we must know the operating system and database and how the user interacts with the application.

Desktop Application Testing



These types of applications concentrate only on the specific environment. The desktop application includes the analysis of the application entirely in elements such as **Graphical User Interface (GUI)**, **functionality**, **backend**, **database**, and **load**.

And the end-user also needs to identify the different methods to test the memory leaks and unhandled exceptions.

Advantages of Desktop Application testing

Following are some of the significant advantage of desktop application testing:

- Security is there because data hacking is not possible in desktop application testing.
- These applications are faster to access.
- There is no need for any server while performing the desktop application.

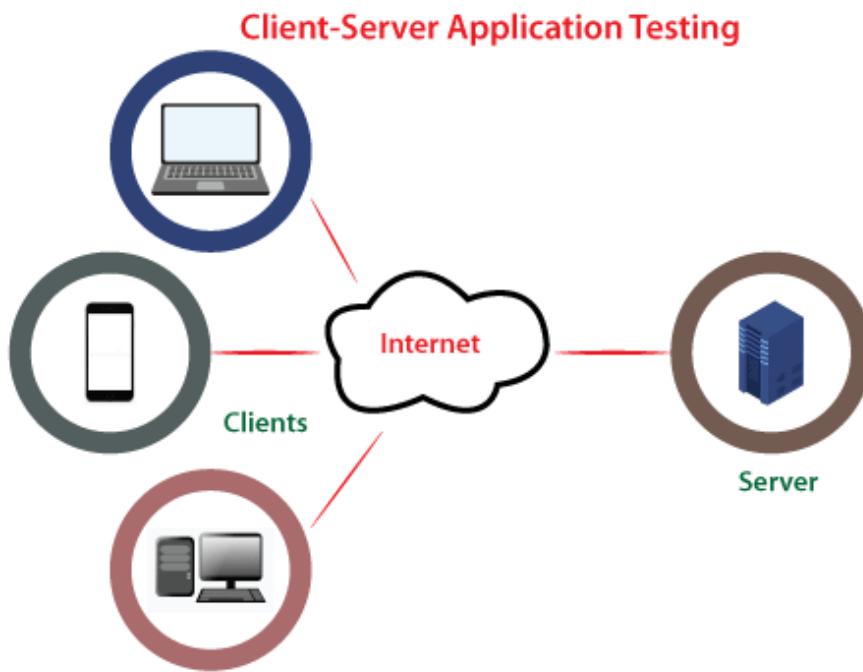
Disadvantages of Desktop Application testing

Some of the vital drawbacks of desktop application are as follows:

- Only one user can access the stand-alone application at a time.
- The installation process is necessary in order to execute the desktop/stand-alone application.
- In this application testing, system resource is occupied.
- Data sharing is not possible while using desktop applications.
- The maintains is very high or difficult in desktop application testing because the application is installed in the user machine. If any issues have occurred, we need to contact the particular person to fix those issues.

What is Client-Server Application Testing?

Client-server application is another type of software application. These types of applications are single-user applications, which execute on two or more systems and knowledge of networking is needed while executing the client-server applications.



Basically, in the client-server application, we will test the application graphical user interface on both the systems such as **server and client** and also test the **functionality, load, database, and communication between client and server**.

The end-user needs to identify the load and performance issues and also work on the code space under the Client-server application. In client-server types of application testing, we can obtain the **test cases** and **test scenarios** with the help of requirements and understanding.

We can implement the following types of tests under the Client-server application:

- **Manual testing**
- **Interoperability testing**
- **User interface testing**

- **Compatibility testing**
- **Functionality testing**
- **Configuration testing**

Advantages of Client-Server Application testing

Following are some of the significant advantage of Client-server application:

- In the client-server application testing maintains is a bit easy as compared to the other applications.
- We can access these applications rapidly.
- The client-server application is secured in terms of data sharing.
- Various users can access the application simultaneously.
- We can easily share the data in client-server application testing.

Disadvantages of Client-server Application testing

Some of the vital drawbacks of Client-server application are as follows:

- If the server is down, no one can access the application, which is the major drawback of the client-server application testing.
- Installation is necessary in order to execute the client-server application testing.
- In this type of application testing, the system resource will be inhabited the space.

What is Web Application Testing?

Web applications are applications where we test the application on **different browsers and diverse versions of the same browser and different operating systems**. And this type of application can be executed on **two or more machines**.

In other words, we can say that the web-based application executes in a web browser instead of being installed on an end user's device. That means the entire software is installed at a server, and the end-user can access the particular application by using the URL.



These applications are used to analyze the browser similarity and operating system compatibility, attributes such as **performance, usability, back-end, Graphical User Interface, accessibility**.

These applications can be executed on any device that has the internet connection such as **desktop computers, tablets, and mobile phones**.

Web applications are easily accessible, support several browsers and devices, platform-independent, which help us decrease the cost compared to the other types of software applications. Web-based applications are more complicated and require comprehensive testing.

To implement the web-based application, the test engineers need to know how the web application is communicating with the user. Understanding several technologies such as **JavaScript**, **PHP**, **debugging** is also necessary while performing the web application testing.

We can implement the following types of tests under the web-based applications:

- **Security testing**
- **User interface testing**
- **Browser compatibility testing**
- **Functionality testing**
- **Operating System compatibility testing**
- **Load testing**
- **Inter-operability testing**
- **Performance testing**
- **Storage and data volume testing**
- **Stress testing**

Advantages of Web Application testing

Following are some of the significant advantage of web application:

- Like client-server applications, the web application can also be accessed by several users at a time.
- We can easily share the data in web-based application testing.
- These types of application testing can be accessed more rapidly.
- There is no need for any maintains and installation while implementing the web-based applications.
- In terms of data sharing, the web-based application is highly secured.

Disadvantages of Web Application testing

Following are some of the most common disadvantage of web application:

- No one can access the application if the server is down, which is the primary disadvantage of Web-based application testing.

Desktop Application VS Client-Server Application VS Web Application testing

Let us significant differences between **desktop**, **client-server**, and **web application's** testing into the below table:



01

VS



02

VS



Desktop Application Testing

Client-Server Application Testing

Web Application Testing

S.No.	Comparison Basis	Desktop Application Testing	Client-Server Application Testing	Web Application Testing
1.	Definition	The desktop application is those applications which are installed on one computer and only accessible by one person.	The Client-server application is those applications, which are installed on both client and server software in order to access the application.	Web-based applications are those applications, which are accessed through a browser. And these applications are URL-driven and executed on different web browsers.
2.	Number of users	The desktop application only accesses by a single user.	The client-server can be accessed by multiple users but limited number.	The web-based application can access by the n-numbers of users across the internet and browsers.
3.	Requirements	In order to implement the desktop application, we need at least one computer system or the workstation.	In order to execute the client-server application testing, we need at least one server for loading the application and one client machine or system.	The web application can be implemented by personal laptops remotely by using a web browser and internet connectivity.
4.	Executed on	The desktop application is executed on a single machine or workstation.	Generally, the Client-server application is executed on a 2-tier application.	Usually, web-based application testing is executed on the 3-tier application.
5.	Environments	The stand-alone or desktop application testing are platform-dependent; that's why the environment is a user machine .	Usually, in client-server application testing, environment is the intranet .	The environment is web browsers in the web-based application testing.

