## 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.

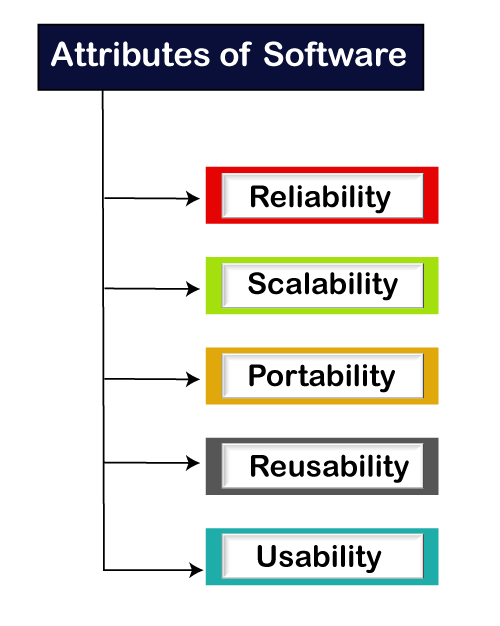
Software testing is the process of finding errors in the developed product. It also checks whether the real outcomes can match expected results, as well as aids in the identification of defects, missing requirements, or gaps.

Testing is the penultimate step before the launch of the product to the market. It includes examination, analysis, observation, and evaluation of different aspects of a product.

Professional [software testers](https://www.testim.io/blog/software-tester-qualities/) use a combination of manual testing with automated tools. After conducting tests, the testers report the results to the development team. The end goal is to deliver a quality product to the customer, which is why software testing is so important.

## **Testing in Software Engineering**

As per ANSI/IEEE 1059, **Testing in Software Engineering** is a process of evaluating a software product to find whether the current software product meets the required conditions or not. The testing process involves evaluating the features of the software product for requirements in terms of any missing requirements, bugs or errors, security, reliability and performance.

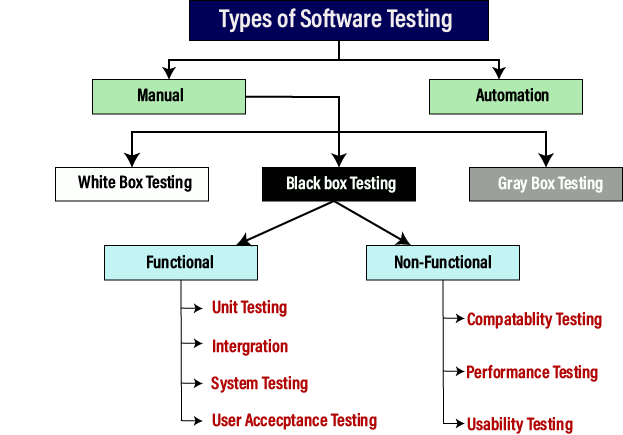


## What is Testing

Testing is a group of techniques to determine the correctness of the application under the predefined script but, testing cannot find all the defects of the 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 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.

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.



### 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 we 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**

Manual testing is the process of testing the software by hand to learn more about it, to find what is and isn’t working.

This usually includes verifying all the features specified in requirements documents, but often also includes the testers trying the software with the perspective of their end users in mind.

Manual test plans vary from fully scripted test cases, giving testers detailed steps and expected results, through to high-level guides that steer exploratory testing sessions.

## **Testing Methods:**

1. Static Testing
2. Dynamic Testing

### **Static Testing:**

It is also known as Verification in Software Testing.

Verification is a static method of checking documents and files. Verification is the process, to ensure that whether we are building the product right i.e., to verify the requirements which we have and to verify whether we are developing the product accordingly or not.

Activities involved here are Inspections, Reviews, Walkthroughs

### **Dynamic Testing:**

It is also known as Validation in Software Testing.

Validation is a dynamic process of testing the real product. Validation is the process, whether we are building the right product i.e., to validate the product which we have developed is right or not.

Activities involved in this is Testing the software application (Desktop application, Web application, Mobile Apps)

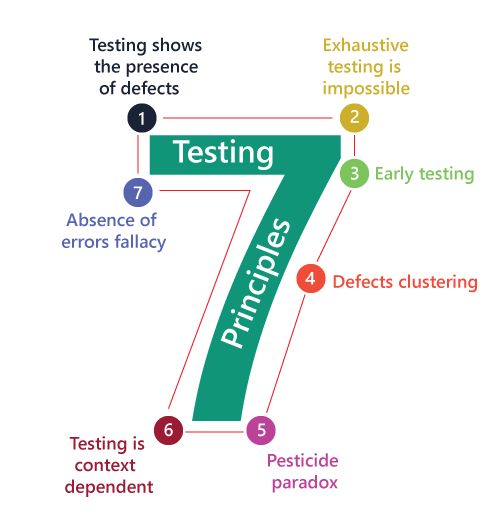
## **Testing Artifacts:**

Test Artifacts are the deliverables that are given to the stakeholders of a software project. A software project which follows SDLC undergoes the different phases before delivering to the customer. In this process, there will be some deliverables in every phase. Some of the deliverables are provided before the testing phase commences and some are provided during the testing phase and the rest after the testing phase is completed.

Some of the test deliverables are as follows:

1. Test plan
2. Test case
3. Traceability matrix
4. Test script
5. Test suite
6. Release Note
7. Test data or Test Fixture
8. Test harness

# 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.

* 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.

Testing shows the presence of defects in the software. The goal of testing is to make the software fail. Sufficient testing reduces the presence of defects. In case testers are unable to find defects after repeated regression testing doesn’t mean that the software is bug-free.

Testing talks about the presence of defects and don’t talk about the absence of defects.

### 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 variation according to the importance of the modules because the product timelines will not permit us to perform such types of testing scenarios. If we keep on testing all possible test conditions then the software execution time and costs will rise. So instead of doing exhaustive testing, risks and priorities will be taken into consideration whilst doing testing and estimating testing efforts.

### 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 defines 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. As per the [Pareto Principle](https://en.wikipedia.org/wiki/Pareto_principle#In_software) (80-20 Rule), 80% of issues comes from 20% of modules and remaining 20% of issues from remaining 80% of modules. So we emphasize testing on the 20% of modules where we face 80% of bugs. With the help of this, we can find the uncertain modules, but this method has its difficulties if the same tests are performed regularly, hence the same test will not be able to identify the new defects.

### Pesticide paradox

This principle defines 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. Testing approach depends on the context of the software we develop. We do test the software differently in different contexts. For example, online banking application requires a different approach of testing compared to an e-commerce site.

### 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 software development. 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 deliverables 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

Software Development Life Cycle

* [Requirement Phase](https://www.javatpoint.com/software-development-life-cycle#requirement-phase)
* [Design Phase](https://www.javatpoint.com/software-development-life-cycle#design-phase)
* [Build /Development Phase](https://www.javatpoint.com/software-development-life-cycle#build-development-phase)
* [Testing Phase](https://www.javatpoint.com/software-development-life-cycle#testing-phase)
* [Deployment/ Deliver Phase](https://www.javatpoint.com/software-development-life-cycle#deployment-deliver-phase)
* [Maintenance](https://www.javatpoint.com/software-development-life-cycle#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 languages** 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 the 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, customers 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 the product from 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 processes or approaches which are being selected for the development of a 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**

# 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:

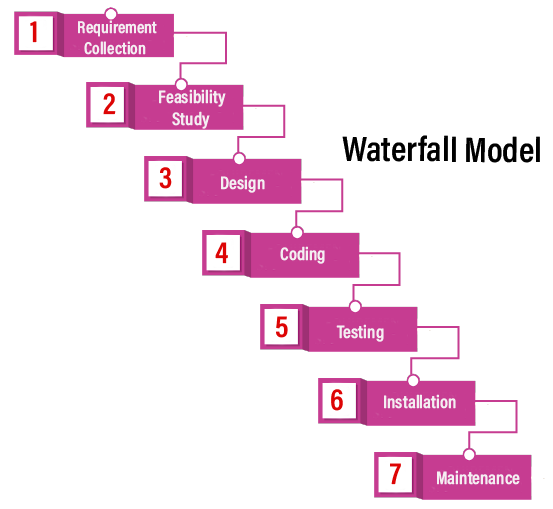
### Requirement Collection

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 operations that are given by the clients? |
| **Economic** | Should the company be able to complete the product within a 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 be handed over to the concerned 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 changed 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 issues get resolved, the application will be deployed for the end-users use.

### Maintenance

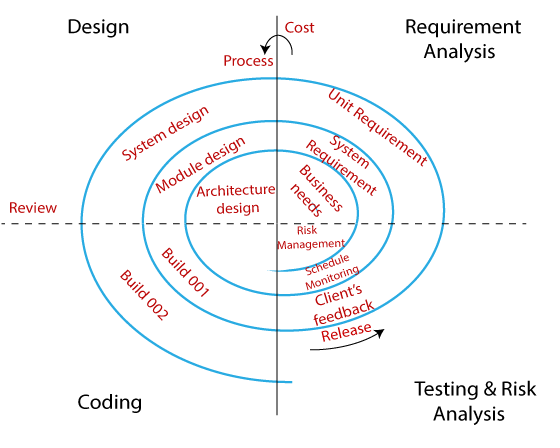
After completing the six stages successfully, we will move to the last stage (maintenance) of the waterfall model. In this, the process will remain until the software comes to an end, where the end-user starts using the application, and they may have some issues that need to be tested and fixed. Taking 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

## 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 it takes a long duration to complete the product, and the software becomes 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 hand it 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 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 designing, where we will plan the logical design, architectural design, flow charts, decision tree, and so on.

### Coding

After the completion 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.

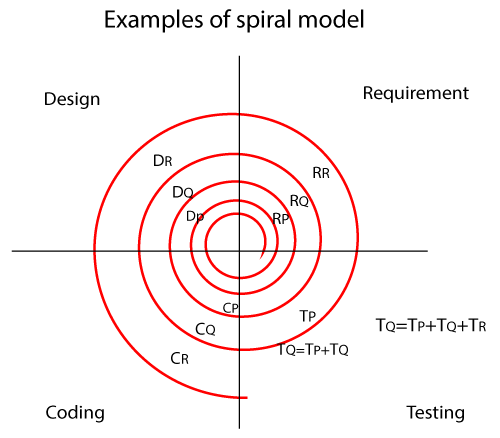
### Testing and Risk Analysis

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.



**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.

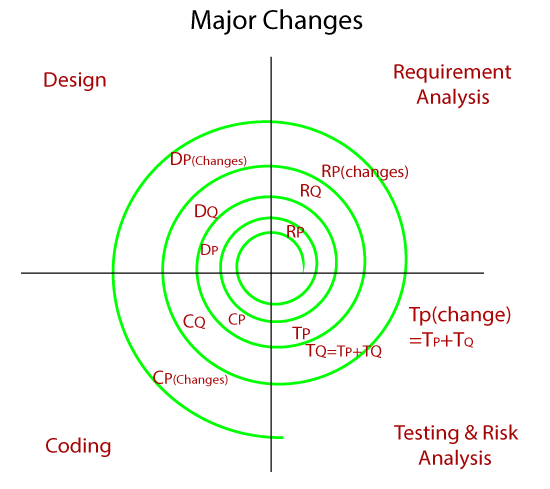
**CP:** Coding of module P, and similarly CQ, CR.

**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

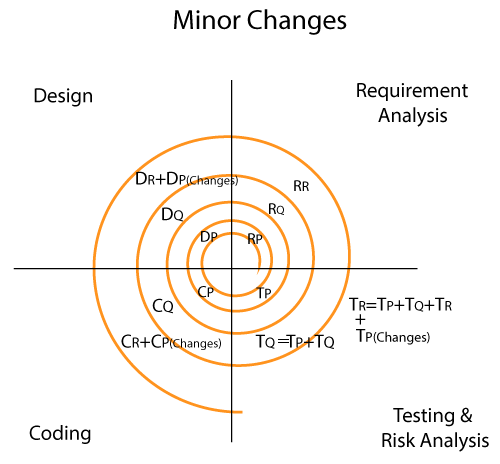
### Major Changes

When the customer requests 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.



### Minor changes

Whenever the client requests for the minor changes in the particular application, then the software team makes the smaller changes along with the new module that 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).



## 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. |

# 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 in the **downward direction**, and at one point in time, it starts moving in the **upward direction** to reuse the test document for the testing process and forms a **V** shape. Hence it is known as the **V model**.

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 |
| 5. | 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 | Username→ 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 |
| 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. It is based on the association of a testing phase for each corresponding development stage. Development of each step directly associated with the testing phase. The next phase starts only after completion of the previous phase i.e. for each development activity, there is a testing activity corresponding to it.

Where the verification and validation process includes different stages:

**Verification:** It involves static analysis technique (review) done without executing code. It is the process of evaluation of the product development phase to find whether specified requirements meet.

**Validation:** It involves dynamic analysis technique (functional, non-functional), testing done by executing code. Validation is the process to evaluate the software after the completion of the development phase to determine whether software meets the customer expectations and requirements.

So V-Model contains Verification phases on one side and the Validation phases on the other side. Verification and Validation phases are joined by coding phases in V-shape. Thus it is called V-Model.

**Design Phase:**

* **Requirement Analysis:** This phase contains detailed communication with the customer to understand their requirements and expectations. This stage is known as Requirement Gathering.
* **System Design:** This phase contains the system design and the complete hardware and communication setup for developing the product.
* **Architectural Design:** System design is broken down further into modules taking up different functionalities. The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood.
* **Module Design:** In this phase the system breaks down into small modules. The detailed design of modules is specified, also known as Low-Level Design (LLD).

**Testing Phases:**

* **Unit Testing:** Unit Test Plans are developed during the module design phase. These Unit Test Plans are executed to eliminate bugs at code or unit level.
* **Integration testing:** After completion of unit testing Integration testing is performed. In integration testing, the modules are integrated and the system is tested. Integration testing is performed on the Architecture design phase. This test verifies the communication of modules among themselves.
* **System Testing:** System testing tests the complete application with its functionality, inter dependency, and communication.It tests the functional and nonfunctional requirements of the developed application.
* **User Acceptance Testing (UAT):** UAT is performed in a user environment that resembles the production environment. UAT verifies that the delivered system meets the user's requirement and the system is ready for use in the real world.

**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**

Once the test engineer team reviews the CRS and finds 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 develops 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.

## Advantage (Pros) of V-Model:

1. Easy to Understand.
2. Testing Methods like planning, test designing happens well before coding.
3. This saves a lot of time. Hence a higher chance of success over the waterfall model.
4. Avoids the downward flow of the defects.
5. Works well for small plans where requirements are easily understood

## Disadvantage (Cons) of V-Model:

1. Very rigid and least flexible.
2. Not a good for a complex project.
3. Software is developed during the implementation stage, so no early prototypes of the software are produced.
4. If any changes happen in the midway, then the test documents along with the required documents, has to be updated.

# **Agile Methodology: What is Agile Software Development Model & Process in Testing?**

Agile Methodology means a practice that promotes **continuous iteration** of development and testing throughout the software development life cycle of the project. In the Agile model in software testing, both development and testing activities are concurrent, unlike the Waterfall model.