6.	Awareness	There is no client and server on the desktop or stand-alone applications.	In the Client-server application, we must be aware of the server location.	In the web application, we may or may not have any information about the server location.
7.	Type of driven applications	The stand-alone application is desktop-driven .	The client-server applications are menu-driven application testing.	Web application is usually URL-driven testing .
8.	Connectivity	In stand-alone application testing, there is no server or client. We are already aware that the Desktop applications don't need any internet connectivity because they are only hosted on the user machines.	There are two or more than two systems in client-server application testing where one is server , and another one is the client . And the application is loaded on the server, and installed on the client machines.	There are two or more systems in web application testing, where one is the server , and another one is the client . In this, the application is loaded on the server, and there is not an executable file. As we know, Web applications rely on internet connectivity to perform, and internet connectivity tests need to be executed to analyze the application functioning at different speeds.
9.	Accessibility	These applications may or may not needed authentic access in order to use it.	These application's user is already known before as they might have a username/password to open the application.	All users can approach these applications.
10.	Types of Tests Performed	In stand-alone applications, we test the following characteristics of the application or the software: <ul style="list-style-type: none"> ○ Load ○ Graphical User Interface (GUI) features ○ Back-end (database) with the memory leaks issue. 	In the Client-Server application, we will test the following qualities of applications: <ul style="list-style-type: none"> ○ Functionality ○ Load and performance aspects ○ GUI (Graphical User Interface) on both sides. ○ back-end. 	In Web application testing, we will test the following features of the application: <ul style="list-style-type: none"> ○ Browser and Operating System (OS) compatibility. ○ User Interface Testing ○ Broken Link Testing ○ Data Volume testing ○ Static page testing ○ Load, Stress, Volume ○ Cross-Browser testing.

11.	Examples	Some of the important examples of desktop/stand application are as follows: <ul style="list-style-type: none"> ○ Adobe Photoshop ○ Installing software w of a Calculator ○ AutoCAD ○ MS Office 	Some of the important examples of Client-server application are as follows: <ul style="list-style-type: none"> ○ Web Browsers ○ FTP ○ .NET ○ E-mail ○ Gateway 	Some of the significant examples of the web application is as follows: <ul style="list-style-type: none"> ○ Facebook ○ yahoo ○ Gmail ○ Twitter
-----	-----------------	--	---	--

Conclusion

After seeing all the significant differences between desktop, client-server, and web application testing, we can conclude that all of these applications are entirely different and have various benefits and drawbacks.

In simple words, we can say that if we are testing a stand-alone or desktop application, then the center should be environment-oriented because the stone-alone/desktop application is executed only in a precise environment.

To check the application's usability, backend, load, and other functionality, we should perform the Desktop applications.

On the other hand, in **the client-server application testing**, the user performance is predictable and manageable because Client-server applications do not need any browser as per the media in order to access the application.

Whereas performing the Web application testing itself is a crucial process to make sure that the web application under test can be easily retrieved by the actual users and working appropriately throughout all the operating systems and browsers.

To check the application's performance, **features, accessibility, security, and usability**, we should perform the Web applications.

Nearly every test engineer works with applications such as Desktop/stand-alone application, Client-Server application, and Web Application Testing.

Software Testing MCQ

In this section, we are going to see a list of mostly asked Software Testing questions in MCQ style with an explanation of the answer for competitive exams and interviews. These frequently asked Software testing questions are given with the correct choice of answer among various options.

1) Which methodology is used to performed Maintenance testing?

- a. Breadth test and depth test
- b. Confirmation testing
- c. Retesting
- d. Sanity testing

[Show Answer](#)

[Workspace](#)

2) Which of the following is not part of the Test document?

- a. Test Case
- b. Requirements Traceability Matrix [RTM]
- c. Test strategy
- d. Project Initiation Note [PIN]

[Show Answer](#)

[Workspace](#)

3) Which term is used to define testing?

- a. Evaluating deliverable to find errors
- b. Finding broken code
- c. A stage of all projects
- d. None of the above

[Show Answer](#)

[Workspace](#)

4) Which of the following is not a valid phase of SDLC (Software Development Life Cycle)?

- a. Testing Phase
- b. Requirement Phase
- c. Deployment phase
- d. Testing closure

[Show Answer](#)

[Workspace](#)

5) Which of the following testing is also known as white-box testing?

- a. Structural testing
- b. Error guessing technique
- c. Design based testing
- d. None of the above

[Show Answer](#)

[Workspace](#)

6) Which of the following testing is related to the boundary value analysis?

- a. White box and black box testing
- b. White-box testing
- c. Black box testing
- d. None of the above

[Show Answer](#)

[Workspace](#)

7) Functional testing is a -----?

- a. Test design technique
- b. Test level
- c. SDLC Model
- d. Test type

[Show Answer](#)

[Workspace](#)

8) What are the different levels of Testing?

- a. Integration testing
- b. Unit testing
- c. System testing
- d. All of the above

[Show Answer](#)

[Workspace](#)

9) Which of the following is not a part of STLC (Software Testing Life Cycle)?

- a. Testing Planning
- b. Requirement Gathering
- c. Test Design
- d. Testing closure

[Show Answer](#)

[Workspace](#)

10) Sanity testing is a -----?

- a. Test type
- b. Test Execution Level
- c. Test Level
- d. Test design technique

[Show Answer](#)

[Workspace](#)

11) White box testing techniques are?

- a. Statement coverage testing
- b. Decision coverage testing

c. Data flow testing

d. All of the above

Show Answer

Workspace

12) In which environment we can performed the Alpha testing?

a. User's end

b. Developer's end

c. User's and developer's end

d. None of the above

Show Answer

Workspace

13) Which of the below is not a part of the Test Plan?

a. Schedule

b. Risk

c. Incident reports

d. Entry and exit criteria

Show Answer

Workspace

14) What is the key objective of Integration testing?

a. Design Errors

b. Interface Errors

c. Procedure Errors

d. None of the mentioned

Show Answer

Workspace

15) Exploratory testing is a -----?

a. Experience-based Test Design Technique

b. White Box Test Design Technique

c. Black Box Test Design Technique

d. Grey Box Test Design Technique

Show Answer

Workspace

16) What is the best time to perform Regression testing?

a. After the software has been modified

b. As frequently as possible

c. When the environment has been modified

d. Both option a & c

Show Answer

Workspace

17) Does the customer get a 100% bug-free product?

- a. Product is old
- b. Developers are super
- c. The testing team is not good
- d. All of the above

[Show Answer](#)

[Workspace](#)

18) Cyclomatic complexity is?

- a. White-box testing
- b. Black box testing
- c. Grey box testing
- d. All of the above

[Show Answer](#)

[Workspace](#)

19) Which of the following is not part of the Test type?

- a. Function testing
- b. System testing
- c. Statement testing
- d. Database testing

[Show Answer](#)

[Workspace](#)

20) Which Test Document is used to define the Exit Criteria of Testing?

- a. Defect Report
- b. Test Summary Report
- c. Test Case
- d. Test Plan

[Show Answer](#)

[Workspace](#)

21) Impact analysis helps us to decide which of the following testing?

- a. Exit Criteria
- b. How much regression testing should be done?
- c. Different Tools to perform Regression Testing
- d. How many more test cases need to write?

[Show Answer](#)

[Workspace](#)

22) Which testing technique is used for usability testing?

- a. White-box testing
- b. Grey box testing

c. Black Box testing

d. Combination of all

Show Answer

Workspace

23) Which is not the right approach of Incremental testing approach?

a. Big bang approach

b. Top-down approach

c. Functional incrimination

d. Bottom-up approach

Show Answer

Workspace

24) In which environment we can performed the Beta testing?

a. User's and developer's end

b. Developer's end

c. User's end

d. None of the above

Show Answer

Workspace

25) What is error guessing in software testing?

a. Test control management techniques

b. Test verification techniques

c. Test execution techniques

d. Test case design/ data management techniques

Show Answer

Workspace

26) After which phase, we can proceed to the white box testing?

a. After the coding phase

b. After designing phase

c. After SRS creation

d. After the installation phase

Show Answer

Workspace

27) Which of the following is not another name of white box testing?

a. Structural testing

b. Behavioral testing

c. Glass box testing

d. None of the mentioned

Show Answer

Workspace

28) The test levels are performed in which of the following order?

- a. Unit, Integration, System, Acceptance
- b. It is based on the nature of the project
- c. Unit, Integration, Acceptance, System
- d. Unit, System, Integration, Acceptance

[Show Answer](#)

[Workspace](#)

29) Define the term failure?

- a. A human action that produces an incorrect result.
- b. Its departure from specified behavior
- c. Found in the software; the result of an error.
- d. It is procedure or data definition in a computer database.

[Show Answer](#)

[Workspace](#)

30) "V" model is?

- a. Test type
- b. Test Level
- c. Test design technique
- d. Software development testing (SDLC) model

[Show Answer](#)

[Workspace](#)

31) Which of the below testing is executed without documentation and planning is known as?

- a. Regression Testing
- b. Adhoc Testing
- c. Unit Testing
- d. None of the above

[Show Answer](#)

[Workspace](#)

32) Which of the below testing is related to Non-functional testing?

- a. Unit Testing
- b. Black-box Testing
- c. Performance Testing
- d. None of the above

[Show Answer](#)

[Workspace](#)

33) Which of the below testing is related to black-box testing?

- a. Boundary value analysis
- b. Code path analysis

- c. Basic path testing
- d. None of the above

[Show Answer](#)[Workspace](#)

34) Which of the following testing is also called Acceptance testing?

- a. Beta testing
- b. White-box testing
- c. Grey box testing
- d. Alpha testing

[Show Answer](#)[Workspace](#)

35) ----- testing is used to check the code?

- a. Grey box testing
- b. Black box testing
- c. White-box testing
- d. Red box testing

[Show Answer](#)[Workspace](#)

36) The Regression test case is not a -----?

- a. Tests that focus on the software components, which have been modified.
- b. Low-level components are combined into clusters, which perform a specific software sub-function.
- c. Additional tests that emphasize software functions, which are likely to be affected by the change.
- d. A representative sample of tests, which will exercise all software functions.

[Show Answer](#)[Workspace](#)

37) Generally, which testing is used when shrink-wrapped software products are being established and part of an integration testing?

- a. Integration Testing
- b. Validation testing
- c. Regression Testing
- d. Smoke testing

[Show Answer](#)[Workspace](#)

38) Which of the following statement is used to discover errors in the test case?

- a. Incorrect logical operators or precedence
- b. Non-existent loop termination
- c. Comparison of different data types
- d. All of the above

[Show Answer](#)[Workspace](#)

39) The Decision table testing is a -----?

- a. White box Test Design Technique
- b. Black Box Test Design Technique
- c. Experience-based Test Design Technique
- d. Grey Box Test Design Technique

[Show Answer](#)[Workspace](#)

40) When we have to stop the testing?

- a. The faults have been fixed
- b. All the tests run
- c. The time completed
- d. The risk is resolved

[Show Answer](#)[Workspace](#)

41) ----- are those software mistakes that occurred during the coding phase?

- a. Defects
- b. Failures
- c. Errors
- d. Bugs

[Show Answer](#)[Workspace](#)

42) Which of the following is not a valid software testing technique?

- a. Inspections
- b. Data flow analysis
- c. Error guessing
- d. Walkthrough

[Show Answer](#)[Workspace](#)

43) Define the term verification in V and V model?

- a. Checking that we are building the system right
- b. Making sure that it is what the user wants
- c. Performed by an independent test team
- d. Checking that we are building the right system

[Show Answer](#)[Workspace](#)

44) What is the full form of SRS?

- a. Software respond system

- b. Software requirements specification
- c. System responds software
- d. System requirements specification

Show Answer

Workspace

45) What is the main task of test planning?

- a. Measuring and analyzing results
- b. Evaluating exit criteria and reporting
- c. Determining the test approach
- d. Preparing the test specification

Show Answer

Workspace

46) Which of the below statement is true about the Equivalence Partitioning technique?

- a. A black box testing technique appropriate to all levels of testing.
- b. A white box testing technique appropriate for component testing.
- c. The black box testing technique is used only by developers.
- d. A black box testing technique that can only be used during system testing.

Show Answer

Workspace

47) ITG stands for-----?

- a. Integration Testing Group
- b. Instantaneous Test Group
- c. Independent Test Group
- d. Individual Testing Group

Show Answer

Workspace

48) Which of the following testing is refers to as a fault-based testing technique?

- a. Stress testing
- b. Mutation testing
- c. Beta testing
- d. Unit testing

Show Answer

Workspace

49) ----- are the problems that threaten the success of a project but which has not yet happened.

- a. Risk
- b. Bug
- c. Failure
- d. Error

[Show Answer](#)[Workspace](#)

50) What is component testing?

- a. White-box testing
- b. Grey box testing
- c. Black box testing
- d. Both a & c

[Show Answer](#)[Workspace](#)

Software Testing MCQ Part-2

In this section, we are going to see a list of the top 50 frequently asked Software Testing questions in MCQ style with the correct choice of answer among various options along with suitable Explanation.

These Software Testing questions and answers emphasize all the areas of a specific topic. And this article is prepared to covers over more than ten topics in software testing. From the competitive exams and interviews point of view, these multiple-choice questions and answers are very helpful.

1) Which of the following testing technique deeply emphasizes on testing of one specific module?

- a. Inter-system testing
- b. Gorilla Testing
- c. Breadth Testing
- d. Fuzz Testing

[Show Answer](#)

[Workspace](#)

2) Which of the below testing is implemented initially?

- a. Static Testing
- b. Black-box Testing
- c. White-box Testing
- d. Dynamic Testing

[Show Answer](#)

[Workspace](#)

3) Which of the following testing tool does not supported by Database Testing?

- a. Unified Functional Testing [UFT]
- b. Selenium
- c. Rational Functional Tester [RFT]
- d. Application Lifecycle Management [ALM]

[Show Answer](#)

[Workspace](#)

4) ----- is one of the reputed testing standards.

- a. Microsoft
- b. ISO
- c. QAI
- d. M Bridge awards

[Show Answer](#)

[Workspace](#)

5) Which of the following abbreviation is correct for the terms SPICE?

- a. Software Process Improvement and Control Determination
- b. Software Process Improvement and Capability Determination
- c. Software Process Improvement and Compatibility Determination

d. None of the above

Show Answer

Workspace

6) ----- is not used in evaluating the size of the software?

- a. Function points
- b. KLOC
- c. Size of module
- d. None of the above

Show Answer

Workspace

7) Which of the following tool is not an open-source tool?

- a. Cucumber
- b. Selenium
- c. Bugzilla
- d. BugHost

Show Answer

Workspace

8) In which of the following testing level, the main focus is on customer usage?

- a. Validation Testing
- b. Alpha Testing
- c. Both Alpha and Beta Testing
- d. Beta Testing

Show Answer

Workspace

9) Which of the following testing technique can be used in order to determine the validation test?

- a. Black-box Testing
- b. White-box Testing
- c. Yellow- box Testing
- d. All of the above

Show Answer

Workspace

10) -----testing types is not a part of system testing?

- a. Stress Testing
- b. Recovery testing
- c. Random testing
- d. System Testing

Show Answer

Workspace

11) Which of the following evaluation method is used to evaluate the quality test cases?

- a. Verification
- b. Mutation Analysis
- c. Performance Analysis
- d. Validation

Show Answer

Workspace

12) ----- is not a part of the execution flow throughout the debugging process?

- a. Step Up
- b. Step Over
- c. Step Out
- d. Step into

Show Answer

Workspace

13) Which of the following testing techniques is used to test the code?

- a. Complex path testing
- b. Quality assurance of software
- c. Control structure testing
- d. Code coverage

Show Answer

Workspace

14) ----- testing is a testing technique where the actual data verified in the real environment.

- a. Regression Testing
- b. Alpha Testing
- c. Beta Testing
- d. None of the above

Show Answer

Workspace

15) Which of the below statements is true about reviews?