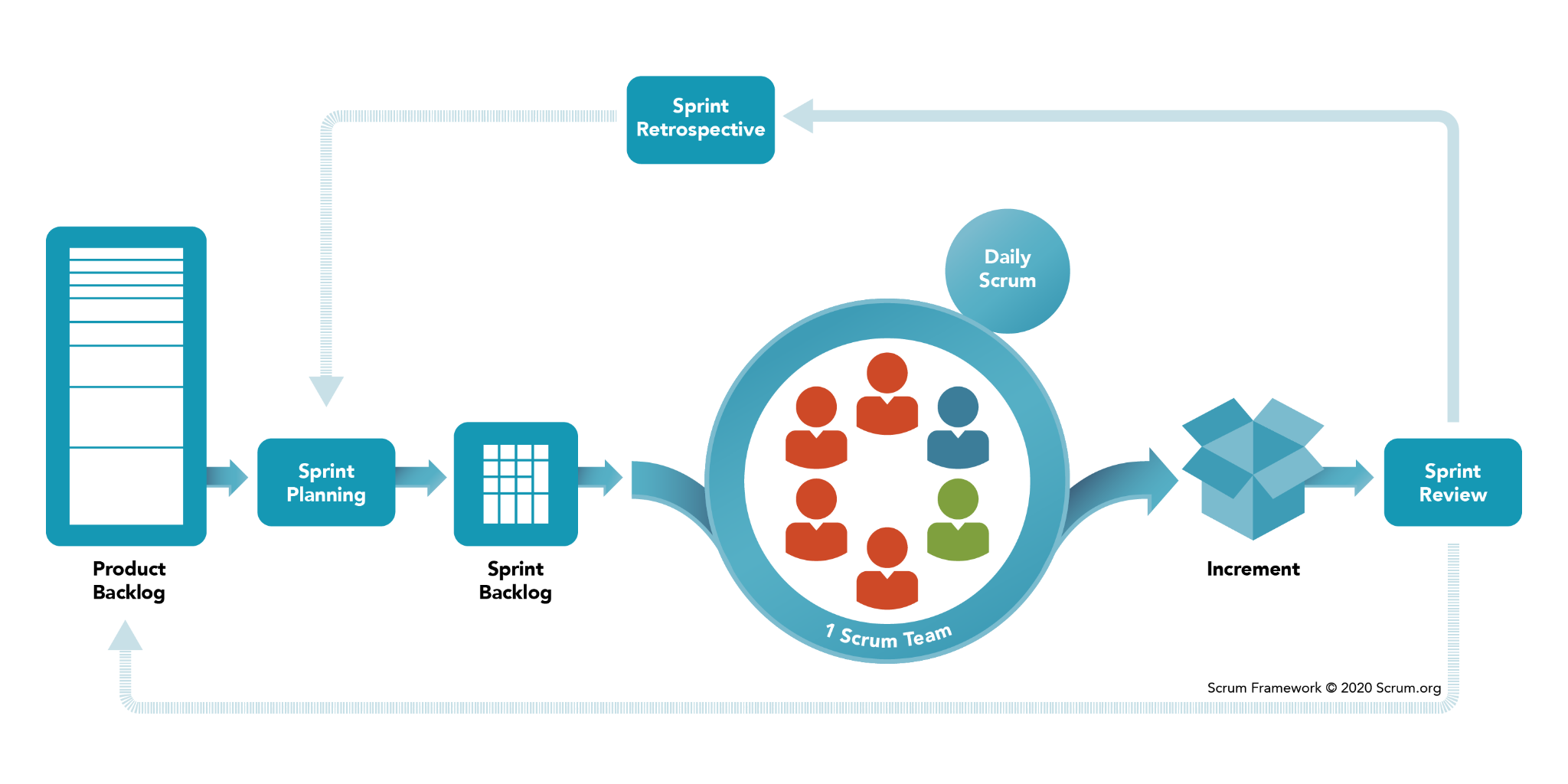
## **What is Agile Software Development?**

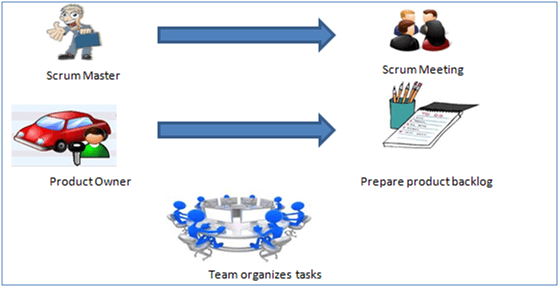
The **Agile software development** methodology is one of the simplest and effective processes to turn a vision for a business need into software solutions. Agile is a term used to describe software development approaches that employ continual planning, learning, improvement, team collaboration, evolutionary development, and early delivery. It encourages flexible responses to change.

Agile software development emphasizes four core values.

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

## **Scrum**



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

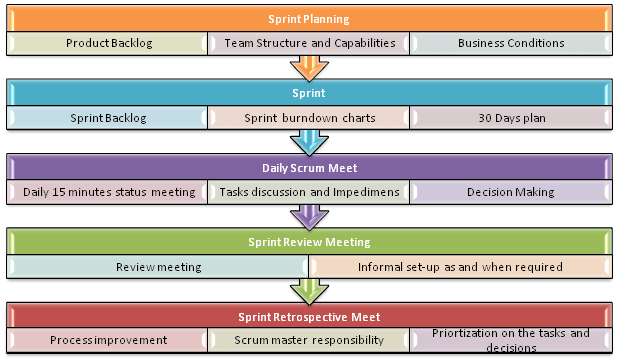
* Scrum Master
  + Master is responsible for setting up the team, sprint meeting and removes obstacles to progress
* Product owner
  + The Product Owner creates product backlog, prioritizes the backlog and is responsible for the delivery of the functionality at each iteration
* Scrum Team
  + Team manages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

## **Product Backlog**

This is a repository where requirements are tracked with details on the number of requirements(user stories) to be completed for each release. It should be maintained and prioritized by the Product Owner, and it should be distributed to the scrum team. Team can also request for a new requirement addition or modification or deletion

## **Scrum Practices**

Practices are described in detailed:



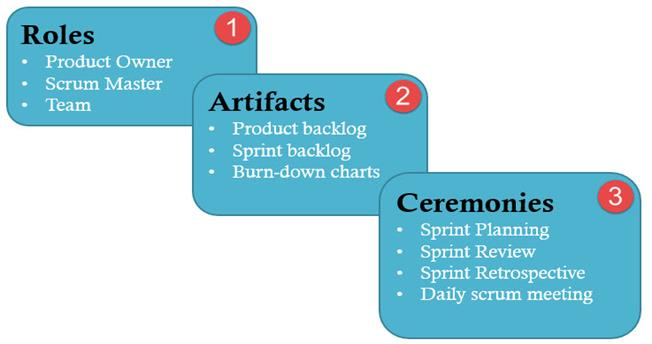
## **Process flow of Scrum Methodologies:**

Process flow of scrum testing is as follows:

* Each iteration of a scrum is known as Sprint
* Product backlog is a list where all details are entered to get the end-product
* During each Sprint, top user stories of Product backlog are selected and turned into Sprint backlog
* Team works on the defined sprint backlog
* Team checks for the daily work
* At the end of the sprint, team delivers product functionality

Business requirements are gathered in terms of stories. All those stories are stored in a place called the parking lot.

In this type of methodology, releases are based on the shorter cycles called Iterations with a span of 14 days time period. Each iteration includes phases like coding, unit testing and system testing where at each phase some minor or major functionality will be built in the application.

A scrum process includes

* **User stories:** They are a short explanation of functionalities of the system under test. Example for Insurance Providers is – “Premium can be paid using the online system.”
* **Product Backlog:** It is a collection of user stories captured for a scrum product. **The product owner prepares** and maintains the product backlog. It is prioritized by the product owner, and anyone can add to it with approval from the product owner.
* **Release Backlog:** A release is a time frame in which the number of iterations is completed. **The product owner coordinates** with the scrum master to decide which stories should be targeted for a release. Stories in the release backlog are targeted to be completed in a release.
* **Sprints:** It is a set period of time to complete the user stories, decided by the product owner and developer team, usually 2-4 weeks of time.
* **Sprint Backlog:** It’s a set of user stories to be completed in a sprint. During the sprint backlog, work is never assigned, and the team signs up for work on their own. It is owned and managed by the team while the estimated work remaining is updated daily. It is the list of task that has to be performed in Sprint
* **Block List:** It is a list of blocks and unmade decisions owned by scrum master and updated daily
* **Burndown chart:** Burn-down chart represents overall progress of the work in progress and work completed throughout the process. It represents in a graph format the stories and features not completed

## **Testing Activities in Scrum**

Testers do following activities during the various stages of Scrum-

**Sprint Planning**

* In sprint planning, a tester should pick a user-story from the product backlog that should be tested.
* As a tester, he/she should decide how many hours (Effort Estimation) it should take **to finish** testing for each of selected user stories.
* As a tester, he/she must know what sprint goals are.
* As a tester, contribute to the prioritizing process

**Sprint**

* Support developers in unit testing
* Test user-story when completed. **Test execution is performed** in a lab where both tester and developer work hand in hand. Defect are logged in Defect Management tool which are tracked on a daily basis. Defects can be conferred and analyzed during the scrum meeting. Defects are retested as soon as it is **resolved** and deployed for testing
* As a tester, he/she attends all daily standup meeting to speak up
* As a tester, he/ she can bring any backlog item that cannot be completed in the current sprint and put to the next sprint
* Tester is responsible for developing automation scripts. He schedules automation testing with Continuous Integration (CI) system. Automation receives the importance due to short delivery timelines. Test Automation can be accomplished by utilizing various open source or paid tools available in the market. This proves effective in ensuring that everything that needs to be tested was covered. Sufficient Test coverage can be achieved with a close communication with the team.
* Review CI automation results and send Reports to the stakeholders
* Executing non-functional testing for approved user stories
* Coordinate with customer and product owner to define acceptance criteria for Acceptance Tests
* At the end of the sprint, the tester also does acceptance testing(UAT) in some case and confirms testing completeness for the current sprint

**Sprint Retrospective**

* As a tester, he will figure out what went wrong and what went right in the current sprint
* As a tester, he identifies lesson learned and best practices

## **Test Reporting**

Scrum Test metrics reporting provides transparency and visibility to stakeholders about the project. The metrics that are reported allow a team to analyze their progress and plan their future strategy to improve the product. There are two metrics that are frequently used to report.

**Burn down chart:** Each day, Scrum Master records the estimated remaining work for the sprint. This is nothing but the Burn Down Chart. It is updated daily.

A burndown chart gives a quick overview of the project progress, this chart contains information like the total amount of work in the project that must be completed, amount of work completed during each sprint and so on.

## Twelve Principles of Agile Manifesto

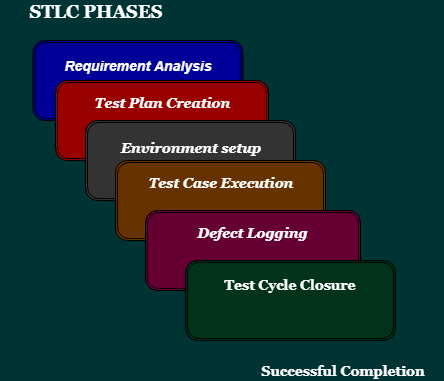
* Customer Satisfaction − Highest priority is given to satisfy the requirements of customers through early and continuous delivery of valuable software.
* Welcome Change − Changes are inevitable during software development. Ever-changing requirements should be welcome, even late in the development phase. Agile processes should work to increase customers' competitive advantage.
* Deliver a Working Software − Deliver a working software frequently, ranging from a few weeks to a few months, considering shorter time-scale.
* Collaboration − Business people and developers must work together during the entire life of a project.
* Motivation − Projects should be built around motivated individuals. Provide an environment to support individual team members and trust them so as to make them feel responsible to get the job done.
* Face-to-face Conversation − Face-to-face conversation is the most efficient and effective method of conveying information to and within a development team.
* Measure the Progress as per the Working Software − Working software is the key and it should be the primary measure of progress.
* Maintain Constant Pace − Agile processes aim towards sustainable development. The business, the developers, and the users should be able to maintain a constant pace with the project.
* Monitoring − Pay regular attention to technical excellence and good design to enhance agility.
* Simplicity − Keep things simple and use simple terms to measure the work that is not completed.
* Self-organized Teams − An agile team should be self-organized and should not depend heavily on other teams because the best architectures, requirements, and designs emerge from self-organized teams.
* Review the Work Regularly − Review the work done at regular intervals so that the team can reflect on how to become more effective and adjust its behavior accordingly.

# 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](https://www.javatpoint.com/software-testing-life-cycle#requirement-analysis)
2. [Test Plan Creation](https://www.javatpoint.com/software-testing-life-cycle#test-plan-creation)
3. [Environment setup](https://www.javatpoint.com/software-testing-life-cycle#environment-setup)
4. [Test case Execution](https://www.javatpoint.com/software-testing-life-cycle#test-case-execution)
5. [Defect Logging](https://www.javatpoint.com/software-testing-life-cycle#defect-logging)
6. [Test Cycle Closure](https://www.javatpoint.com/software-testing-life-cycle#test-cycle-closure)

## Requirement Analysis:

The first step of the manual testing procedure is requirement analysis. In this phase, the tester analyses the 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 andTest 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 specifications.  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, and 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 an 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, and possible defects in order to use these practices in the future if there is software with the same specification.

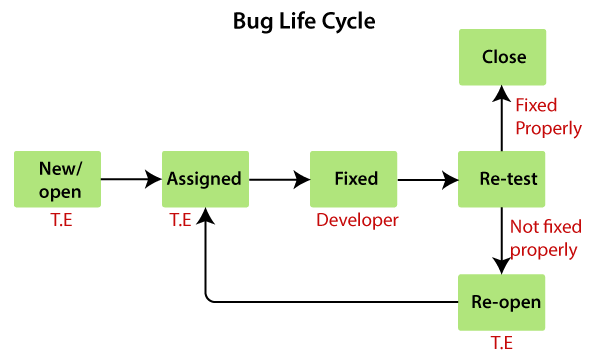
| Entry Criteria | Activities | Deliverable |
| --- | --- | --- |
| All document and reports related to software. | Evaluates the strategy of development, testing procedure, possible defects to use these practices in the future if there is a software with the same specification | Test closure report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

# 

# Bug Life cycle

Defect life cycle, also known as Bug Life cycle is the journey of a defect cycle, which a defect goes through during its lifetime. It varies from organization to organization and also from project to project as it is governed by the software testing process and also depends upon the tools used.

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.

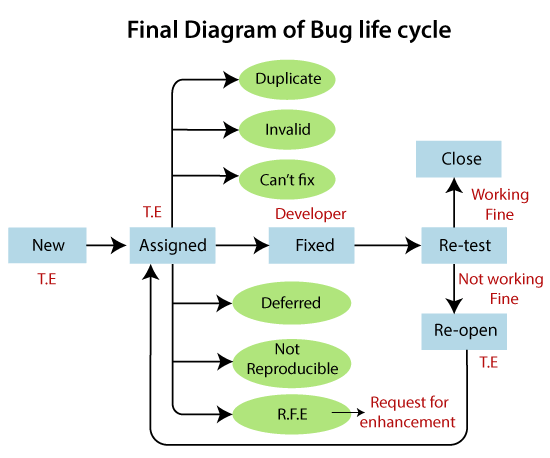
**OR**

**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.

## Another status of the bug

Once we prepare a bug report and send it to the Developers, the Developer will accept the bug and start 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)**

### 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).

### Duplicate

When the same bug has been reported multiple times by the different test engineers are known as a **duplicate** bug.

### Not Reproducible

The Developer accepts the bug, but not able to Reproduce due to some reasons.

**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**

### Can't fix

When the Developer accepts the bug and is 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 did not have 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 the development lead cannot reject the bug.
* **The cost of fixing a bug is more than keeping it.**

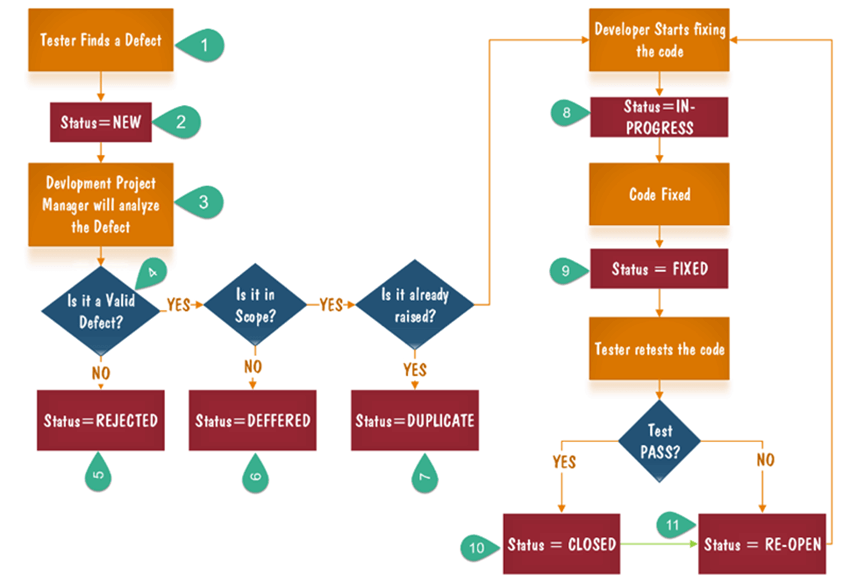
### Deferred / postponed

The deferred/postpone is a status in which the bugs are postponed to the future release due to time constraints.

### 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**.

## **Defect Life Cycle Explained**

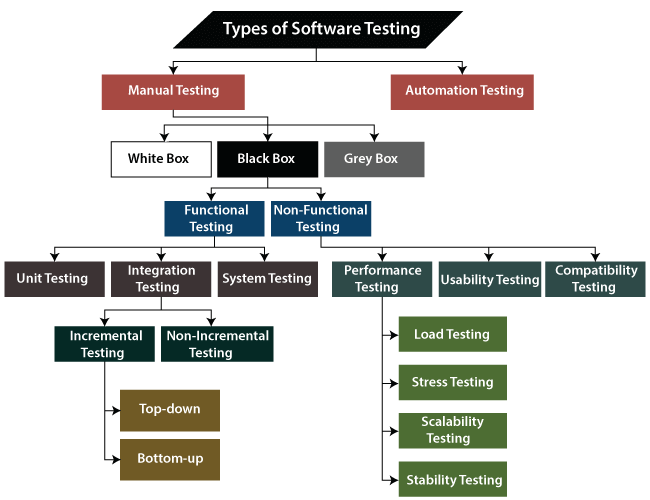
****

* 1. Tester finds the defect
  2. Status assigned to defect- New
  3. A defect is forwarded to Project Manager for analyze
  4. Project Manager decides whether a defect is valid
  5. Here the defect is not valid- a status is given “Rejected.”
  6. So, project manager assigns a status **rejected**. If the defect is not rejected then the next step is to check whether it is in scope. Suppose we have another function- email functionality for the same application, and you find a problem with that. But it is not a part of the current release when such defects are assigned as a **postponed or deferred** status.
  7. Next, the manager verifies whether a similar defect was raised earlier. If yes defect is assigned a status **duplicate**.
  8. If no the defect is assigned to the developer who starts fixing the code. During this stage, the defect is assigned a status **in- progress.**
  9. Once the code is fixed. A defect is assigned a status **fixed**
  10. Next, the tester will re-test the code. In case, the [Test Case](https://www.guru99.com/test-case.html) passes the defect is **closed.** If the test cases fail again, the defect is **re-opened** and assigned to the developer.
  11. Consider a situation where during the 1st release of Flight Reservation a defect was found in Fax order that was fixed and assigned a status closed. During the second upgrade release the same defect again re-surfaced. In such cases, a closed defect will be **re-opened.**

# Types of Software Testing

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.