- a. Reviews should be performed on specifications, code, and test plans.
- b. Reviews are doubtful to identify faults in test plans.
- c. Reviews are the minimum operative way of testing code.
- d. Reviews cannot be performed on user requirements specifications.

Show Answer

Workspace

16) The following statement is related to which of the below options:

"It determines the quality of processes used to create a quality product. It is system of management activities, and preventive process. It applies for entire life cycle and deals with process?"

- a. Quality control
- b. Validation

- c. Verification
- d. Quality Assurance

Show Answer

Workspace

17) Which of the following is not called as white-box testing?

- a. Open box Testing
- b. Glass box Testing
- c. Clear box Testing
- d. Closed box Testing

Show Answer

Workspace

18) ----- is known as a variance from software product specifications.

- a. Defects
- b. Review
- c. Requirement
- d. Report

Show Answer

Workspace

19) Re-testing of a single program or component after a modification has been made is known as-----?

- a. Regional Regression Testing
- b. Retesting
- c. Unit Regression Testing
- d. Full Regression Testing

Show Answer

Workspace

20) Select whether the given statement is true or false:

"Requirement specification, design, coding, testing, installation and maintenance is the various phases of SDLC (software development life cycle)."

- a. True
- b. False

Show Answer

Workspace

21) The given statement is related to which of the following options: "It determines the quality of product, and it is a specific part of the Quality assurance procedure. It is a corrective process, and it applies to a specific product and deals with the product."

- a. Verification
- b. Quality control
- c. Quality Assurance
- d. Validation

[Show Answer](#)[Workspace](#)

22) Which of the following approaches are the part of Integration testing?

- a. Top-down approach
- b. Bottom-up approach
- c. Big-bang approach
- d. All of the above

[Show Answer](#)[Workspace](#)

23) Which of the following testing is the part of non-functional testing?

- a. Unit Testing
- b. Performance Testing
- c. System Testing
- d. Integration Testing

[Show Answer](#)[Workspace](#)

24) Which of the following is/are true regarding the below statement? "Product risks affects the performance or quality of the software or the application."

- a. True
- b. False

[Show Answer](#)[Workspace](#)

25) In -----, the test engineer implements the same test cases on a modified build.

- a. Adhoc Testing
- b. Regression Testing
- c. Sanity Testing
- d. Retesting

[Show Answer](#)[Workspace](#)

26) "Automation testing should be performed before starting the manual testing" is the true statement or false?

- a. True
- b. False

[Show Answer](#)[Workspace](#)

27) Notifying the developers which bugs need to be fixed first is known as-----.

- a. Fixability
- b. Traceability
- c. Priority

d. Severity

Show Answer

Workspace

28) ----- is the process of re-testing the modules that connected to the program or components after the modification has occurred.

- a. Regional regression Testing
- b. Re-testing
- c. Full Regression Testing
- d. Unit Regression Testing

Show Answer

Workspace

29) Which of the following life cycle contains the below phases?

Requirement specification, test case design, test execution, Defect tracking, and maintenance.

- a. BLC
- b. SQLC
- c. STLC
- d. SDLC

Show Answer

Workspace

30) Which of the following testing techniques includes how well the user will understand and interact with the system?

- a. Alpha Testing
- b. User Acceptance Testing
- c. Beta Testing
- d. Usability Testing

Show Answer

Workspace

31) In which of the following SDLC models we needs to start testing activities along with development activities?

- a. Spiral Model
- b. V-Model
- c. Liner Model
- d. Waterfall Model

Show Answer

Workspace

32) How rigorously the bug is affecting the application is known as-----.

- a. Fixability
- b. Priority
- c. Traceability

d. Severity

[Show Answer](#)

[Workspace](#)

33) ----- plan is used to overcome the risk.

- a. Master Plan
- b. Mitigation Plan
- c. Migration Plan
- d. Maintenance Plan

[Show Answer](#)

[Workspace](#)

34) Which of the following quality assurance approaches are usually measured?

- a. Preventive
- b. Detective
- c. Proactive
- d. Corrective

[Show Answer](#)

[Workspace](#)

35) Which of the following statement is true regarding verification?

- a. Performed by an independent test team
- b. Checking that we are building the right system
- c. Making sure that it is what the user wants
- d. Checking that we are building the system right

[Show Answer](#)

[Workspace](#)

36) If an expected result is not identified, then---?

- a. It may difficult to repeat the test
- b. We cannot automate the user inputs
- c. We cannot run the test
- d. It may be difficult to determine if the test has passed or failed.

[Show Answer](#)

[Workspace](#)

37) A regression test-----?

- a. Will help us to ensure unchanged areas of the software have not been affected
- b. Can only be run during user acceptance testing
- c. Will always be automated
- d. Will help us to ensure changed areas of the software have not been affected

[Show Answer](#)

[Workspace](#)

38) Which of the following test activity is used to identify the bugs most cost-effectively?

- a. Planning
- b. Execution
- c. Check exit criteria completion
- d. Design

[Show Answer](#)

[Workspace](#)

39) Which of the following process starting with the terminal modules?

- a. Module integration
- b. Top-down integration
- c. Bottom-up integration
- d. Now of the above

[Show Answer](#)

[Workspace](#)

40) The primary objective of the requirement phase is-----.

- a. To understand the user requirement
- b. To define the scope of testing
- c. To check the needs
- d. All of the mentioned above

[Show Answer](#)

[Workspace](#)

41) Which of the following does not include in the defect management process?

- a. Deliverable base-lining
- b. Management reporting
- c. Defect prevention
- d. None of the above

[Show Answer](#)

[Workspace](#)

42) In loop testing methodology, we can successfully test-----?

- a. Nested loop
- b. Concatenated loop
- c. Simple loop
- d. All of the above

[Show Answer](#)

[Workspace](#)

43) The primary purpose of acceptance testing is to -----?

- a. Test by an independent test team
- b. Test from a business point of view
- c. Test the system with other systems
- d. Finding faults in the system

Show Answer**Workspace**

44) ----- is not a part of performance testing?

- a. Recovery testing
- b. Measuring the response time
- c. Simulating many users
- d. Generating many transactions

Show Answer**Workspace**

45) Which of the following options are part of non-functional testing?

- a. Performance testing
- b. System testing
- c. Usability testing
- d. Both A & C

Show Answer**Workspace**

46) Acceptance test cases are based on-----.

- a. Design
- b. Requirements
- c. Decision table
- d. Code

Show Answer**Workspace**

47) Test cases are designed during which of the following technique?

- a. Test configuration
- b. Test recording
- c. Test specification
- d. Test planning

Show Answer**Workspace**

48) To test a function, the programmer has to write a----- that's called the function to be tested and passes its test data.

- a. Proxy
- b. Stub
- c. None of the above

Show Answer**Workspace**

49) SRS is also known as the specification of-----.

- a. Integrated Testing

- b. Black-box Testing
- c. Stress Testing
- d. White-box Testing

Show Answer

Workspace

50) Which of the following statement is true about the test design technique?

- a. A process for selecting test cases
- b. A way to determine in a test plan what has to be done
- c. A process for measuring expected output
- d. A way to determine the quality of software

Show Answer

Workspace

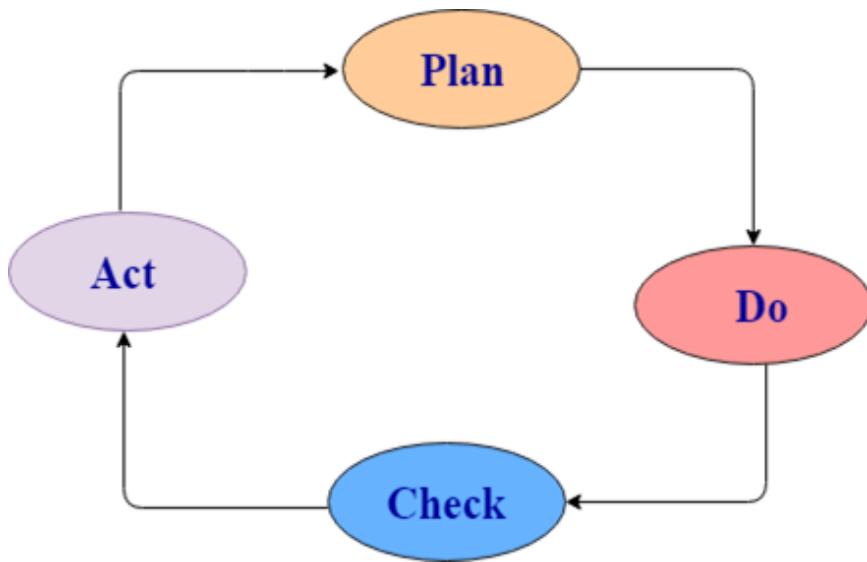
Software Testing Interview Questions



A list of mostly asked **software testing interview questions** or **QTP interview questions** and answers are given below.

1) What is the PDCA cycle and where testing fits in?

There are four steps in a normal software development process. In short, these steps are referred to as PDCA.



PDCA stands for Plan, Do, Check, Act.

- **Plan:** It defines the goal and the plan for achieving that goal.
- **Do/ Execute:** It depends on the plan strategy decided during the planning stage. It is done according to this phase.
- **Check:** This is the testing part of the software development phase. It is used to ensure that we are moving according to plan and getting the desired result.
- **Act:** This step is used to solve if there any issue has occurred during the check cycle. It takes appropriate action accordingly and revises the plan again.

The developers do the "planning and building" of the project while testers do the "check" part of the project.

2) What is the difference between the white box, black box, and gray box testing?

Black box Testing: The strategy of black box testing is based on requirements and specification. It requires no need of knowledge of internal path, structure or implementation of the software being tested.

White box Testing: White box testing is based on internal paths, code structure, and implementation of the software being tested. It requires a full and detail programming skill.

Gray box Testing: This is another type of testing in which we look into the box which is being tested, It is done only to understand how it has been implemented. After that, we close the box and use the black box testing.

Following are the differences among white box, black box, and gray box testing are:

Black box testing	Gray box testing	White box testing
Black box testing does not need the implementation knowledge of a program.	Gray box testing knows the limited knowledge of an internal program.	In white box testing, implementation details of a program are fully required.
It has a low granularity.	It has a medium granularity.	It has a high granularity.
It is also known as opaque box testing, closed box testing, input-output testing, data-driven testing, behavioral testing and functional testing.	It is also known as translucent testing.	It is also known as glass box testing, clear box testing.
It is a user acceptance testing, i.e., it is done by end users.	It is also a user acceptance testing.	Testers and programmers mainly do it.
Test cases are made by the functional specifications as internal details are not known.	Test cases are made by the internal details of a program.	Test cases are made by the internal details of a program.

3) What are the advantages of designing tests early in the life cycle?

Designing tests early in the life cycle prevent defects from being in the main code.

4) What are the types of defects?

There are three types of defects: Wrong, missing, and extra.

Wrong: These defects are occurred due to requirements have been implemented incorrectly.

Missing: It is used to specify the missing things, i.e., a specification was not implemented, or the requirement of the customer was not appropriately noted.

Extra: This is an extra facility incorporated into the product that was not given by the end customer. It is always a variance from the specification but may be an attribute that was desired by the customer. However, it is considered as a defect because of the variance from the user requirements.

5) What is exploratory testing?

Simultaneous test design and execution against an application is called exploratory testing. In this testing, the tester uses his domain knowledge and testing experience to predict where and under what conditions the system might behave unexpectedly.

6) When should exploratory testing be performed?

Exploratory testing is performed as a final check before the software is released. It is a complementary activity to automated regression testing.

7) What are the advantages of designing tests early in the life cycle?

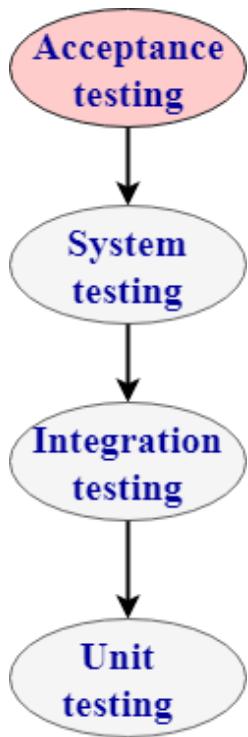
It helps you to prevent defects in the code.

8) Tell me about the risk-based testing.

The risk-based testing is a testing strategy that is based on prioritizing tests by risks. It is based on a detailed risk analysis approach which categorizes the risks by their priority. Highest priority risks are resolved first.

9) What is acceptance testing?

Acceptance testing is done to enable a user/customer to determine whether to accept a software product. It also validates whether the software follows a set of agreed acceptance criteria. In this level, the system is tested for the user acceptability.



Types of acceptance testing are:

1. **User acceptance testing:** It is also known as end-user testing. This type of testing is performed after the product is tested by the testers. The user acceptance testing is testing performed concerning the user needs, requirements, and business processes to determine whether the system satisfies the acceptance criteria or not.
2. **Operational acceptance testing:** An operational acceptance testing is performed before the product is released in the market. But, it is performed after the user acceptance testing.
3. **Contract and regulation acceptance testing:** In the case of contract acceptance testing, the system is tested against certain criteria and the criteria are made in a contract. In the case of regulation acceptance testing, the software application is checked whether it meets the government regulations or not.
4. **Alpha and beta testing:** Alpha testing is performed in the development environment before it is released to the customer. Input is taken from the alpha testers, and then the developer fixes the bug to improve the quality of a product. Unlike alpha testing, beta testing is performed in the customer environment. Customer performs the testing and provides the feedback, which is then implemented to improve the quality of a product.

10) What is accessibility testing?

Accessibility testing is used to verify whether a software product is accessible to the people having disabilities (deaf, blind, mentally disabled etc.).

11) What is Adhoc testing?

Ad-hoc testing is a testing phase where the tester tries to 'break' the system by randomly trying the system's functionality.

12) What is Agile testing?

Agile testing is a testing practice that uses agile methodologies i.e. follow test-first design paradigm.

13) What is API (Application Programming Interface)?

Application Programming Interface is a formalized set of software calls and routines that can be referenced by an application program to access supporting system or network services.

14) What do you mean by automated testing?

Testing by using software tools which execute test without manual intervention is known as automated testing. Automated testing can be used in GUI, performance, API, etc.

15) What is Bottom-up testing?

The Bottom-up testing is a testing approach which follows integration testing where the lowest level components are tested first, after that the higher level components are tested. The process is repeated until the testing of the top-level component.

16) What is Baseline Testing?

In Baseline testing, a set of tests is run to capture performance information. Baseline testing improves the performance and capabilities of the application by using the information collected and make the changes in the application. Baseline compares the present performance of the application with its previous performance.

17) What is Benchmark Testing?

Benchmarking testing is the process of comparing application performance with respect to the industry standard given by some other organization.

It is a standard testing which specifies where our application stands with respect to others.

18) Which types of testing are important for web testing?

There are two types of testing which are very important for web testing:

- **Performance testing:** Performance testing is a testing technique in which quality attributes of a system are measured such as responsiveness, speed under different load conditions and scalability. The performance testing describes which attributes need to be improved before the product is released in the market.
- **Security testing:** Security testing is a testing technique which determines that the data and resources be saved from the intruders.

19) What is the difference between web application and desktop application in the scenario of testing?