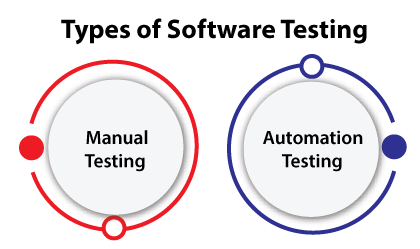


## The different types of Software Testing

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.

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:

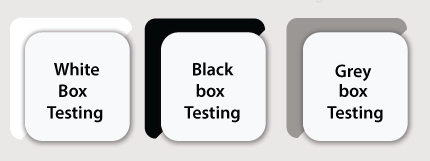
* **Manual Testing**
* **Automation Testing**

## 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.**

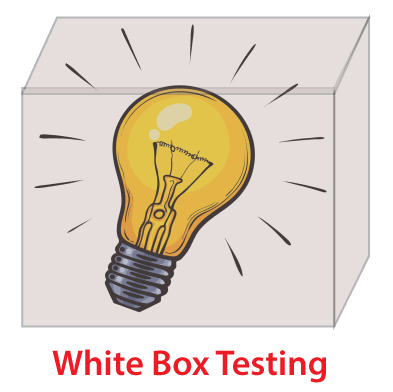
## Classification of Manual Testing

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

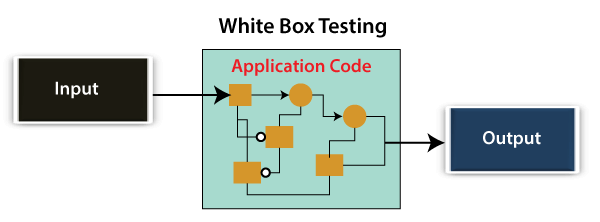
* **White Box Testing**
* **Black Box Testing**
* **Grey Box Testing**

### 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.

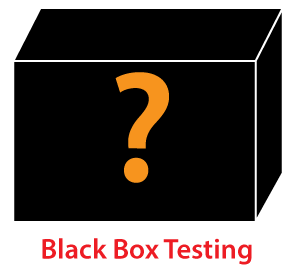
Subsequently, the code is noticeable for developers throughout testing; that's why this process is known as **WBT (White Box Testing)**.

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.

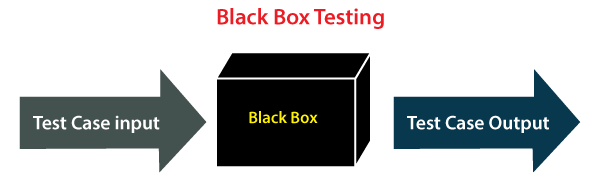
The purpose of implementing the white box testing is to emphasize the flow of inputs and outputs over the software and enhance the security of an application.

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

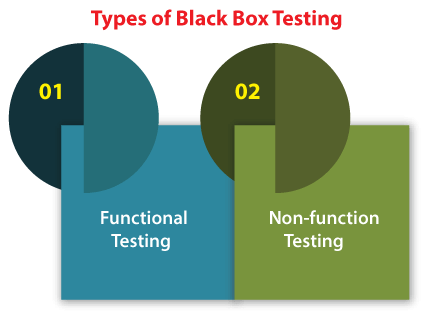
### Black Box Testing

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.

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**.

### **Types of Black Box Testing**

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

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

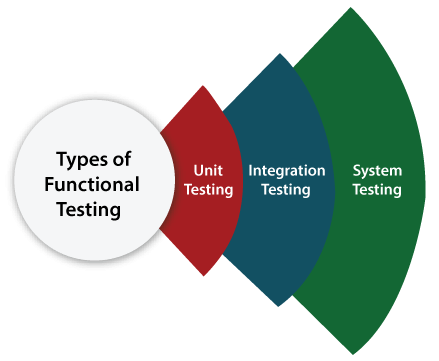
### Functional Testing

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.

### 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**

### 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.

### Integration Testing

Once we are successfully implementing the unit testing, we will go [integration testing](https://www.javatpoint.com/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**

### Incremental Integration Testing

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

**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**

**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**.

### 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.

## Non-function 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.

### Types of Non-functional Testing

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

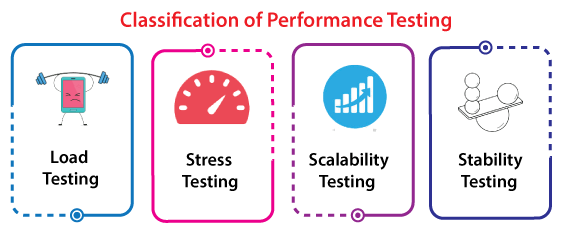
* **Performance Testing**
* **Usability Testing**
* **Compatibility Testing**

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**

**Load Testing**

It is a type of Non-Functional Testing and the objective of Load Testing is to check how much load or maximum workload a system can handle without any performance degradation.

[Load Testing](https://www.softwaretestinghelp.com/introduction-to-performance-testing-loadrunner-training-tutorial-part-1/) helps to find the maximum capacity of the system under specific load and any issues that cause software performance degradation. Load testing is performed using tools like [JMeter](https://www.softwaretestinghelp.com/jmeter-tutorials/), LoadRunner, WebLoad, Silk performer, etc.

**Stress Testing**

This testing is done when a system is stressed beyond its specifications in order to check how and when it fails.

This is performed under heavy load like putting large numbers beyond storage capacity, complex database queries, continuous input to the system or database load.

| Load Testing | Stress Testing |
| --- | --- |
| [Load Testing](https://www.guru99.com/load-testing-tutorial.html) is to test the system behavior under normal workload conditions, and it is just testing or simulating with the actual workload | Stress testing is to test the system behavior under extreme conditions and is carried out till the system failure. |
| Load testing does not break the system | stress testing tries to break the system by testing with overwhelming data or resources. |

**Scalability Testing**

Scalability Testing is a non functional testing method that measures performance of a system or network when the number of user requests are scaled up or down. The purpose of Scalability testing is to ensure that the system can handle projected increase in user traffic, data volume, transaction counts frequency, etc. It tests the system's ability to meet the growing needs.

| Scalability Testing | Load Testing |
| --- | --- |
| * It focusses on the performance of your websites, software, hardware, and application when changes are done in the size or volume of the system to meet a growing need. | * [Load testing](https://www.guru99.com/load-testing-tutorial.html) focusses on testing an application under heavy loads, to determine at what point the system response time fails. |

### 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:

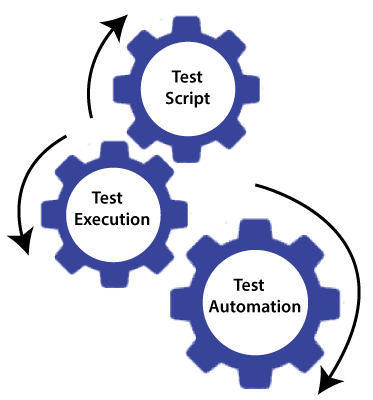
* The application should be easy to understand, which means that all the features must be visible to end-users.
* The application's look and feel should be good. That means the application should be pleasant looking and make a feel to the end-user to use it.

### 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.

## 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.

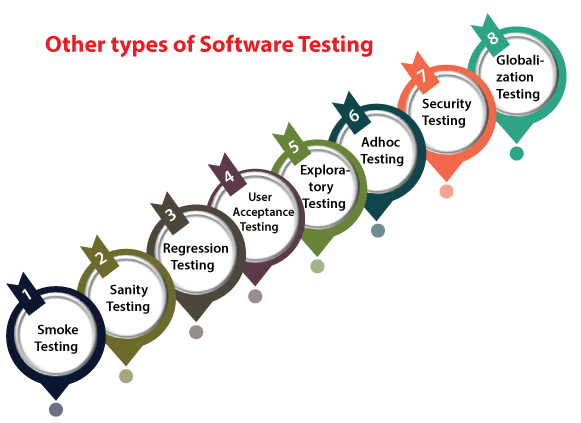
Automation testing is the best way to enhance the efficiency, productivity, and coverage of Software testing.

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, it 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.

## 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**

### smoke testing

### 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.

### 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.

### 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.

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**.

### 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.

### 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 went for 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.

### 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 uses the application casually, he/she may detect a bug. Still, the specialized test engineer uses the software thoroughly, so he/she may not identify a similar detection.

**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.

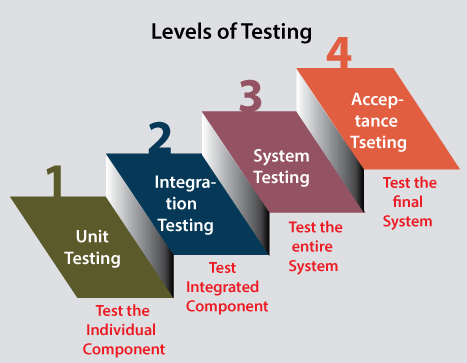
### Globalization Testing

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.

# 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](https://www.javatpoint.com/software-testing-tutorial), we have four different levels of testing, which are as discussed below:

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

### 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**](https://www.javatpoint.com/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 for 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.

### 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.

### Level3: System Testing

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

### 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.

# Performance Testing

*Checking the behavior of an application by applying some load is known as performance testing.*

***Performance Testing*** *is a software testing process used for testing the speed, response time, stability, reliability, scalability and resource usage of a software application under particular workload. The main purpose of performance testing is to identify and eliminate the performance bottlenecks in the software application. It is a subset of performance engineering and also known as “Perf 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 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 are using the application simultaneously for a particular time.*

## Types of Performance Testing

Following are the types of performance testing:

* **Load testing**
* **Stress testing**
* **Scalability testing**
* **Stability testing**

## **1) Load Testing**:

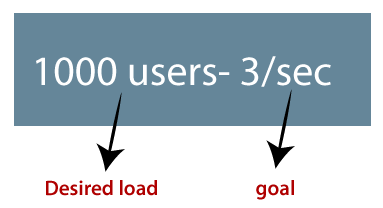
Load Testing is a type of performance testing to check a system with constantly increasing the load on the system until the time load reaches its threshold value. Here Increasing load means increasing number of concurrent users, transactions & check the behavior of application under test. It is normally carried out underneath a controlled environment in order to distinguish between two different systems. It is also called “**Endurance testing**” and “**Volume testing**”. The main purpose of load testing is to monitor the response time and staying power of application when the system is performing well under heavy load.

The successfully executed load testing is only if the specified test cases are executed without any error in allocated time.

Simple examples of load testing:

* Testing printers by sending large jobs.
* Editing a very large document for testing of word processor
* Continuously reading and writing data into the hard disk.
* Running multiple applications simultaneously on the server.
* Testing of mail server by accessing thousands of mailboxes

**For example:** In the below image, **1000 users** is 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.

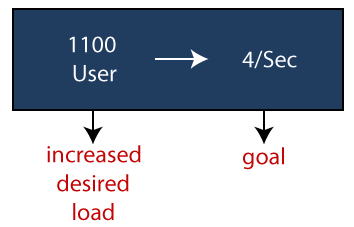


## **2) Stress Testing:**

Stress Testing is a performance testing type to check the stability of software when hardware resources are not sufficient like CPU, memory, disk space etc. Stress testing is Negative testing where we load the software with a large number of concurrent users/processes which cannot be handled by the systems hardware resources. The main idea behind stress testing is to determine the failure of a system and to keep an eye on how the system gracefully recovers, this quality is known as recoverability.

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.



## **3) Spike testing:**

Spike testing is a subset of Stress Testing. A spike test is carried out to validate the performance characteristics when the system under test is subjected to workload models and load volumes that repeatedly increase beyond anticipated production operations for short periods of time.

## **4) Endurance testing**:

Endurance testing involves testing a system with an expected amount of load over a long period of time to find the behavior of the system. Let’s take an example where a system is designed to work for 3 hrs of time but the same system lasts for 6 hrs of time to check the staying power of the system. Most commonly test cases are executed to check the behavior of a system like memory leaks or system fails or random behavior.

**5) Scalability Testing**:

Scalability Testing is a type of non-functional tests and it is the testing of a software application for determining its capability to scale up in terms of any of its non-functional capability like the user load supported, the number of transactions, the data volume etc. The main aim of this testing is to understand at what peak the system prevents more scaling.

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**

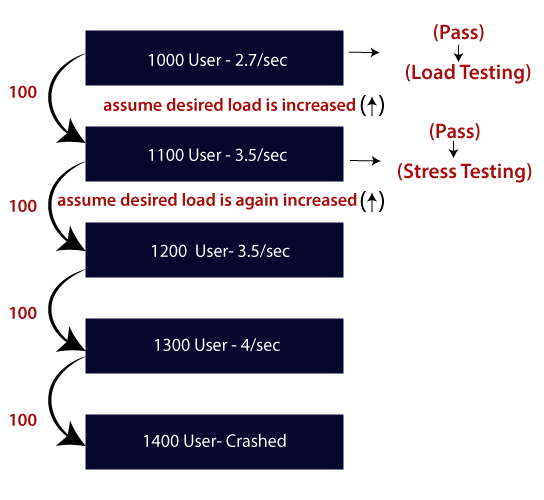
The downward scalability testing is used when the load testing is not passed, then start **decreasing the no. of users in a particular interval** until the goal is achieved. So that it is easy to identify the bottleneck (bug).

## 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 users.
* **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**

****

# 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 is 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.

Testers follow the following steps in the functional testing:

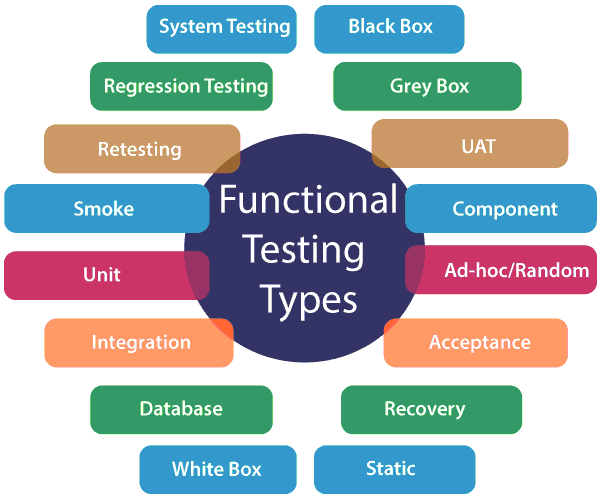
* Tester does verification of the requirement specification in the software application.
* After analysis, the requirement specification tester will make a plan.
* After planning the tests, the tester will design the test case.
* After designing the test case, the tester will make a document of the traceability 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.

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 tests 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 is displayed.

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



**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 is not known to the tester. This type of testing is functional testing. Why we called this type of testing is black-box testing, in the 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 be involved.

**Retesting: Retesting** is a type of testing performed to check the test cases that were unsuccessful in the final execution and successfully pass after the defects are fixed. Usually, testers assign the bug when they find it while testing the product or its component. The bug is 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.

# 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**.

## Unit Testing Tools

We have various types of unit testing tools available in the market, which are as follows:

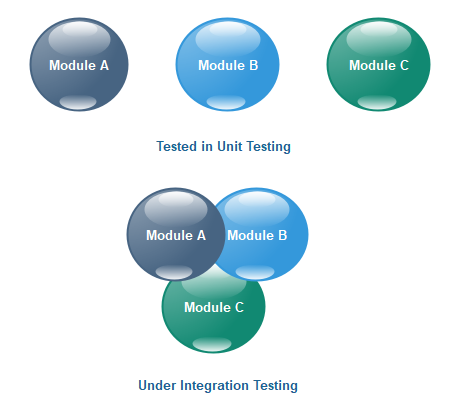
* NUnit
* JUnit
* PHPunit
* Parasoft Jtest
* EMMA

## Unit Testing Techniques:

Unit testing uses all white box testing techniques as it uses the code of software application:

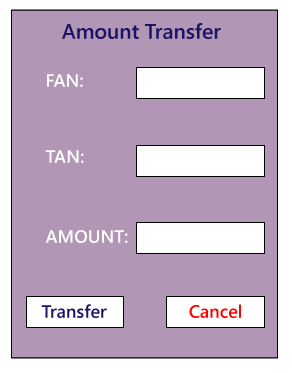
* Data flow Testing
* Control Flow Testing
* Branch Coverage Testing
* Statement Coverage Testing
* Decision Coverage Testing

# Integration testing

Integration testing is the second level of the software testing process 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](https://www.javatpoint.com/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.



* 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**.

## 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

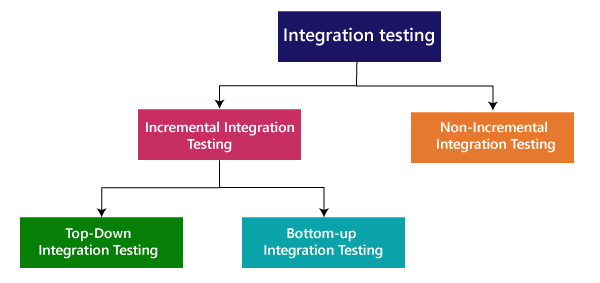
### White Box Testing

* 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.