The difference between a web application and desktop application is that a web application is open to the world with potentially many users accessing the application simultaneously at various times, so load testing and stress testing are important. Web applications are also prone to all forms of attacks, mostly DDOS, so security testing is also very important in the case of web applications.

20) What is the difference between verification and validation?

Difference between verification and validation:

Verification	Validation
Verification is Static Testing.	Validation is Dynamic Testing.
Verification occurs before Validation.	Validation occurs after Verification.
Verification evaluates plans, document, requirements and specification.	Validation evaluates products.
In verification, inputs are the checklist, issues list, walkthroughs, and inspection.	Invalidation testing, the actual product is tested.
Verification output is a set of document, plans, specification and requirement documents.	Invalidation actual product is output.

21) What is the difference between Retesting and Regression Testing?

A list of differences between Retesting and Regression Testing:

Regression	Retesting
Regression is a type of software testing that checks the code change does not affect the current features and functions of an application.	Retesting is the process of testing that checks the test cases which were failed in the final execution.
The main purpose of regression testing is that the changes made to the code should not affect the existing functionalities.	Retesting is applied on the defect fixes.
Defect verification is not an element of Regression testing.	Defect verification is an element of regression testing.
Automation can be performed for regression testing while manual testing could be expensive and time-consuming.	Automation cannot be performed for Retesting.
Regression testing is also known as generic testing.	Retesting is also known as planned testing.
Regression testing concern with executing test cases that was passed in earlier builds. Retesting concern with executing those test cases that are failed earlier.	Regression testing can be performed in parallel with the retesting. Priority of retesting is higher than the regression testing.

22) What is the difference between preventative and reactive approaches to testing?

Preventative tests are designed earlier, and reactive tests are designed after the software has been produced.

23) What is the purpose of exit criteria?

The exit criteria are used to define the completion of the test level.

24) Why is the decision table testing used?

A decision table consists of inputs in a column with the outputs in the same column but below the inputs.

The decision table testing is used for testing systems for which the specification takes the form of rules or cause-effect combination. The reminders you get in the table explore combinations of inputs to define the output produced.

25) What is alpha and beta testing?

These are the key differences between alpha and beta testing:

No.	Alpha Testing	Beta Testing
1)	It is always done by developers at the software development site.	It is always performed by customers at their site.
2)	It is also performed by Independent testing team	It is not be performed by Independent testing team
3)	It is not open to the market and public.	It is open to the market and public.
4)	It is always performed in a virtual environment.	It is always performed in a real-time environment.
5)	It is used for software applications and projects.	It is used for software products.

6)	It follows the category of both white box testing and Black Box Testing.	It is only the kind of Black Box Testing.
7)	It is not known by any other name.	It is also known as field testing.

26) What is Random/Monkey Testing?

Random testing is also known as monkey testing. In this testing, data is generated randomly often using a tool. The data is generated either using a tool or some automated mechanism.

Random testing has some limitations:

- Most of the random tests are redundant and unrealistic.
- It needs more time to analyze results.
- It is not possible to recreate the test if you do not record what data was used for testing.

27) What is the negative and positive testing?

Negative Testing: When you put an invalid input and receive errors is known as negative testing.

Positive Testing: When you put in the valid input and expect some actions that are completed according to the specification is known as positive testing.

28) What is the benefit of test independence?

Test independence is very useful because it avoids author bias in defining effective tests.

29) What is the boundary value analysis/testing?

In boundary value analysis/testing, we only test the exact boundaries rather than hitting in the middle. For example: If there is a bank application where you can withdraw a maximum of 25000 and a minimum of 100. So in boundary value testing we only test above the max and below the max. This covers all scenarios.

The following figure shows the boundary value testing for the above-discussed bank application. TC1 and TC2 are sufficient to test all conditions for the bank. TC3 and TC4 are duplicate/redundant test cases which do not add any value to the testing. So by applying proper boundary value fundamentals, we can avoid duplicate test cases, which do not add value to the testing.

30) How would you test the login feature of a web application?

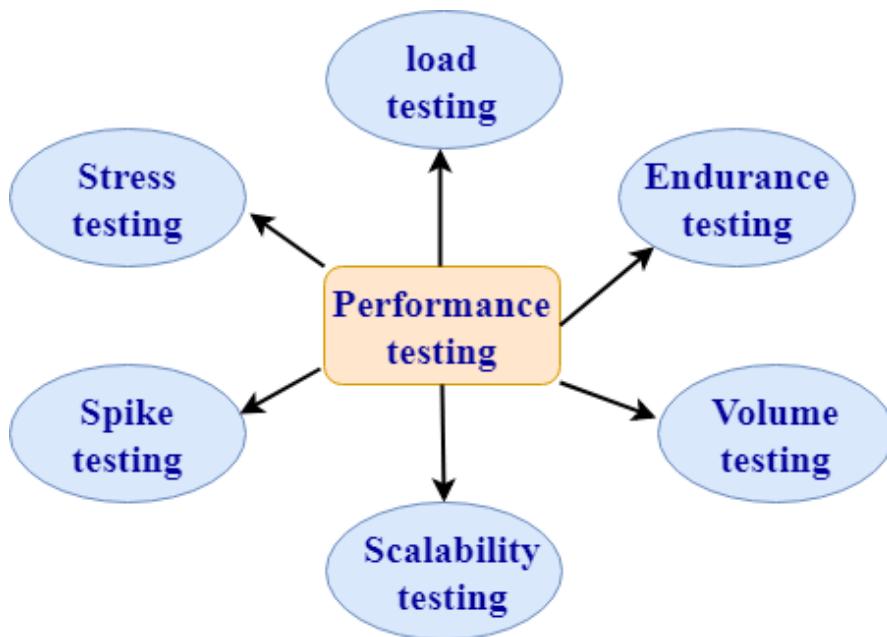
There are many ways to test the login feature of a web application:

- Sign in with valid login, Close browser and reopen and see whether you are still logged in or not.
- Sign in, then log out and then go back to the login page to see if you are truly logged out.
- Log in, then go back to the same page, do you see the login screen again?
- Session management is important. You must focus on how do we keep track of logged in users, is it via cookies or web sessions?
- Sign in from one browser, open another browser to see if you need to sign in again?
- Log in, change the password, and then log out, then see if you can log in again with the old password.

31) What are the types of performance testing?

Performance testing: Performance testing is a testing technique which determines the performance of the system such as speed, scalability, and stability under various load conditions. The product undergoes the performance testing before it gets live in the market.

Types of software testing are:



1. Load testing:

- Load testing is a testing technique in which system is tested with an increasing load until it reaches the threshold value.

Note: An increasing load means the increasing the number of users.

- The main purpose of load testing is to check the response time of the system with an increasing amount of load.
- Load testing is non-functional testing means that the only non-functional requirements are tested.
- Load testing is performed to make sure that the system can withstand a heavy load

2. Stress testing:

- Stress testing is a testing technique to check the system when hardware resources are not enough such as CPU, memory, disk space, etc.
- In case of stress testing, software is tested when the system is loaded with the number of processes and the hardware resources are less.
- The main purpose of stress testing is to check the failure of the system and to determine how to recover from this failure is known as recoverability.
- Stress testing is non-functional testing means that the only non-functional requirements are tested.

3. Spike testing:

- Spike testing is a subset of load testing. This type of testing checks the instability of the application when the load is varied.
- There are different cases to be considered during testing:
 - The first case is not to allow the number of users so that the system will not suffer heavy load.
 - The second case is to provide warnings to the extra joiners, and this would slow down the response time.

4. Endurance testing:

- Endurance testing is a subset of load testing. This type of testing checks the behavior of the system.
- Endurance testing is non-functional testing means that the only non-functional requirements are tested.