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

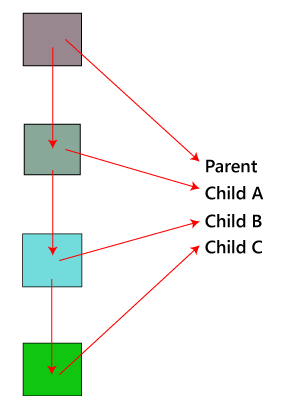
Incremental integration testing is carried out by further methods:

Top-Down approach

Bottom-Up approach

### Top-Down ApproachIntegration testing

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 are 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.

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:

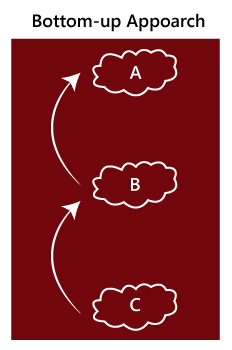
**Advantages:**

* Identification of defect is difficult.
* An early prototype is possible.

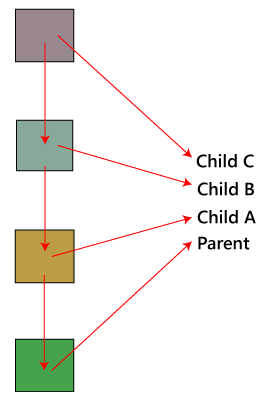
**Disadvantages:**

* Due to the high number of stubs, it gets quite complicated.
* Lower level modules are tested inadequately.
* Critical Modules are tested first so that there are 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.

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.

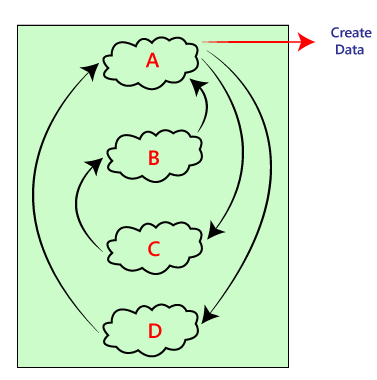
**Disadvantages**

* Critical modules are tested last due to which the defects can occur.
* There is no possibility of an early prototype.

### 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:**

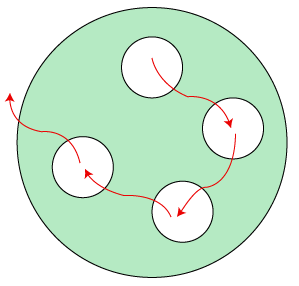
* It is convenient for small size software systems.

**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.

# System Testing

System Testing includes testing of a fully integrated software 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.

**System testing falls under Black box testing** as it includes testing of the external working of the software. Testing follows the 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

## Types of System TestingSystem Testing

System testing is divided into more than 50 types, but software testing companies typically uses some of them. These are listed below:

# 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.

## 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.

## 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

1. Customers are not willing to do that; it defeats the whole point of acceptance testing.
2. If test cases are written by someone else, the customer does not understand them, so the tester has to perform the inspections by themselves only.

# **Operational Acceptance Testing (OAT)**

**Operational Acceptance Testing (OAT)** is a type of [software testing](https://www.geeksforgeeks.org/software-testing-basics/) that is performed to conduct operational pre-release of a software, system or application to check the quality of it. Operational Acceptance Testing is a very usual software testing whose type is non-functional and it is mainly used in software development and software maintenance projects.

Operational Acceptance Testing mainly focuses on the operational readiness of the software and to become part of the production environment.

**Operational Acceptance Testing (OAT)** is a software testing technique which evaluates the operational readiness of a software application prior to the release or production. The goal of Operational acceptance testing is ensuring system and component compliance and smooth system operation in its Standard Operating Environment(SOE).

## **Types of Operational Testing**

* Installation Testing
* Load & Performance Test Operation
* Backup and Restore Testing
* [Security Testing](https://www.guru99.com/what-is-security-testing.html)
* Code Analysis
* Failover Testing
* Recovery Testing
* End-to-End [Test Environment](https://www.guru99.com/test-environment-software-testing.html) Operational Testing
* Operational Documentation Review

# **Confirmation Testing**

**Confirmation testing** is a type of [software testing](https://www.softwaretestingmaterial.com/software-testing/) that allows testers to retest a software product to make sure the previously posted bugs are fixed or not in the system.

Usually, testers report a bug when a test fails. The Dev Team releases a new version of the software after the defect is fixed. Now the testing team will retest to make sure the reported bug is actually fixed or not. This is known as **confirmation testing**.

Confirmation testing is also known as **re-testing**.

Whenever the dev team did some changes in the build to fix a defect then confirmation testing or retesting is done.

It is done repeatedly on every new agile sprint.

It is performed to make sure the test cases which were failed in the earlier build and defects reported are getting passed in the new build or not.

Testers execute the same test cases (which were failed in the old build) to validate the previously reported bugs that have been remediated in the new build. It makes sure the defects reported earlier have been successfully fixed or not. If those issues are fixed then testers mark those bugs as fixed in the Bug Tracking System.

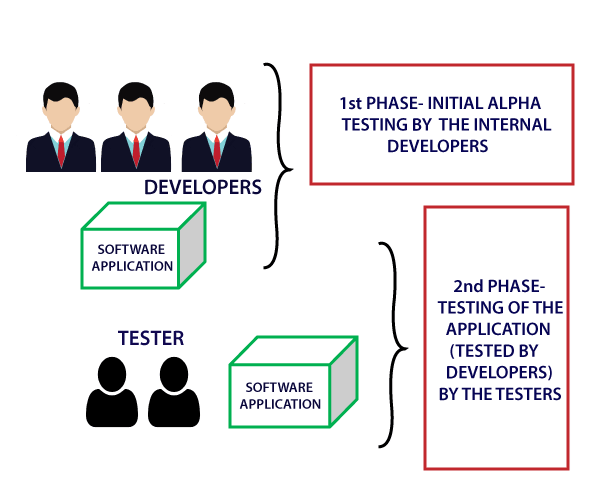
# Alpha Testing Introduction

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 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.

**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 performing the alpha testing by involving black box and white box techniques.

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.

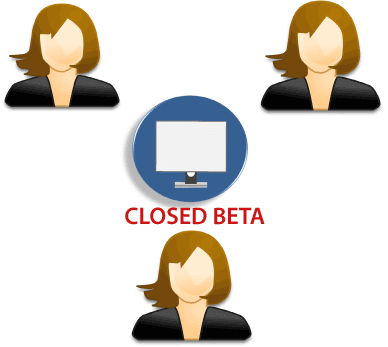
# Beta Testing Introduction

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.

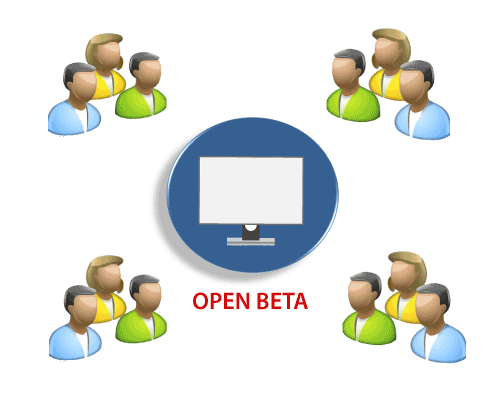
**Features of beta testing are:**

* 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.

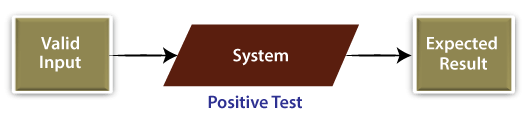


**2) Open beta version:** Open beta is also known as a public beta. The open beta opened to the public. Any user as a tester can assess the beta version to provide the relevant feedback and reviews. Open beta version improves the quality of the final release. This version helps to find the various undetected errors and issues.

***The beta testing process orients this beta version.***

# Positive Testing

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.

Usually, positive testing is implemented to make sure that the particular application or the software product meets the client's specifications and prospects.

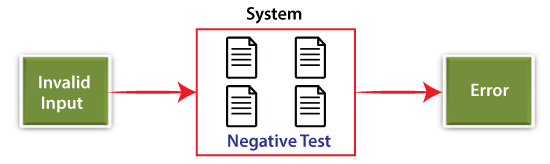
In other words, we can state that positive testing is mainly used to help the test engineer to check whether the software is working as expected by using positive inputs or not.

Concisely, positive testing is used to test the software or an application precisely what it's meant to perform.

# 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 affect 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 ensure 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.

# **END-To-END Testing Tutorial: What is E2E Testing**

End to end testing (E2E testing) refers to a software testing method that involves testing an application’s workflow from beginning to end. This method basically aims to replicate real user scenarios so that the system can be validated for integration and data integrity.

Essentially, the test goes through every operation the application can perform to test how the application communicates with hardware, network connectivity, external dependencies, databases, and other applications. Usually, E2E testing is executed after functional and system testing is complete.

End-to-end testing is a technique used to test whether the flow of an application right from start to finish is behaving as expected. The purpose of performing end-to-end testing is to identify system dependencies and to ensure that the data integrity is maintained between various system components and systems.

The entire application is tested for critical functionalities such as communicating with the other systems, interfaces, database, network, and other applications.

## **Why End to End Testing?**

**End To End Testing** verifies complete system flow and increases confidence by detecting issues and increasing [Test Coverage](https://www.guru99.com/test-coverage-in-software-testing.html) of subsystems. Modern software systems are complex and interconnected with multiple subsystems that may differ from current systems. The whole system can collapse by failure of any subsystem that is a major risk which can be avoided by End-to-End testing.

## **End to End Testing Vs System Testing**

| End to End Testing | System Testing |
| --- | --- |
| Validates the software system as well as interconnected sub-systems | Validates just the software system as per the requirements specifications. |
| It checks the complete end-to-end process flow. | It checks system functionalities and features. |
| All interfaces, backend systems will be considered for testing | Functional and Non-Functional Testing will be considered for testing |
| It’s executed once System Testing is completed. | It’s executed after [Integration Testing](https://www.guru99.com/integration-testing.html). |
| End to End testing involves checking external interfaces which can be complex to automate. Hence [Manual Testing](https://www.guru99.com/manual-testing.html) is preferred. | Both Manual and Automation can be performed for System Testing |

# **Maintenance Testing**

Maintenance Testing is done on the already deployed software. The deployed software needs to be enhanced, changed or migrated to other hardware. The Testing done during this enhancement, change and migration cycle is known as maintenance testing.

Once the software is deployed in operational environment it needs some maintenance from time to time in order to avoid system breakdown, most of the banking software systems needs to be operational 24\*7\*365. So it is very necessary to do maintenance testing of software applications.

**Why is Maintenance Testing required**

Users may need some more new features in the existing software which requires modifications to be done in the existing software and these modifications need to be tested.

End users might want to migrate the software to other latest hardware platforms or change the environment like OS version, Database version etc. which requires testing the whole application on new platforms and environments.

# 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](https://www.javatpoint.com/bug-in-software-testing) fixes, or any change in the existing feature.

Regression testing is a type of [software testing](https://www.javatpoint.com/software-testing-tutorial). 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](https://www.javatpoint.com/test-case) are required to execute many times and running the same test case again and again manually, is time-consuming and tedious too.

## 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 is 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 was previously applicable.

**3. When the defect fixed**

**Example:**

Assume the 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 is fixed by developers, testers test it to make sure Login Button is working as per the expected result. Simultaneously, the 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.

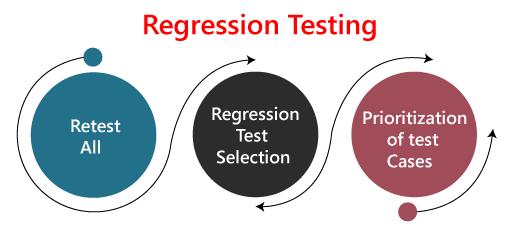
**5. When there is an environment change**

**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 the defect is fixed. In this case, we will need to execute the test again to confirm that the fault is 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:**

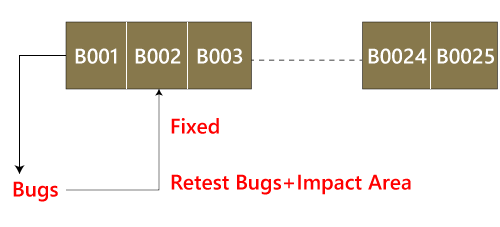
* In this technique, a selected test-case suit will execute rather than an entire test-case suit.
* The selected test case suits divided in two cases
  1. Reusable Test cases.
  2. Obsolete Test cases.
* Reusable test cases can be used in succeeding regression cycles.
* Obsolete test cases can't be used in succeeding regression cycle.

**3. Prioritization of test cases:**

Prioritize the test case depending on business impact, critical and frequently used functionality. Selection of test cases will reduce the regression test suite.

### Regression testing across the builds

Whenever the bug is fixed, we retest the Bug, and if there is any dependent module, we go for a Regression Testing.



| 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 are 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 do 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 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 makes sure that the original fault is corrected. | Regression testing checks for unexpected side effects. |
| 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 be done before starting testing. | Regression testing can obtain test cases from the functional specification, user tutorials and manuals, and defect reports in regards to the corrected problem. |

# What is re-testing?

Re-testing is executing a previously failed test against new software to check if the problem is resolved. After a defect has been fixed, re-testing is performed to check the scenario under the same environmental conditions.

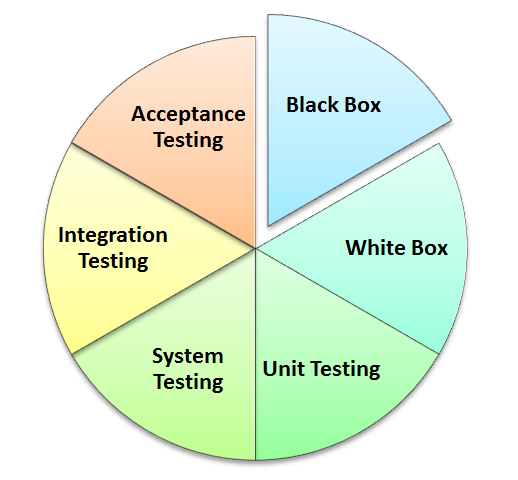
During Re-testing, testers look for granular details at the changed area of functionality, whereas regression testing covers all the main functions to ensure that no functionalities are broken due to this change.

**Retesting** is a process to check specific test cases that are found with bug/s in the final execution. Generally, testers find these bugs while testing the software application and assign it to the developers to fix it. Then the developers fix the bug/s and assign it back to the testers for verification. This continuous process is called Retesting.

# **Manual Testing**

Manual testing is a testing process that is carried out manually in order to find defects without the usage of tools or automation scripting.

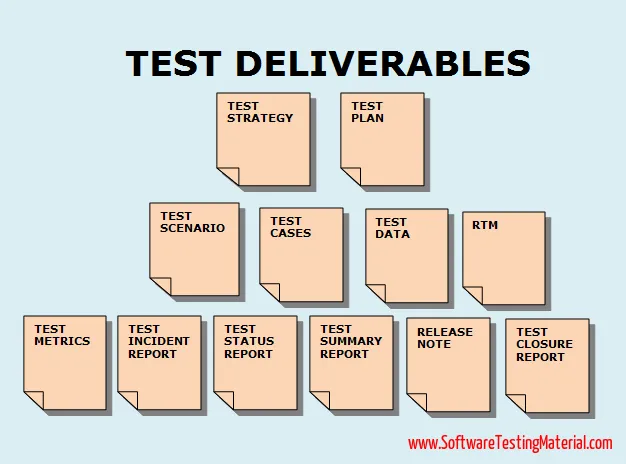
Following are the testing techniques that are performed manually during the test life cycle:

* Acceptance Testing
* White Box Testing
* Black Box Testing
* Unit Testing
* System Testing
* Integration Testing

## **Tools to Automate Manual Testing**

* [Selenium](https://www.guru99.com/selenium-tutorial.html)
* [QTP](https://www.guru99.com/quick-test-professional-qtp-tutorial.html)
* [Jmeter](https://www.guru99.com/jmeter-tutorials.html)
* [Loadrunner](https://www.guru99.com/loadrunner-v12-tutorials.html)
* [TestLink](https://www.guru99.com/testlink-tutorial-complete-guide.html)
* [Quality Center(ALM)](https://www.guru99.com/hp-alm-free-tutorial.html)

# **Test Deliverables in Software Testing**

Test Deliverables are the test artifacts which are given to the stakeholders of a software project during the [SDLC (Software Development Life Cycle)](https://www.softwaretestingmaterial.com/sdlc-software-development-life-cycle/). A software project which follows SDLC undergoes the different phases before delivering to the customer. In this process, there will be some deliverables in every phase. Some of the deliverables are provided before the testing phase commences and some are provided during the testing phase and rest after the testing phase is completed.

Some of the test deliverables are as follows:

1. Test plan
2. Test case
3. Traceability matrix
4. Test script
5. Test suite
6. Release Note
7. Test data or Test Fixture
8. Test harness

# 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 documents which we used Infrequently.

## Features of Test Strategy Document

In [SDLC (Software Development Life Cycle)](https://www.javatpoint.com/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 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 a test strategy document.

## **Difference Between Test Strategy and Test Plan**

| **Test Plan** | **Test Strategy** |
| --- | --- |
| * A test plan for software project can be defined as a document that defines the scope, objective, approach and emphasis on a software testing effort | * Test strategy is a set of guidelines that explains test design and determines how testing needs to be done |
| * Components of Test plan include- Test plan id, features to be tested, test techniques, testing tasks, features pass or fail criteria, test deliverables, responsibilities, and schedule, etc. | * Components of Test strategy includes- objectives and scope, documentation formats, test processes, team reporting structure, client communication strategy, etc. |
| * Test plan is carried out by a testing manager or lead that describes how to test, when to test, who will test and what to test | * A test strategy is carried out by the project manager. It says what type of technique to follow and which module to test |
| * Test plan narrates about the specification | * Test strategy narrates about the general approaches |
| * Test plan can change | * Test strategy cannot be changed |
| * Test planning is done to determine possible issues and dependencies in order to identify the risks. | * It is a long-term plan of action.You can abstract information that is not project specific and put it into test approach |
| * A test plan exists individually | * In smaller project, test strategy is often found as a section of a test plan |
| * It is defined at project level | * It is set at organization level and can be used by multiple projects |

# **TEST PLAN**

A **Test Plan** is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product. Test Plan helps us determine the effort needed to validate the quality of the application under test. The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager.

As per ISTQB definition: “Test Plan is A document describing the scope, approach, resources, and schedule of intended test activities.”

## **What is the Importance of Test Plan?**

Making Test Plan document has multiple benefits

* Help people outside the test team such as developers, business managers, customers **understand** the details of testing.
* Test Plan **guides** our thinking. It is like a rule book, which needs to be followed.
* Important aspects like test estimation, test scope, [Test Strategy](https://www.guru99.com/how-to-create-test-strategy-document.html) are **documented** in Test Plan, so it can be reviewed by Management Team and re-used for other projects.

## **How to write a Test Plan**

You already know that making a **Test Plan** is the most important task of Test Management Process. Follow the seven steps below to create a test plan as per IEEE 829

1. Analyze the product
2. Design the Test Strategy
3. Define the Test Objectives
4. Define Test Criteria
5. Resource Planning
6. Plan Test Environment
7. Schedule & Estimation
8. Determine Test Deliverables

### **Step 1) Analyze the product**

How can you test a product **without** any information about it? The answer is **Impossible.** You must learn a product **thoroughly** before testing it.

The product under test is Guru99 banking website. You should research clients and the end users to know their needs and expectations from the application

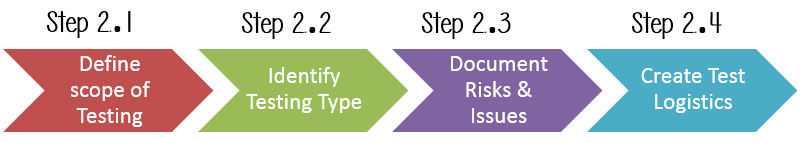
* Who will use the website?
* What is it used for?
* How will it work?
* What are software/ hardware the product uses?

### **Step 2) Develop Test Strategy**

Test Strategy is a **critical step** in making a Test Plan in Software Testing. A Test Strategy document, is a high-level document, which is usually developed by Test Manager. This document defines:

* The project’s **testing objectives** and the means to achieve them
* Determines testing **effort** and **costs**

Back to your project, you need to develop Test Strategy for testing that banking website. You should follow steps below



### **Step 2.1) Define Scope of Testing**

Before the start of any test activity, scope of the testing should be known. You must think hard about it.

* The components of the system to be tested (hardware, software, middleware, etc.) are defined as “**in scope**“
* The components of the system that will not be tested also need to be clearly defined as being “**out of scope**.”

***How do you determine scope your project?***

To determine scope, you must –

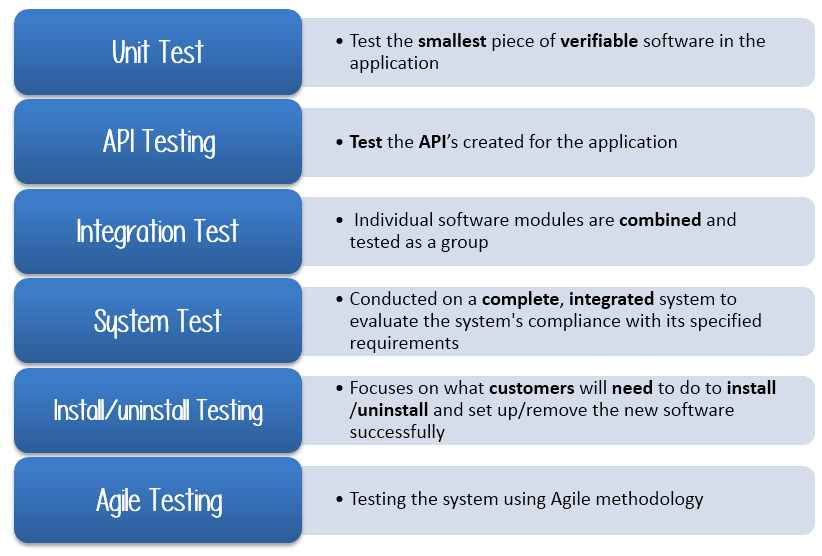
* Precise customer requirement
* Project Budget
* Product Specification
* Skills & talent of your test team

### **Step 2.2) Identify Testing Type**

A **Testing Type** is a standard test procedure that gives an expected test outcome.

Each testing type is formulated to identify a specific type of product bugs. But, all Testing Types are aimed at achieving one common goal “**Early detection of** all the defects before releasing the product to the customer”

The **commonly used** testing types are described as following figure



There are **tons of Testing Types** for testing software products. Your team **cannot make** enough efforts to handle all kinds of testing. As Test Manager, you must set **priority** of the Testing Types

### **Step 2.3) Document Risk & Issues**

Risk is future’s **uncertain event** with a probability of **occurrence** and a **potential** for loss. When the risk actually happens, it becomes the ‘**issue’.**

In the article [Risk Analysis and Solution](https://www.guru99.com/how-precaution-becomes-cure-risk-analysis-and-solutions-in-test-management.html), you have already learned about the ‘Risk’ analysis in detail and identified potential risks in the project.

In the QA Test Plan, you will document those risks

| **Risk** | **Mitigation** |
| --- | --- |
| Team member lack the required skills for website testing. | Plan **training course** to skill up your members |
| The project schedule is too tight; it’s hard to complete this project on time | Set **Test Priority** for each of the test activity. |
| Test Manager has poor management skill | Plan **leadership training** for manager |
| A lack of cooperation negatively affects your employees’ productivity | **Encourage** each team member in his task, **and inspire** them to greater efforts. |
| Wrong budget estimate and cost overruns | Establish the **scope** before beginning work, pay a lot of attention to project planning and constantly track and measure the progress |

### **Step 2.4) Create Test Logistics**

In Test Logistics, the Test Manager should answer the following questions:

* **Who** will test?
* **When** will the test occur?

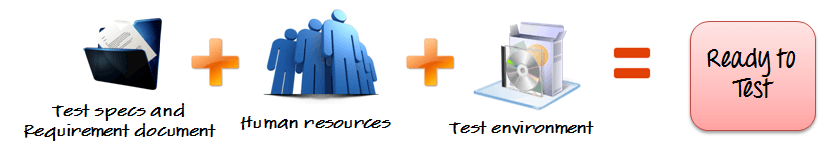
**Who will test?**

You may not know exact names of the tester who will test, but the **type of tester** can be defined.

**When will the test occur?**

Test activities must be matched with associated development activities.

You will start to test when you have **all required items** shown in following figure



### **Step 3) Define Test Objective**

Test Objective is the overall goal and achievement of the test execution. The objective of the testing is finding as many software defects as possible; ensure that the software under test is **bug free** before release.

To define the test objectives, you should do 2 following steps

1. List all the software features (functionality, performance, GUI…) which may need to test.
2. Define the **target** or the **goal** of the test based on above features

### **Step 4) Define Test Criteria**

Test Criteria is a standard or rule on which a test procedure or test judgment can be based. There Are 2 types of test criteria as following

### **Suspension Criteria**

Specify the critical suspension criteria for a test. If the suspension criteria are met during testing, the active test cycle will be **suspended** until the criteria are **resolved**.

Test Plan Example: If your team members report that there are **40%** of test cases failed, you should **suspend** testing until the development team fixes all the failed cases.

### **Exit Criteria**

It specifies the criteria that denote a **successful** completion of a test phase. The exit criteria are the targeted results of the test and are necessary before proceeding to the next phase of development. Example: **95%** of all critical test cases must pass.

### **Step 5) Resource Planning**

Resource plan is a **detailed summary** of all types of resources required to complete project task. Resource could be human, equipment and materials needed to complete a project

The resource planning is important factor of the test planning because helps in **determining** the **number** of resources (employee, equipment…) to be used for the project. Therefore, the Test Manager can make the correct schedule & estimation for the project.

### **Step 6) Plan Test Environment**

### **What is the Test Environment**

A testing environment is a setup of software and hardware on which the testing team is going to execute test cases. The test environment consists of **real business** and **user** environment, as well as physical environments, such as server, front end running environment.

### **How to setup the Test Environment**

Back to your project, how do you set up **test environment** for this banking website?

To finish this task, you need **a strong cooperation** between Test Team and Development Team

You should ask the developer some questions to understand the web application under test **clearly**. Here’re some recommended questions. Of course, you can ask the other questions if you need.

* What is the maximum user connection which this website can handle at the same time?
* What are hardware/software requirements to install this website?
* Does the user’s computer need any particular setting to browse the website?

### **Step 7) Schedule & Estimation**

In the article [Test estimation](https://www.guru99.com/an-expert-view-on-test-estimation.html), you already used some techniques to estimate the effort to complete the project. Now you should include that estimation as well as the schedule to the Test Planning

In the Test Estimation phase, suppose you break out the whole project into small tasks and add the estimation for each task as below

| **Task** | **Members** | **Estimate effort** |
| --- | --- | --- |
| **Create the test specification** | Test Designer | 170 man-hour |
| **Perform Test Execution** | Tester, Test Administrator | 80 man-hour |
| **Test Report** | Tester | 10 man-hour |
| **Test Delivery** |  | 20 man-hour |
| **Total** |  | **280 man-hour** |

### **Step 8) Test Deliverables**

Test Deliverables is a list of all the documents, tools and other components that has to be developed and maintained in support of the testing effort.

## Login Scenarios:

## **UI Test Scenarios**

1. Verify that all the controls including text-boxes, buttons, and links are present on the Login page.
2. Verify that the font specifications of the labels and the text written on the different elements should be clear.
3. Verify that the size, color, and UI of the different elements are as per the specifications.
4. Verify that the application’s UI is responsive i.e. it should adjust to different screen resolutions and devices.

## **Functional Test Scenarios**

1. Verify that as soon as the login page opens, by default the cursor should remain on the username textbox.
2. Verify that the user is able to navigate or access the different controls by pressing the ‘Tab’ key on the keyboard.
3. Verify that the password is in masked form when entered.
4. Verify if the password can be copy-pasted or not.
5. Verify that the user is able to login by entering valid credentials and clicking on the ‘Login’ button.
6. Verify that the user is able to login by entering valid credentials and pressing Enter key.
7. Verify that the user is not able to login with an invalid username and password.
8. Verify that the validation message gets displayed in case the user leaves the username or password field as blank.
9. Verify that the validation message is displayed in the case the user exceeds the character limit of the user name and password fields.
10. Verify that reset button functionality on the login page. Clicking on it should clear the textbox’s content.
11. Verify if there is a checkbox with the label “remember password” on the login page.
12. Verify that closing the browser should not log-out an authenticated user. Launching the application should lead the user to login state only.

## **Security Test Scenarios**

1. Verify the limit on the total number of unsuccessful login attempts.
2. Verify that in case of incorrect credentials, a message like “incorrect username or password” should get displayed instead of an exact message pointing to the incorrect field. Because a message like “incorrect password” will give a hint to the hacker that the username is correct and he just needs to try a different combination on the password field only.
3. Verify the login session timeout duration.
4. Verify that clicking the back button doesn’t logout the user once the user is logged in.

# **How to Write Test Cases: Sample Template with Examples**

A **TEST CASE** is a set of actions executed to verify a particular feature or functionality of your software application. A Test Case contains test steps, test data, precondition, postcondition developed for specific test scenario to verify any requirement. The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the customer.

### [**Test Scenario Vs Test Case**](https://www.guru99.com/test-case-vs-test-scenario.html)

Test scenarios are rather vague and cover a wide range of possibilities. Testing is all about being very specific.

## **The format of Standard Test Cases**

Below is a format of a standard login Test cases example.

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Pass/Fail** |
| --- | --- | --- | --- | --- | --- | --- |
| TU01 | Check Customer Login with valid Data | 1. Go to site http://demo.guru99.com 2. Enter UserId 3. Enter Password 4. Click Submit | Userid = guru99 Password = pass99 | User should Login into an application | As Expected | Pass |
| TU02 | Check Customer Login with invalid Data | 1. Go to site http://demo.guru99.com 2. Enter UserId 3. Enter Password 4. Click Submit | Userid = guru99 Password = glass99 | User should not Login into an application | As Expected | Pass |

# **Test Scenarios Registration Form**

1. Verify that the Registration form contains Username, First Name, Last Name, Password, Confirm Password, Email Id, Phone number, Date of birth, Gender, Location, Terms of use, Submit, Login (If you already have an account)
2. Verify that tab functionality is working properly or not
3. Verify that Enter/Tab key works as a substitute for the Submit button
4. Verify that all the fields such as Username, First Name, Last Name, Password and other fields have a valid placeholder
5. Verify that the labels float upward when the text field is in focus or filled (In case of floating label)
6. Verify that all the required/mandatory fields are marked with \* against the field
7. Verify that clicking on submit button after entering all the mandatory fields, submits the data to the server
8. Verify that the system generates a validation message when clicking on the submit button without filling all the mandatory fields.
9. Verify that entering blank spaces on mandatory fields lead to validation error
10. Verify that clicking on submit button by leaving optional fields, submits the data to the server without any validation error
11. Verify that case sensitivity of Username (usually Username field should not follow case sensitivity – ‘rajkumar’ & ‘RAJKUMAR’ acts same)
12. Verify that system generates a validation message when entering existing username
13. Verify that the character limit in all the fields (mainly username and password) based on business requirement
14. Verify that the username validation as per business requirement (in some application, username should not allow numeric and special characters)
15. Verify that the validation of all the fields are as per business requirement
16. Verify that the date of birth field should not allow the dates greater than current date (some applications have age limit of 18 in that case you have to validate whether the age is greater than or equal to 18 or not)
17. Verify that the validation of email field by entering incorrect email id
18. Verify that the validation of numeric fields by entering alphabets and characters
19. Verify that the “terms and conditions” checkbox is unselected by default (depends on business logic, it may be selected or unselected)
20. Verify that the validation message is displayed when clicking on submit button without selecting “terms and conditions” checkbox
21. Verify that the password is in encrypted form when entered
22. Verify whether the password and confirm password are same or not

# **Test Cases for ATM**

### UI test cases

1. Verify that power backup is present at ATM
2. Verify that card reader is present at ATM
3. Verify that receipt printer is present and receipt roll is available and working
4. Verify that cash dispenser is present and working
5. Verify that keys and keypad is working and covered
6. Verify that buttons on ATM screen is present and working
7. Verify that font on the screen is clear and visible

### Functionality test cases

1. When card is inserted in ATM, pin should be asked from users
2. When user enters incorrect pin for particular number of times, the card must get blocked
3. When user enters correct pin, then user details and welcome message must be displayed on screen
4. After the user enters the correct pin, the ATM must ask the user to input the amount to be withdrawn.
5. If user enters amount greater than available balance than error message must display
6. If user does not enters amount in round off digits, then error message must be displayed
7. If user enters valid amount the exact same amount must be dispatched from ATM
8. Verify how much time is taken in a transaction
9. Verify how much time is taken by system to logout user
10. Verify that user is able to use card of other bank on the ATM
11. Verify that message is displayed when the cash in the ATM is finished
12. Verify that correct message is displayed after the transaction
13. Verify that user is presented with an option to select language of operation
14. Verify that pin is displayed in masked format
15. Verify that error message is displayed when entered amount is greater than account balance
16. Verify that session is timeout if no activity is performed for particular time

### Negative test cases

1. Verify the functionality by entering wrong pin number for a particular number of times.
2. Verify the expired ATM card functionality.
3. Verify the cash withdrawal functionality by entering invalid amount such as 10, 20, 50.
4. Verify the functionality by entering amount greater than per day and per transaction limit

# **Do We Really Write Test Cases For All Testing Types**

**Smoke Testing:** Smoke testing is a type of testing which will be done to verify whether the released build is testable or not.

This is just to test whether the build which was released by dev team is testable. So we just test the basic features like login. To do smoke testing, we pick up the test cases which we have already written. We don’t write test cases particularly to do smoke testing.

**Sanity Testing:** Sanity testing is a type of testing which will be done during the release phase to verify the main functionalities of the application without going deeper. to do this we pick up the test cases which we have already written. same as smoke testing, we don’t write test cases particularly to do sanity testing.

**Functional Testing:** We do test each and every feature of the software application to make sure whether each feature is behaving as specified in the requirement document. To do functional testing, we do write test cases.

**Integration Testing:** [Integration Testing](https://www.softwaretestingmaterial.com/integration-testing/) is the process of testing the interface between two software units. To do Integration testing, we do write test cases.

**System Testing:** Testing the fully integrated application to ensure that the application works as intended or not. To do System testing, we do write test cases.