- Endurance testing is also known as Soak testing.
- Endurance testing checks the issues such as memory leak. A memory leak occurs when the program does not release its allocated memory after its use. Sometimes the application does not release its memory even after its use and this unusable memory cause memory leak. This causes an issue when the application runs for a long duration.
- Some of the main issues that are viewed during this testing are:
 - Memory leaks occurred due to an application.
 - Memory leaks occurred due to a database connection.
 - Memory leaks occurred due to a third party software.

5. Volume testing:

- Volume testing is a testing technique in which the system is tested when the volume of data is increased.
- Volume testing is also known as flood testing.
- Volume testing is non-functional testing means that the only non-functional requirements are tested.
- For example: If we want to apply the volume testing then we need to expand the database size, i.e., adding more data into the database table and then perform the test.

6. Scalability testing

- Scalability testing is a testing technique that ensures that the system works well in proportion to the growing demands of the end users.
- Following are the attributes checked during this testing:
 - Response time
 - Throughput
 - Number of users required for performance test
 - Threshold load
 - CPU usage
 - Memory usage
 - Network usage

32) What is the difference between functional and non-functional testing?

Basis of comparison	Functional testing	Non-functional testing
Description	Functional testing is a testing technique which checks that function of the application works under the requirement specification.	Non-functional testing checks all the non-functional aspects such as performance, usability, reliability, etc.
Execution	Functional testing is implemented before non-functional testing.	Non-functional testing is performed after functional testing.
Focus area	It depends on the customer requirements.	It depends on the customer expectations.
Requirement	Functional requirements can be easily defined.	Non-functional requirements cannot be easily defined.

Manual testing	Functional testing can be performed by manual testing.	Non-functional testing cannot be performed by manual testing.
Testing types	<p>Following are the types of functional testing:</p> <ul style="list-style-type: none"> ◦ Unit testing ◦ Acceptance testing ◦ Integration testing ◦ System testing 	<p>Following are the types of non-functional testing:</p> <ul style="list-style-type: none"> ◦ Performance testing ◦ Load testing ◦ Stress testing ◦ Volume testing ◦ Security testing ◦ Installation testing ◦ Recovery testing

33) What is the difference between static and dynamic testing?

Static testing	Dynamic testing
Static testing is a white box testing technique which is done at the initial stage of the software development lifecycle.	Dynamic testing is a testing process which is done at the later stage of the software development lifecycle.
Static testing is performed before the code deployment.	Dynamic testing is performed after the code deployment.
It is implemented at the verification stage.	It is implemented at the validation stage.
Execution of code is not done during this type of testing.	Execution of code is necessary for the dynamic testing.
In the case of static testing, the checklist is made for the testing process.	In the case of dynamic testing, test cases are executed.

34) What is the difference between negative and positive testing?

Positive testing	Negative testing
Positive testing means testing the application by providing valid data.	Negative testing means testing the application by providing the invalid data.
In case of positive testing, tester always checks the application for a valid set of values.	In the case of negative testing, tester always checks the application for the invalid set of values.
Positive testing is done by considering the positive point of view for example: checking the first name field by providing the value such as "Akshay".	Negative testing is done by considering the negative point of view for example: checking the first name field by providing the value such as "Akshay123".
It verifies the known set of test conditions.	It verifies the unknown set of conditions.
The positive testing checks the behavior of the system by providing the valid set of data.	The negative testing tests the behavior of the system by providing the invalid set of data.

The main purpose of the positive testing is to prove that the project works well according to the customer requirements.	The main purpose of the negative testing is to break the project by providing the invalid set of data.
The positive testing tries to prove that the project meets all the customer requirements.	The negative testing tries to prove that the project does not meet all the customer requirements.

35) What are the different models available in SDLC?

There are various models available in software testing, which are the following:

- Waterfall model
- Spiral Model
- Prototype model
- Verification and validation model
- Hybrid model
- Agile model
- Rational unified process model[RUP]
- Rapid Application development [RAD]

36) List out the difference between smoke testing and sanity testing and dry run testing?

Following are the differences between smoke, sanity, and dry run testing:

Smoke testing	Sanity testing	Dry-run testing
It is shallow, wide and scripted testing.	It is narrow and deep and unscripted testing	A dry run testing is a process where the effects of a possible failure are internally mitigated.
When the builds come, we will write the automation script and execute the scripts. So it will perform automatically.	It will perform manually.	For Example, An aerospace company may conduct a Dry run of a takeoff using a new aircraft and a runway before the first test flight.
It will take all the essential features and perform high-level testing.	It will take some significant features and perform in-depth testing.	

37) How do we test a web application? What are the types of tests we perform on the web application?

To test any web application such as **Yahoo**, **Gmail**, and so on, we will perform the following testing:

- Functional testing
- Integration testing
- System testing
- Performance testing
- Compatibility testing (test the application on the various operating systems, multiple browsers, and different version)

- Usability testing (check whether it is user friendly)
- Ad-hoc testing
- Accessibility testing
- Smoke testing
- Regression testing
- Security testing
- Globalization testing (only if it is developed in different languages)

38) Why do we need to perform compatibility testing?

We might have developed the software in one platform, and the chances are there that users might use it in the different platforms. Hence, it could be possible that they may encounter some bugs and stop using the application, and the business might get affected. Therefore, we will perform one round of Compatibility testing.

39) How many test cases we can write in a day?

We can tell anywhere between 2-5 test cases.

- First test case → 1st day, 2nd day.
- Second test case → 3rd day, 4th day.
- Forth test case → 5th day.
- 9-10 test cases → 19th day.

Primarily, we use to write 2-5 test cases, but in future stages we write around 6-7 because, at that time, we have the better product knowledge, we start re-using the test cases, and the experience on the product.

40) How many test cases can we review per day?

It would be around 7 test cases we write so that we can review $7 \times 3 = 21$ test cases. And we can say that 25-30 test case per day.

41) How many test cases can we run in a day?

We can run around 30-55 test cases per day.

Note: For these types of questions (39-41), always remember the ratio: x test cases we can write, 3x test cases we can review, and 5x test cases we can execute per day.

42) Does the customer get a 100% bug-free product?

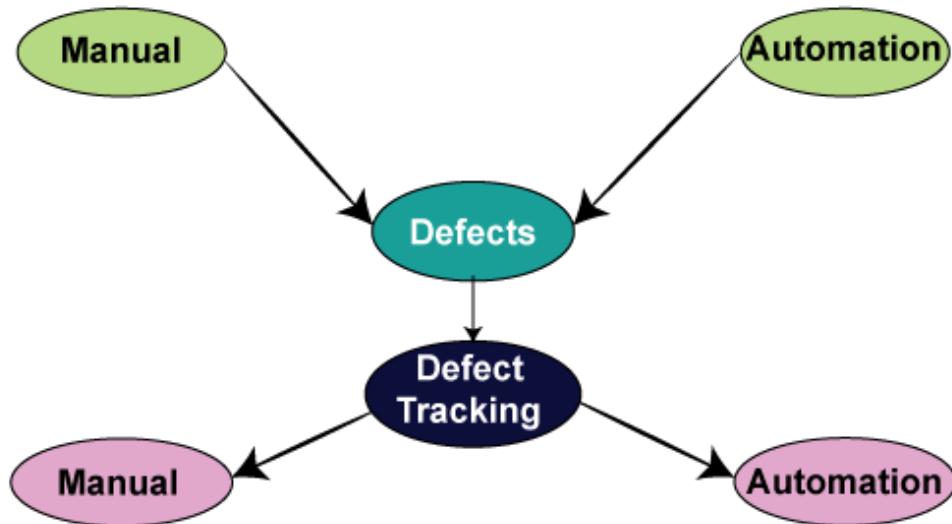
1. The testing team is not good
2. Developers are super
3. Product is old
4. All of the above

The correct answer is **testing team is not good** because sometimes the fundamentals of software testing define that no product has zero bugs.

43) How to track the bug manually and with the help of automation?