**Acceptance Testing:** This is done by the end users along with the testers to validate the functionality of the application. Most probably, Acceptance testing will be done by customers, so they may write test cases to do acceptance testing.

**Adhoc Testing:** Ad-hoc testing is quite opposite to the formal testing. It is an informal testing type. In Adhoc testing, testers randomly test the application without following any documents and test design techniques. So, we don’t write any test cases.

**Performance Testing:** Performance is concerned with achieving response times, throughput, and resource-utilization levels that meet the performance objectives of the product. Even though we use tools like LoadRunner, we pickup some scenarios and write test cases to do performance testing. Performance testing test cases are not similar to functional testing test cases.

**Usability Testing:** To verify whether an application is user-friendly or not. Main focus will be on look and feel of the application. We follow a checklist to test the look and feel of the application. So, we don’t write specific test cases to do usability testing.

**Accessibility Testing:** To verify whether a user is able to use it comfortably or not. We follow a checklist to test the look and feel of the application same like usability testing. So, we don’t write specific test cases to do usability testing.

**Reliability Testing:** In terms of reliability testing, we do Perform testing on the application continuously for long period of time in order to verify the stability of the application. We don’t write manual test cases to do reliability testing. We use automation tools to do this by picking up some scenarios.

**Regression Testing:** Repeated testing of an already tested program, after modification, to discover any defects introduced or uncovered as a result of the changes in the software being tested or in another related or unrelated software components.

We use already written test cases to do regression testing. So, we don’t write specific test cases to do regression testing.

**Retesting:** We do retesting to ensure that the defects which were found and posted in the earlier build were fixed or not in the current build. We don’t write test cases to do retesting. We use bug report to do retesting.

**Recovery Testing:** We do recovery testing to determine how quickly the system could recover from crashes. To do recovery testing, we write test cases.

**Globalization Testing:** Globalization is a process of designing a software application so that it can be adapted to various languages and regions without any changes. TO do globalization testing, we write test cases.

# **Software Testing Metrics**

**Software Testing Metrics** are the quantitative measures used to estimate the progress, quality, productivity and health of the software testing process. The goal of software testing metrics is to improve the efficiency and effectiveness in the software testing process and to help make better decisions for further testing process by providing reliable data about the testing process.

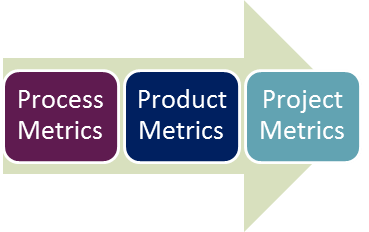
Software testing metrics – Improves the efficiency and effectiveness of a software testing process.

## **Why are Test Metrics Important?**

"We cannot improve what we cannot measure" and Test Metrics helps us to do exactly the same.

* Take decision for next phase of activities
* Evidence of the claim or prediction
* Understand the type of improvement required
* Take decision or process or technology change

## **Types of Test Metrics**

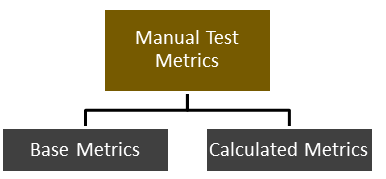
****

* **Process Metrics:** It can be used to improve the process efficiency of the SDLC ( Software Development Life Cycle)
* **Product Metrics:** It deals with the quality of the software product
* **Project Metrics:** It can be used to measure the efficiency of a project team or any testing tools being used by the team members

## **Manual Test Metrics**

In Software Engineering, Manual test metrics are classified into two classes

* **Base Metrics**
* **Calculated Metrics**

****

Base metrics is the raw data collected by Test Analyst during the test case development and execution (**# of test cases executed, # of test cases**). While calculated metrics are derived from the data collected in base metrics. Calculated metrics are usually followed by the test manager for test reporting purposes (**% Complete, % Test Coverage**).

## Base metrics

* Number of test cases written/executed.
* Number of test cases passed/ failed.
* Number of defects found/ accepted/ rejected/ deferred.
* Number of critical/ low/ high/ medium defects.
* Number of planned hours and actual test hours.
* Number of defects found during retesting/ regression testing.

## Calculated metrics

While calculated metrics are derived from the data collected in base metrics. There are two types of calculated metrics.

1. Process metrics
2. Product metrics

**Process metrics** are used during test preparation and test execution. Following are the list of the metrics that falls under this category.

* Test tracking metrics: it shows the percentage of passed, failed, blocked tests etc.

**Example:**

Total tests :240

Passed tests: 181

Failed tests: 28

Blocked tests 31

Formula =

Passed tests = 181/240 x 100 = 75.42%

Failed tests = 28/240 x 100 = 11.67%

Blocked tests = 31/240 x 100 = 12.92%

* Test case preparation productivity: it is the way to estimate time of test case preparation, in other words you can get test case per hour ratio

**Example:**

No. of test cases: 240

Effort spent: 40 hours

Formula =

Test case preparation productivity =240/40 = 6 test cases/ hour

* Test design coverage: it is used to measure the percentage of requirements covered by test cases

**Example:**

No. of test cases mapped to the requirements: 92

No. of requirements: 136

Formula =

Test design coverage = 92/136 \*100 = 68%

* Test execution productivity: it is shows how many test cases that can be executed per hour

**Example:**

No. of test cases: 240

Time spent: 10 hours

Formula =

Test design productivity = 240/10 = 240 test cases/hour

* Test execution coverage: it helps to track the progress of testing by comparing the number of already executed and planned tests

**Example:**

No. of test cases planned to be executed: 250

No. of test cases executed: 185

Time spent: 10 hours

Formula =

Test execution coverage = 185/ 250 \*100 = 74%

* Test effectiveness: it allows us to estimate the value of the test set

**Example:**

No. of defects found in test: 145

No. of defects found after shipping: 11

Formula =

Test effectiveness = 145/ (145 +11) \*100 = 93%

**Process metrics** are used on later stages of the testing cycle during the defect analysis

* Error discovery rate : it shows the effectiveness of test cases in percentage

**Example:**

No. of defects found: 42

No. of test cases executed : 319

Formula =

Error discovery rate = 42/ 319 \*100 = 13%

* Error fix rate : it allows estimating the build quality in terms of defect fixing, in this case calculation is bit more comoplex

**Example:**

No. of defects fixed: 50

No. of defects reopened: 6

No. of new defects due to fix : 12

Formula =

Error discovery rate = (50 - 6)/ (50 +12) \*100 = 71%

* Defect density : the numbers of confirmed defects divided by size of the software, size of the software can be measured in number of line of code or numbers of requirements

**Example:**

No. of defects identified : 42

No. of requirements: 136

Defect density per requirement : 12

Formula =

* Defect leakage : It shows the efficiency of testing activity that took place before user acceptance testing

**Example:**

No. of defects in UAT: 23

No. of defects before UAT: 156

Formula =

# **What is Requirements Traceability Matrix (RTM)**

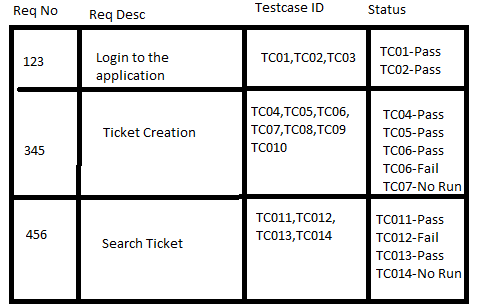
**Requirement Traceability Matrix (RTM)** is a document that maps and traces user requirements with test cases. It captures all requirements proposed by the client and requirement traceability in a single document, delivered at the conclusion of the Software development life cycle. The main purpose of Requirement Traceability Matrix is to validate that all requirements are checked via test cases such that no functionality is unchecked during Software testing.

A simple way is to trace the requirement with its corresponding test scenarios and test cases. This merely is termed as ‘Requirement Traceability Matrix.’

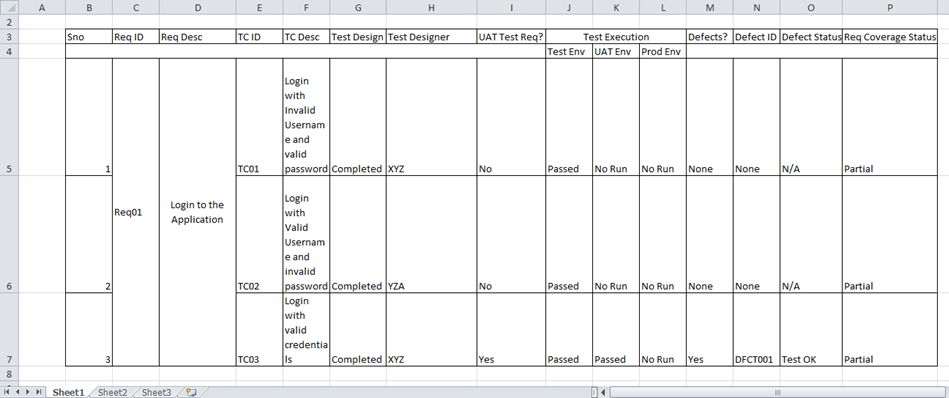
The traceability matrix is typically a worksheet that contains the requirements with all possible test scenarios and cases and their current state, i.e. if they have been passed or failed. This would help the testing team to understand the level of testing activities done for the specific product.

## **Which Parameters to include in the Requirement Traceability Matrix?**

* Requirement ID
* Requirement Type and Description
* Test Cases with Status



But in a typical software testing project, the traceability matrix would have more than these parameters.



As illustrated above, a requirement traceability matrix can:

* Show the requirement coverage in the number of test cases
* Design status as well as execution status for the specific test case
* If there is any User Acceptance test to be done by the users, then UAT status can also be captured in the same matrix.
* The related defects and the current state can also be mentioned in the same matrix.

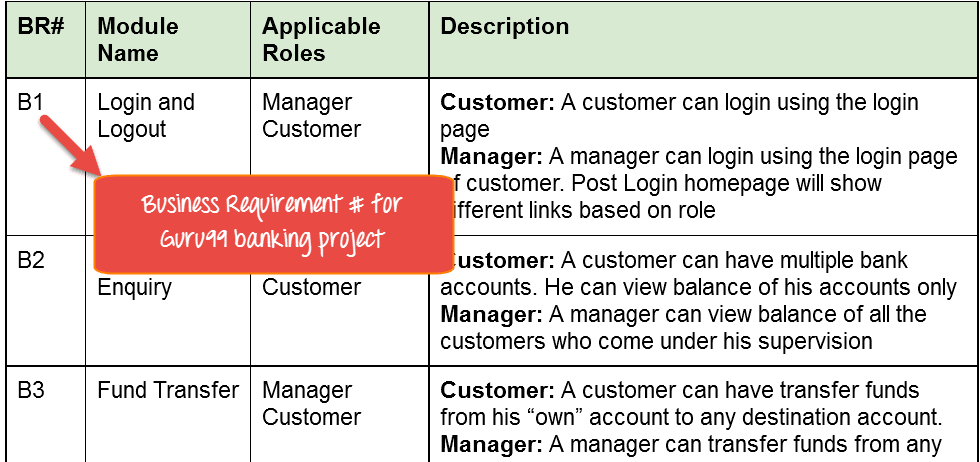
## **How to create Requirement Traceability Matrix**

Let’s understand the concept of Requirement Traceability Matrix through a Guru99 banking project.

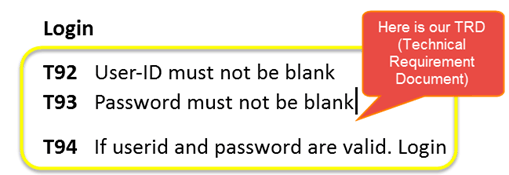
On the basis of **the Business Requirement Document (BRD)** and **Technical Requirement Document (TRD)**, testers start writing test cases.

Let suppose, the following table is our Business Requirement Document or [BRD](https://docs.google.com/document/d/1FsLnZ4thNQF0MhFIFw2Q1KzOTAX89MB8BCvnpsPu5Lc/edit?usp=sharing) for **Guru99 banking project**.

Here the scenario is that the customer should be able to login to Guru99 banking website with the correct password and user#id while manager should be able to login to the website through customer login page.



While the below table is our **Technical Requirement Document (TRD)**.

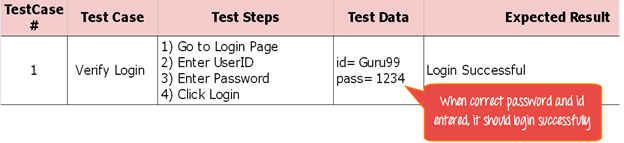


**Note:** QA teams do not document the BRD and TRD. Also, some companies use **Function Requirement Documents (FRD)** which are similar to Technical Requirement Document but the process of creating Traceability Matrix remains the same.

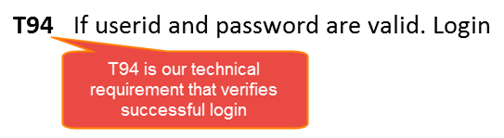
Let’s Go Ahead and create RTM in Testing

**Step 1:** Our [sample Test Case](https://www.guru99.com/test-case.html) is

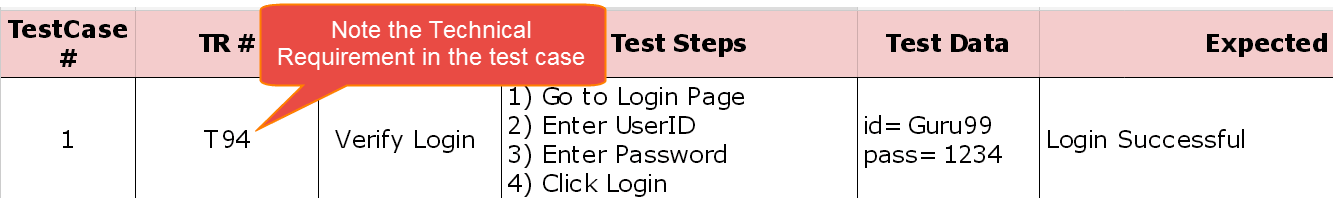
“Verify Login, when correct ID and Password is entered, it should log in successfully”



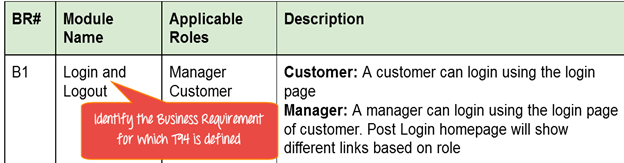
**Step 2**: Identify the Technical Requirement that this test case is verifying. For our test case, the technical requirement is T94 is being verified.



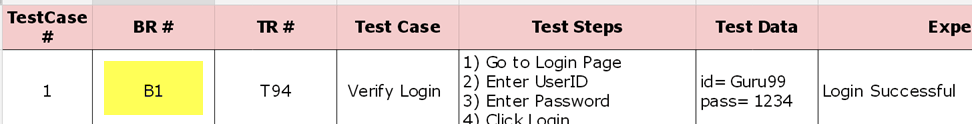
**Step 3:** Note this Technical Requirement (T94) in the Test Case.



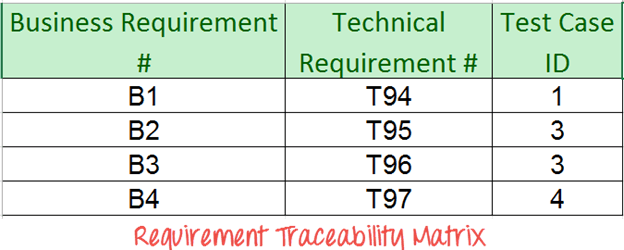
**Step 4:** Identify the Business Requirement for which this TR (Technical Requirement-T94) is defined



**Step 5:** Note the BR (Business Requirement) in Test Case



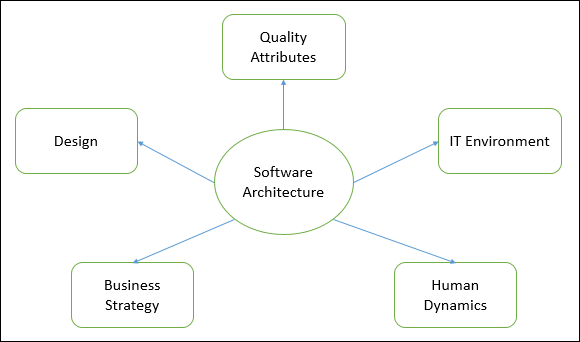
**Step 6:** Do above for all Test Cases. Later Extract the First 3 Columns from your Test Suite. RTM in testing is Ready!



# 

# Software Architecture & Design Introduction

The architecture of a system describes its major components, their relationships (structures), and how they interact with each other. Software architecture and design includes several contributory factors such as Business strategy, quality attributes, human dynamics, design, and IT environment.



We can segregate Software Architecture and Design into two distinct phases: Software Architecture and Software Design. In Architecture, nonfunctional decisions are cast and separated by the functional requirements. In Design, functional requirements are accomplished.

## Software Architecture

Architecture serves as a blueprint for a system. It provides an abstraction to manage the system complexity and establish a communication and coordination mechanism among components.

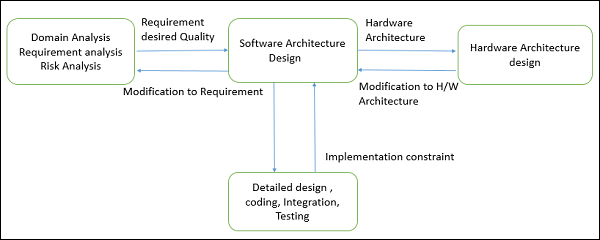
* It defines a structured solution to meet all the technical and operational requirements, while optimizing the common quality attributes like performance and security.
* Further, it involves a set of significant decisions about the organization related to software development and each of these decisions can have a considerable impact on quality, maintainability, performance, and the overall success of the final product. These decisions comprise of −
  + Selection of structural elements and their interfaces by which the system is composed.
  + Behavior as specified in collaborations among those elements.
  + Composition of these structural and behavioral elements into large subsystem.
  + Architectural decisions align with business objectives.
  + Architectural styles guide the organization.

## Software Design

Software design provides a design plan that describes the elements of a system, how they fit, and work together to fulfill the requirement of the system. The objectives of having a design plan are as follows −

* To negotiate system requirements, and to set expectations with customers, marketing, and management personnel.
* Act as a blueprint during the development process.
* Guide the implementation tasks, including detailed design, coding, integration, and testing.

It comes before the detailed design, coding, integration, and testing and after the domain analysis, requirements analysis, and risk analysis.



## Goals of Architecture

The primary goal of the architecture is to identify requirements that affect the structure of the application. A well-laid architecture reduces the business risks associated with building a technical solution and builds a bridge between business and technical requirements.

Some of the other goals are as follows −

* Expose the structure of the system, but hide its implementation details.
* Realize all the use-cases and scenarios.
* Try to address the requirements of various stakeholders.
* Handle both functional and quality requirements.
* Reduce the goal of ownership and improve the organization’s market position.
* Improve quality and functionality offered by the system.
* Improve external confidence in either the organization or system.

### Limitations

Software architecture is still an emerging discipline within software engineering. It has the following limitations −

* Lack of tools and standardized ways to represent architecture.
* Lack of analysis methods to predict whether architecture will result in an implementation that meets the requirements.
* Lack of awareness of the importance of architectural design to software development.
* Lack of understanding of the role of software architect and poor communication among stakeholders.
* Lack of understanding of the design process, design experience and evaluation of design.

## Role of Software Architect

A Software Architect provides a solution that the technical team can create and design for the entire application. A software architect should have expertise in the following areas −

### Design Expertise

* Expert in software design, including diverse methods and approaches such as object-oriented design, event-driven design, etc.
* Lead the development team and coordinate the development efforts for the integrity of the design.
* Should be able to review design proposals and tradeoff among themselves.

### Domain Expertise

* Expert on the system being developed and plan for software evolution.
* Assist in the requirement investigation process, assuring completeness and consistency.
* Coordinate the definition of domain model for the system being developed.

### Technology Expertise

* Expert on available technologies that helps in the implementation of the system.
* Coordinate the selection of programming language, framework, platforms, databases, etc.

### Methodological Expertise

* Expert on software development methodologies that may be adopted during SDLC (Software Development Life Cycle).
* Choose the appropriate approaches for development that helps the entire team.

### Hidden Role of Software Architect

* Facilitates the technical work among team members and reinforces the trust relationship in the team.
* Information specialist who shares knowledge and has vast experience.
* Protect the team members from external forces that would distract them and bring less value to the project.

### Deliverables of the Architect

* A clear, complete, consistent, and achievable set of functional goals
* A functional description of the system, with at least two layers of decomposition
* A concept for the system
* A design in the form of the system, with at least two layers of decomposition
* A notion of the timing, operator attributes, and the implementation and operation plans
* A document or process which ensures functional decomposition is followed, and the form of interfaces is controlled

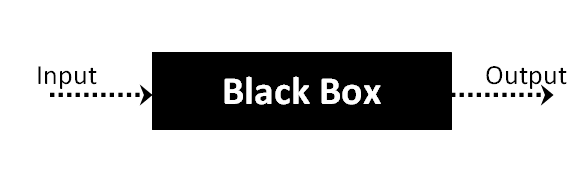
## Quality Attributes

Quality is a measure of excellence or the state of being free from deficiencies or defects. Quality attributes are the system properties that are separate from the functionality of the system.

Implementing quality attributes makes it easier to differentiate a good system from a bad one. Attributes are overall factors that affect runtime behavior, system design, and user experience.

# **What is BLACK Box Testing Techniques**

**Black Box Testing** is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.



## **Types of Black Box Testing**

There are many types of Black Box Testing but the following are the prominent ones –

#### #1) Functional Testing

This testing type deals with the functional requirements or specifications of an application. Here, different actions or functions of the system are being tested by providing the input and comparing the actual output with the expected output.

**For example**, when we test a Dropdown list, we click on it and verify if it expands and all the expected values are showing in the list.

**Few major types of Functional Testing are:**

* Smoke Testing
* Sanity Testing
* Integration Testing
* System Testing
* Regression Testing
* User Acceptance Testing

#### #2) Non-Functional Testing

**Apart from the functionalities of the requirements, there are even several non-functional aspects that are required to be tested to improve the quality and performance of the application.**

**Few major types of Non-Functional Testing include:**

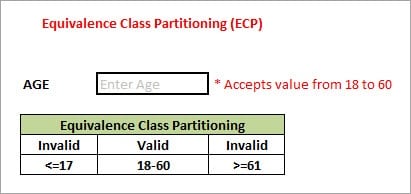
* **Usability Testing**
* **Load Testing**
* **Performance Testing**
* **Compatibility Testing**
* **Stress Testing**
* **Scalability Testing**

# Equivalence Partitioning Technique

This technique is also known as Equivalence Class Partitioning (ECP). In this technique, input values to the system or application are divided into different classes or groups based on its similarity in the outcome.

Hence, instead of using each and every input value, we can now use any one value from the group/class to test the outcome. This way, we can maintain test coverage while we can reduce the amount of rework and most importantly the time spent.

**For Example:**

****

As present in the above image, the “AGE” text field accepts only numbers from 18 to 60. There will be three sets of classes or groups.

**Two invalid classes will be:**

a) Less than or equal to 17.

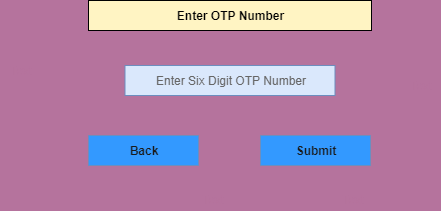
b) Greater than or equal to 61.

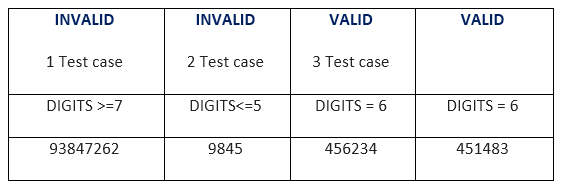
A valid class will be anything between 18 and 60.

We have thus reduced the test cases to only 3 test cases based on the formed classes thereby covering all the possibilities. So, testing with any one value from each set of the class is sufficient to test the above scenario.

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. 1. OTP Number = 6 digits



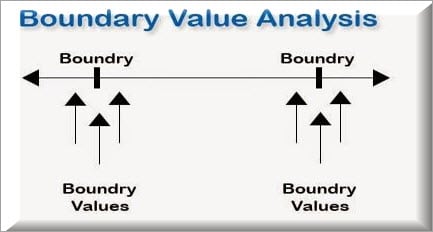


# Boundary Value Analysis

The name itself defines that in this technique, we focus on the values at boundaries as it is found that many applications have a high amount of issues on the boundaries.

Boundary refers to values near the limit where the behavior of the system changes. In boundary value analysis, both valid and invalid inputs are being tested to verify the issues.

**For Example:**

****

If we want to test a field where values from 1 to 100 should be accepted, then we choose the boundary values: 1-1, 1, 1+1, 100-1, 100, and 100+1. Instead of using all the values from 1 to 100, we just use 0, 1, 2, 99, 100, and 101.

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.

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.

Boundary Value Analysis

Testing of boundary values is done by making valid and invalid partitions. Invalid partitions are tested because testing of output in adverse conditions is also essential.

# Decision table technique in Black box testing

A **Decision Table** is a tabular representation of inputs versus rules/cases/test conditions. It is a very effective tool used for both complex software testing and requirements management. Decision table helps to check all possible combinations of conditions for testing and testers can also identify missed conditions easily. The conditions are indicated as True(T) and False(F) values.

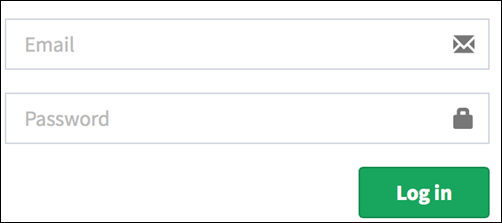
## **What is Decision Table Testing?**

Decision table testing is a software testing technique used to test system behavior for different input combinations. This is a systematic approach where the different input combinations and their corresponding system behavior (Output) are captured in a tabular form. That is why it is also called a **Cause-Effect** table where Cause and effects are captured for better test coverage.

Let’s learn with an example.

## **Example 1: How to make Decision Base Table for Login Screen**

Let’s create a decision table for a login screen.



The condition is simple if the user provides correct username and password the user will be redirected to the homepage. If any of the input is wrong, an error message will be displayed.

| **Conditions** | **Rule 1** | **Rule 2** | **Rule 3** | **Rule 4** |
| --- | --- | --- | --- | --- |
| **Username (T/F)** | F | T | F | T |
| **Password (T/F)** | F | F | T | T |
| **Output (E/H)** | E | E | E | H |

Legend:

* T – Correct username/password
* F – Wrong username/password
* E – Error message is displayed
* H – Home screen is displayed

Interpretation:

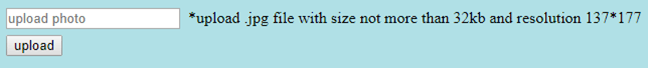
* Case 1 – Username and password both were wrong. The user is shown an error message.
* Case 2 – Username was correct, but the password was wrong. The user is shown an error message.
* Case 3 – Username was wrong, but the password was correct. The user is shown an error message.
* Case 4 – Username and password both were correct, and the user navigated to homepage

## **Example 2: How to make Decision Table for Upload Screen**

Now consider a dialogue box which will ask the user to upload photo with certain conditions like –

1. You can upload only ‘.jpg’ format image
2. file size less than 32kb
3. resolution 137\*177.

If any of the conditions fails the system will throw corresponding error message stating the issue and if all conditions are met photo will be updated successfully



Let’s create the decision table for this case.

| **Conditions** | **Case 1** | **Case 2** | **Case 3** | **Case 4** | **Case 5** | **Case 6** | **Case 7** | **Case 8** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Format** | .jpg | .jpg | .jpg | .jpg | Not .jpg | Not .jpg | Not .jpg | Not .jpg |
| **Size** | Less than 32kb | Less than 32kb | >= 32kb | >= 32kb | Less than 32kb | Less than 32kb | >= 32kb | >= 32kb |
| **resolution** | 137\*177 | Not 137\*177 | 137\*177 | Not 137\*177 | 137\*177 | Not 137\*177 | 137\*177 | Not 137\*177 |
| **Output** | Photo uploaded | Error message resolution mismatch | Error message size mismatch | Error message size and resolution mismatch | Error message for format mismatch | Error message format and resolution mismatch | Error message for format and size mismatch | Error message for format, size, and resolution mismatch |

## **Advantages of Decision Table Testing**

* When the system behavior is different for different inputs and not same for a range of inputs, both equivalence partitioning, and boundary value analysis won’t help, but a decision table can be used.
* The representation is simple so that it can be easily interpreted and is used for development and business as well.
* This table will help to make effective combinations and can ensure a better coverage for testing
* Any complex business conditions can be easily turned into decision tables
* In a case we are going for 100% coverage typically when the input combinations are low, this technique can ensure the coverage.

# **What is State Transition Testing?**

State Transition testing, a black box testing technique, in which outputs are triggered by changes to the input conditions or changes to 'state' of the system. In other words, tests are designed to execute valid and invalid state transitions.

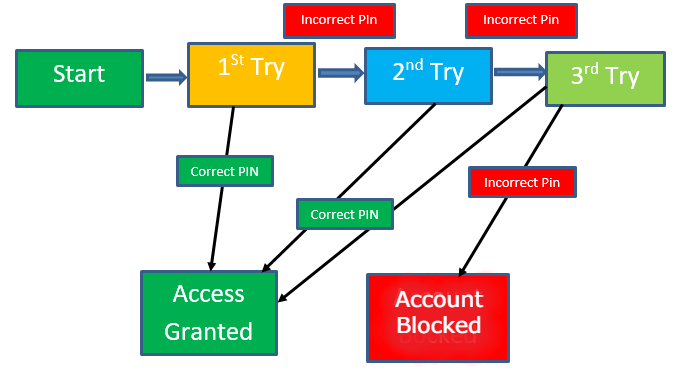
State transition technique is a dynamic testing technique, which is used when the system is defined in terms of a finite number of states and the transitions between the states are governed by the rules of the system.

### **Example 1:**

Let’s consider an ATM system function where if the user enters the invalid password three times the account will be locked.

In this system, if the user enters a valid password in any of the first three attempts the user will be logged in successfully. If the user enters the invalid password in the first or second try, the user will be asked to re-enter the password. And finally, if the user enters incorrect password 3rd time, the account will be blocked.

### **State transition diagram**

****

In the diagram whenever the user enters the correct PIN he is moved to Access granted state, and if he enters the wrong password he is moved to next try and if he does the same for the 3rd time the account blocked state is reached.

### **State Transition Table**

|  | **Correct PIN** | **Incorrect PIN** |
| --- | --- | --- |
| **S1) Start** | **S5** | **S2** |
| **S2) 1st attempt** | **S5** | **S3** |
| **S3) 2nd attempt** | **S5** | **S4** |
| **S4) 3rd attempt** | **S5** | **S6** |
| **S5) Access Granted** | **–** | **–** |
| **S6) Account blocked** | **–** | **–** |

| **Advantages** | **Disadvantages** |
| --- | --- |
| This testing technique will provide a pictorial or tabular representation of system behavior which will make the tester to cover and understand the system behavior effectively. | The main disadvantage of this testing technique is that we can’t rely in this technique every time. For example, if the system is not a finite system (not in sequential order), this technique cannot be used. |
| By using this testing, technique tester can verify that all the conditions are covered, and the results are captured | Another disadvantage is that you have to define all the possible states of a system. While this is all right for small systems, it soon breaks down into larger systems as there is an exponential progression in the number of states. |

# White Box Testing

It tests internal coding and infrastructure of a software focused on checking predefined inputs against expected and desired outputs. It is based on the 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.

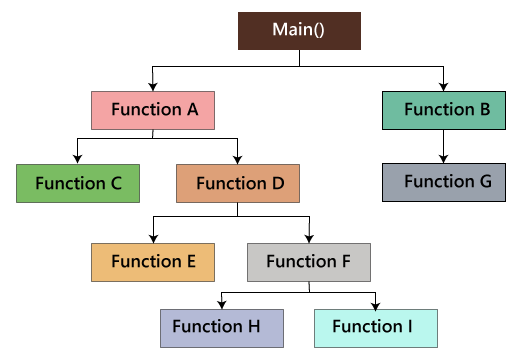
Developers do white box testing. In this, the developer will test every line of the code of the program. The developers perform the White-box testing and then send the application or the software to the testing team, where they will perform the [black box testing](https://www.javatpoint.com/black-box-testing) and verify the application along with the requirements and identify the bugs and send it to the developer.

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:



### 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.

1. {
2. while(50,000)
3. ……
4. ……
5. }

### 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:**

1. if(condition) - true
2. {
3. …..
4. ……
5. ……
6. }
7. else - false
8. {
9. …..
10. ……
11. ……
12. }

### 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.

## 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.

# 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 then it will affect the overall plan.

When we detect 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.

## Types of mutation testing

Mutation testing can be classified into three parts, which are as follows:

* Decision mutations
* value mutations
* Statement mutations

### 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 operators to detect the errors in the program.

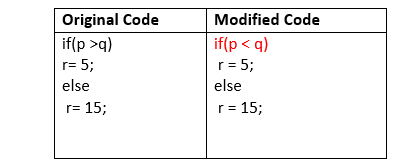
Like if we do the following changes in arithmetic operators:

* 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:

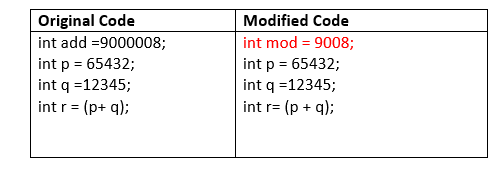


### Value mutations

In this, the values will modify to identify the errors in the program, and generally, we will change the following:

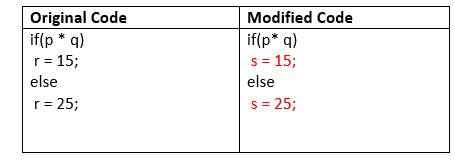
* Small value à higher value
* Higher value àSmall value.

**For Example:**

****

### Statement Mutations

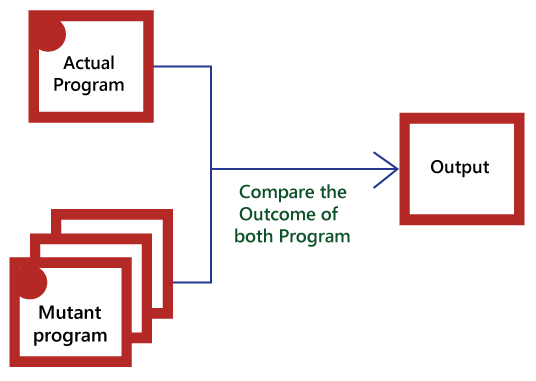
**Statement mutations means that we can do the modifications into the statements by removing or replacing the line as we see in the below example:**

****

**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:**

****

### 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.

# Severity and Priority in 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 the test engineer can continue the testing.