We can track the bug manually as:

- Identify the bug.
- Make sure that it is not duplicate (that is, check it in bug repository).
- Prepare a bug report.
- Store it in bug repository.
- Send it to the development team.
- Manage the bug life cycle (i.e., keep modifying the status).



Tracking the bug with the help of **automation** i.e., bug tracking tool:

We have various bug tracking tools available in the market, such as:

- Jira
- Bugzilla
- Mantis
- Telelogic
- Rational Clear Quest
- Bug_track
- Quality center (it is a test management tool, a part of it is used to track the bugs)

Note: Here, we have two categories of tools:

A product based: In the product based companies, they will use only one bug tracking tool.

Service-based: In service-based companies, they have many projects of different customers, and every project will have different bug tracking tools.

44) Why does an application have bugs?

The software can have a bug for the following reasons:

- Software complexity
- Programming errors
- If no communications are happening between the customer and the company, i.e., an application should or should not perform according to the software's needs.
- Modification in requirements
- Time pressure.

45) When we perform testing?

We will perform testing whenever we need to check all requirements are executed correctly or not, and to make sure that we are delivering the right quality product.

46) When do we stop the testing?

We can stop testing whenever we have the following:

- Once the functionality of the application is stable.
- When the time is less, then we test the necessary features, and we stop it.
- The client's budget.
- When the essential feature itself is not working correctly.

47) For which and all types of testing do we write test cases?

We can write test cases for the following types of testing:

Different types of testing	Test cases
Smoke testing	In this, we will write only standard features; thus, we can pull out some test cases that have all the necessary functions. Therefore, we do not have to write a test case for smoke testing.
Functional/unit testing	Yes, we write the test case for unit testing.
Integration testing	Yes, we write the test case for integration testing.
System testing	Yes, we write the test case for system testing.
Acceptance testing	Yes, but here the customer may write the test case.
Compatibility testing	In this, we don't have to write the test case because the same test cases as above are used for testing on different platforms.
Adhoc testing	We don't write the test case for the Adhoc testing because there are some random scenarios or the ideas, which we used at the time of Adhoc time. Though, if we identify the critical bug, then we convert that scenario into a test case.
Performance testing	We might not write the test cases because we will perform this testing with the help of performance tools.
Usability testing	In this, we use the regular checklist; therefore, we don't write the test case because here we are only testing the look and feel of the application.
Accessibility testing	In accessibility testing, we also use the checklist.
Reliability testing	Here, we don't write the manual test cases as we are using the automation tool to perform reliability testing.
Regression testing	Yes, we write the test cases for functional, integration, and system testing.
Recovery testing	Yes, we write the test cases for recovery testing, and also check how the product recovers from the crash.
Security testing	Yes, we write the test case for security testing.

Globalization testing: Localization testing Internationalization testing	Yes, we write the test case for L10N testing. Yes, we write the test case for I18N testing.
---	---

48) What is the difference between the traceability matrix and the test case review process?

Traceability matrix	Test case review
In this, we will make sure that each requirement has got at least one test case.	In this, we will check whether all the scenarios are covered for the particular requirements.

49) What is the difference between use case and test case?

Following are the significant differences between the use case and the test case:

Test case	Use Case
It is a document describing the input, action, and expected response to control whether the application is working fine based on the customer requirements.	It is a detailed description of Customer Requirements.
It is derived from test scenarios, Use cases, and the SRS.	It is derived from BRS/SRS.
While developing test cases, we can also identify loopholes in the specifications.	A business analyst or QA Lead prepares it.

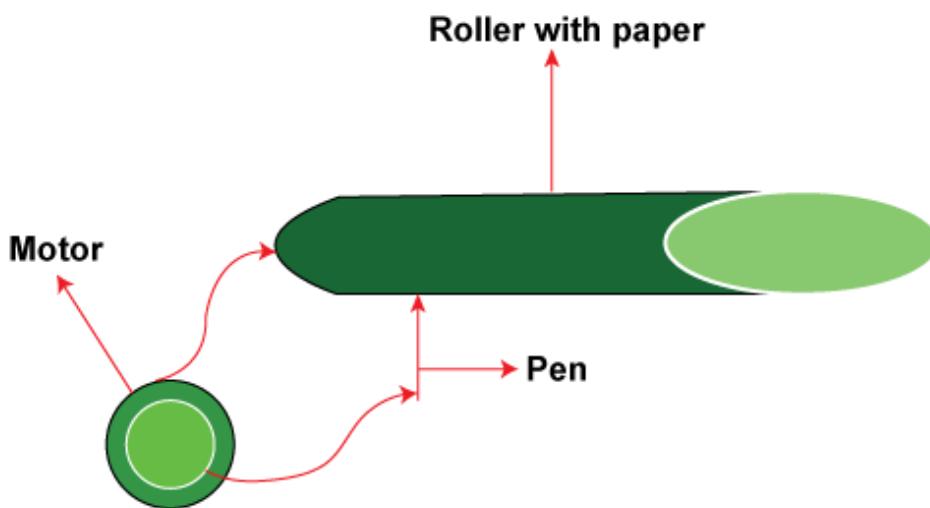
50) How to test a pen?

We can perform both manual and automation testing. First, we will see how we perform manual testing:

Different types of testing	Scenario
Smoke testing	Checks that basic functionality is written or not.
Functional/unit testing	Check that the Refill, pen body, pen cap, and pen size as per the requirement.
Integration testing	Combine pen and cap and integrate other different sizes and see whether they work fine.
Compatibility testing	Various surfaces, multiple environments, weather conditions, and keep it in oven and then write, keep it in the freezer and write, try and write on water.
Adhoc testing	Throw the pen down and start writing, keep it vertically up and write, write on the wall.
Performance testing	Test the writing speed of the pen.
Usability testing	Check whether the pen is user friendly or not, whether we can write it for more extended periods smoothly.
Accessibility testing	Handicapped people use them.
Reliability testing	Drop it down and write, and continuously write and see whether it leaks or not
Recovery testing	Throw it down and write.

Globalization testing	Price should be standard, expiry date format.
Localization testing	
Internationalize testing	Check whether the print on the pen is as per the country language.

Now, we will see how we perform automation testing on a pen:



For this take a roller, now put some sheets of paper on the roller, then connects the pen to the motor and switch on the motor. The pen starts writing on the paper. Once the pen has stopped writing, now observe the number of lines that it has written on each page, length of each track, and multiplying all this, so we can get for how many kilometers the pen can write.

Job/HR Interview Questions	Database Interview Questions
PL/SQL Interview Questions	SQL Interview Questions
Oracle Interview Questions	Android Interview Questions
SQL Server Interview Questions	MySQL Interview Questions
Java Basics Interview Questions	Java OOPs Interview Questions
Java Multithreading Questions	Java String & Exception Questions
Java Collection Interview Questions	JDBC Interview Questions
Servlet Interview Questions	JSP Interview Questions
Spring Interview Questions	Hibernate Interview Questions

You may also like:

- [Java Interview Questions](#)
- [SQL Interview Questions](#)
- [Python Interview Questions](#)
- [JavaScript Interview Questions](#)
- [Angular Interview Questions](#)
- [Selenium Interview Questions](#)
- [Spring Boot Interview Questions](#)
- [HR Interview Questions](#)
- [C Programming Interview Questions](#)
- [C++ Interview Questions](#)

- Data Structure Interview Questions
- DBMS Interview Questions
- HTML Interview Questions
- IAS Interview Questions
- Manual Testing Interview Questions
- OOPs Interview Questions
- .Net Interview Questions
- C# Interview Questions
- ReactJS Interview Questions
- Networking Interview Questions
- PHP Interview Questions
- CSS Interview Questions
- Node.js Interview Questions
- Spring Interview Questions
- Hibernate Interview Questions
- AWS Interview Questions
- Accounting Interview Questions