**Minor:** if the severity of a bug is minor, which means that all the U.I is not working fine, but testing can be processed without interruption.

## Priority

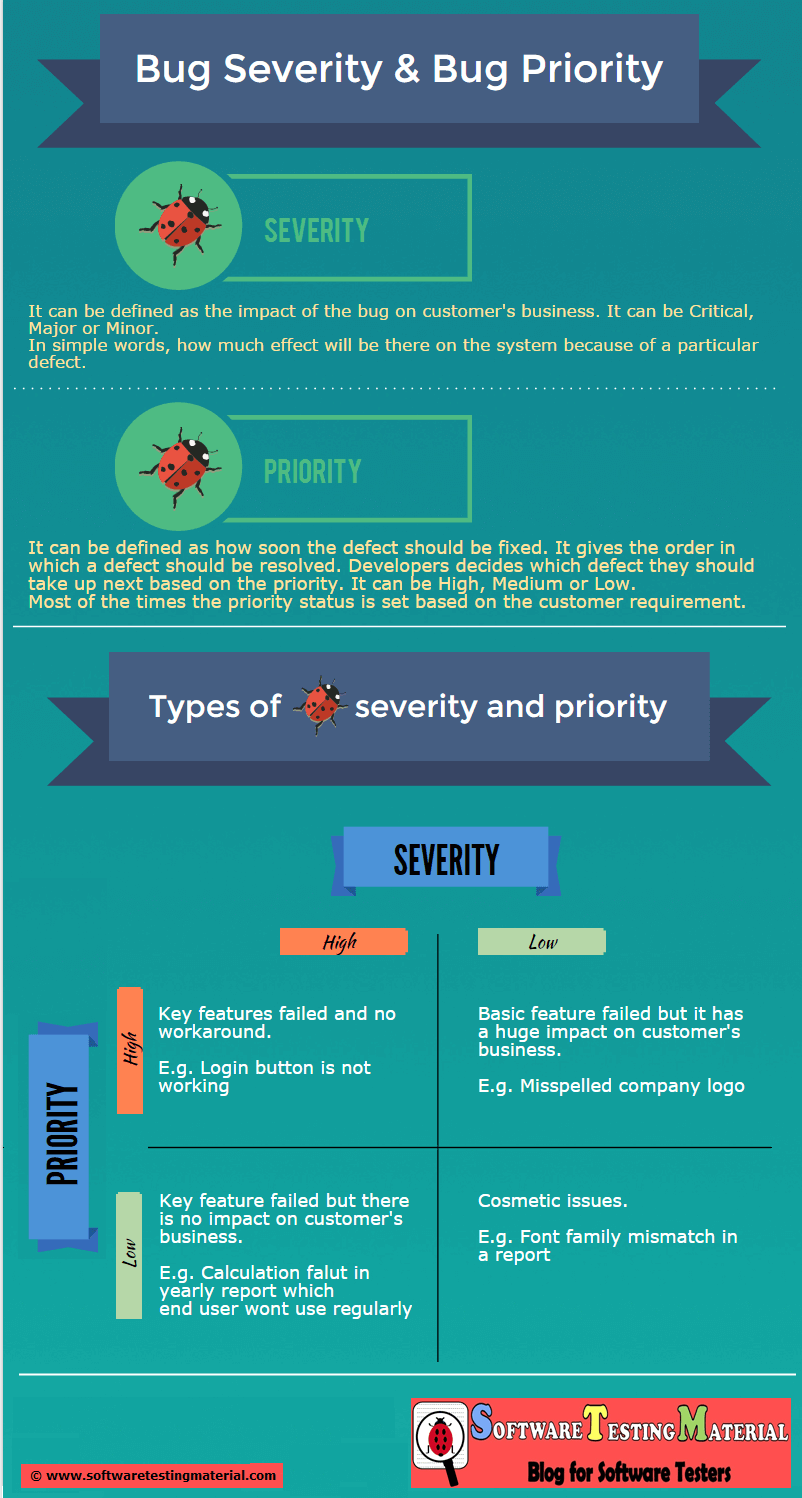
Priority is important for fixing the bug or which bug to be fixed first or how soon the bug should be fixed.

It can be **urgent, high, medium, and low**.

**High:** it has 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 flaw should be fixed if there is time, but it can be deferred with the next release.



# **Defect Triage**

*Defect triage is a process to prioritize the defects based on severity, risk, frequency of occurrence.*

## **Why do we need to have ‘Defect Triage’?**

The goal of Bug Triage is to evaluate, prioritize and assign the resolution of defects. The team needs to validate the severity of the defect, make changes as per need, finalize resolution of the defects, and assign resources. Mainly used in agile project management.

The defect triage process involves holding a session with a triage team, which includes stakeholders like Product Manager, Testing Manager/Lead, Development Manager/Lead, and Business Analysts. The goal of this team is to evaluate the defects, assess them, and attach priorities and severity level.



Priorities correspond to business perspective and severity corresponds to technicalities.

Many times, few defects may be considered trivial and rejected at this stage. Accepted defects are prioritized and assigned for resolution.

Factors to be considered while evaluating and prioritizing the defects are:

* The validity of the defect
* Time sensitivity for resolution
* The complexity involved in the resolution
* Business impact

This process is not just about attaching severity and priority to the defects. It also provides all relevant information required to track, replicate, and fix them.

## **Roles and Responsibilities of participants during ‘Defect Triage.’**

### **Test Team Leader**

* Scheduled bug triage meeting and send meeting notification for attendees.
* Create a defect report and send it to all attendees before the meeting.
* Assign priority and severity of the defects.
* Give a presentation so that other members understand the Root Cause of the defect.
* Every meeting note is captured and sent to meeting attendees.

### **Development Lead**

* Helps in the prioritization of the defects.
* Discuss defect difficulty and explain the risk involved because of that defect.
* Allocate work to fix defects to relevant developers.
* Update the defect resolution and include development notes in case any information is missing or any additional information needed by developers.

### **Project Manager**

* Help in the prioritization of the defects.
* Discuss the next iteration release date for QA.
* Need to make sure that related user representatives are also invited to the bug triage meeting.

## **What happens during ‘Defect Triage’ Meeting?**

* Test Team leader sends out a bug report with the new defects. During the defect triage meeting, each defect is analyzed to see whether right priority and severity are assigned to it.
* Priorities are rearranged if needed.
* Defects are analyzed and evaluated by the degree of their severity.
* This includes discussion regarding the complexity of the defect, risks, rejection, reassignment of errors is done.
* Updates are captured in the bug tracking system.
* The QA engineer will make the changes to each defect and discuss them with each attendee.
* The “Comments” field is updated correctly by noting essential points of the meeting.

## **What is the outcome of the ‘Defect Triage’?**

At the end of every meeting, Defect Triage Metrics will be prepared and given to all the attendees. This report acts as the meeting minutes which will prove helpful for future meetings.

# Performance Testing

Performance testing is a testing measure that evaluates the speed, responsiveness and stability of a computer, network, software program or device under a workload. Organizations will run performance tests in order to identify performance-related [bottlenecks](https://www.techtarget.com/searchnetworking/definition/bottleneck).

### **Performance testing metrics**

A number of performance metrics, or key performance indicators ([KPIs](https://searchbusinessanalytics.techtarget.com/definition/key-performance-indicators-KPIs)), can help an organization evaluate current performance.

Performance metrics commonly include:

* **Throughput.** How many units of information a system processes over a specified time
* [**Memory**](https://searchstorage.techtarget.com/definition/memory-card)**.** The working storage space available to a processor or workload
* **response time, or latency.** The amount of time that elapses between a user-entered request and the start of a system's response to that request
* **Bandwidth.** The volume of data per second that can move between workloads, usually across a network
* **CPU interrupts per second.** The number of hardware interrupts a process receives per second

These metrics and others help an organization perform multiple types of performance tests.

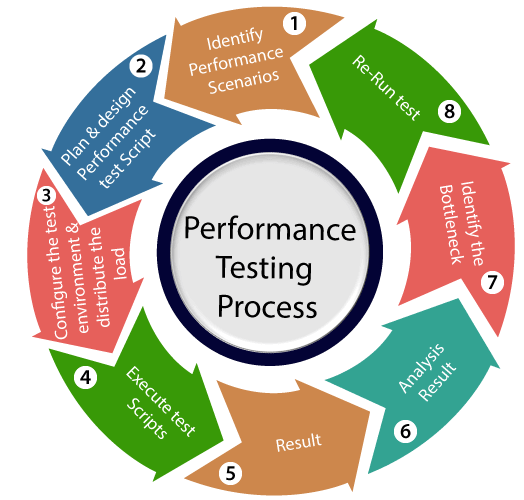
## Performance testing process

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



### Identify performance scenarios

Firstly, we will identify the performance scenarios based on these below factors:

**Most common scenarios:** It means that we can find the performance scenarios based on the scenarios, which are 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.

### 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.

### 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 bottlenecking 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

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.

# Security Testing

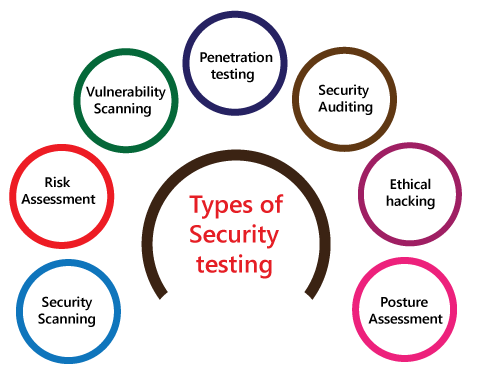
Security testing is an integral part of software testing, which is used to discover the weaknesses, risks, or threats in the software application and also help us to stop the nasty attack from the outsiders and make sure the security of our software applications.

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.

## 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](https://www.javatpoint.com/automation-testing) and [manual testing](https://www.javatpoint.com/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 a 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 generate 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](https://www.javatpoint.com/cyber-security-tutorial) professional tries to identify and exploit the weakness in the computer system. The primary objective of this testing is to simulate outbreaks and also find 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](https://www.javatpoint.com/control-system-tutorial) for the security faults.

### Ethical hacking

[Ethical hacking](https://www.javatpoint.com/ethical-hacking-tutorial) 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.

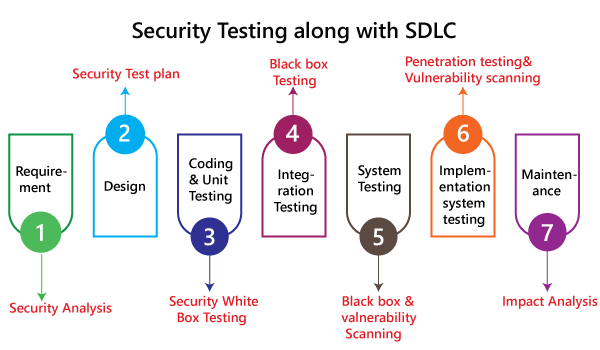
### Posture Assessment

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](https://www.javatpoint.com/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).



**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](https://www.javatpoint.com/functional-testing), [integration testing](https://www.javatpoint.com/integration-testing), [system testing](https://www.javatpoint.com/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 applications are at risk. Here we are going to discuss some common weaknesses of the web application.

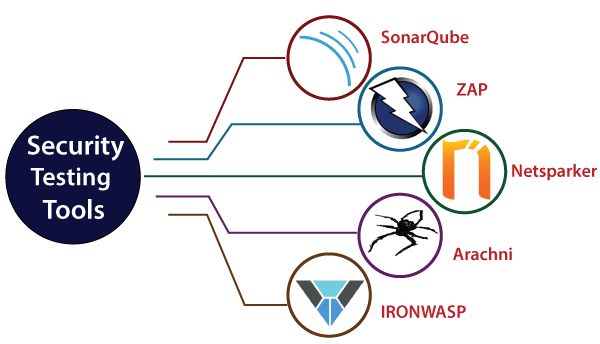
* Client-side attacks
* Authentication
* Authorization
* Command execution
* Logical attacks
* Information disclosure

## **Example Test Scenarios for Security Testing:**

Sample Test scenarios to give you a glimpse of security test cases –

* A password should be in encrypted format
* Application or System should not allow invalid users
* Check cookies and session time for application
* For financial sites, the Browser back button should not work.

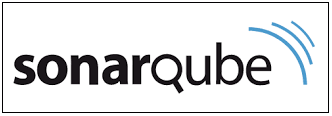
# Security testing tools



## 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#](https://www.javatpoint.com/c-sharp-tutorial), [JavaScript](https://www.javatpoint.com/javascript-tutorial), [PHP](https://www.javatpoint.com/php-tutorial), Ruby, [Cobol](https://www.javatpoint.com/cobol), [C](https://www.javatpoint.com/c-programming-language-tutorial)/[C++](https://www.javatpoint.com/cpp-tutorial) and so on of the web applications. SonarQube tool is written on the [JAVA programming language](https://www.javatpoint.com/java-tutorial).

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](https://www.javatpoint.com/apache-ant-tutorial), [Maven](https://www.javatpoint.com/maven-tutorial), [Gradle](https://www.javatpoint.com/gradle), [Jenkins](https://www.javatpoint.com/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.

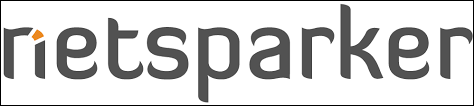


### Features of ZAP

* 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.

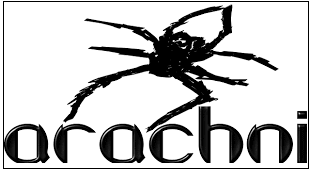


### Features of Netsparker

* 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 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 platforms 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.

## 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](https://www.javatpoint.com/python-tutorial) and [Ruby programming languages](https://www.javatpoint.com/ruby-tutorial).



### 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.

# Volume Testing

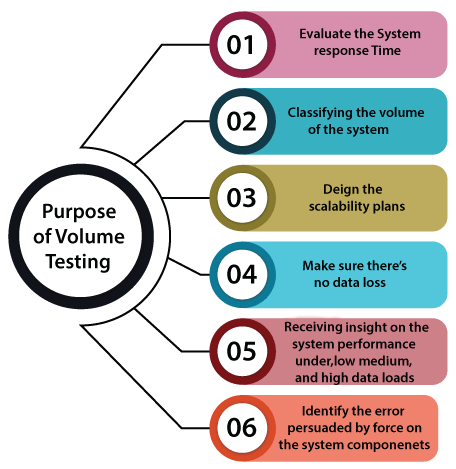
Volume testing comes under [software testing](https://www.javatpoint.com/software-testing-tutorial). It helps us to check the behavior of an application by inserting a massive volume of the load in terms of data known as **volume testing**.

In volume testing, we will concentrate on the number of data rates rather 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 enhanced in the database. In this testing, a massive volume of information is acted upon by the software.

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:



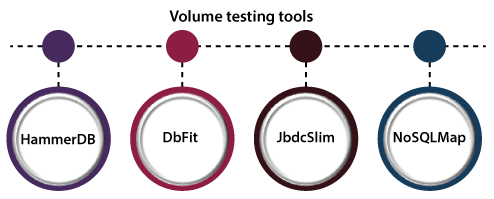
## Volume testing tools

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.

Volume Testing

**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.

# **What is Cloud Testing**

**Cloud Testing** is a type of software testing in which the software application is tested using cloud computing services. The purpose of Cloud testing is to test the software for functional as well as non-functional requirements using cloud computing which ensures faster availability with scalability and flexibility to save time and cost for software testing.

Cloud computing is an internet-based platform that renders various computing services like hardware, software and other computer related services remotely.

There are mainly three models of [Cloud Computing](https://www.guru99.com/cloud-computing-for-beginners.html) :

* **SaaS**– Software as a service
* **PaaS**– Platform as a service
* **IaaS**– Infrastructure as a service

## **Type of Testing in Cloud**

The whole cloud testing is segmented into four main categories

* **Testing of the whole cloud**: The cloud is viewed as a whole entity and based on its features testing is carried out. Cloud and SaaS vendors, as well as end users, are interested in carrying out this type of testing
* **Testing within a cloud**: By checking each of its internal features, testing is carried out. Only cloud vendors can perform this type of testing
* **Testing across cloud**: Testing is carried out on different types of cloud-like private, public and hybrid clouds
* **SaaS testing in cloud**: Functional and non-functional testing is carried out on the basis of application requirements

| Types of Cloud Testing | Task Performed |
| --- | --- |
| * SaaS or Cloud-oriented Testing: | This type of testing is usually performed by cloud or SaaS vendors. The primary objective is to assure the quality of the provided service functions offered in a cloud or a SaaS program. Testing performed in this environment is integration, functional, security, unit, system function validation and [Regression Testing](https://www.guru99.com/regression-testing.html) as well as performance and scalability evaluation. |
| * Online based application testing on a cloud: | Online application vendors perform this testing that checks performance and [Functional Testing](https://www.guru99.com/functional-testing.html) of the cloud-based services. When applications are connected with legacy systems, the quality of the connectivity between the legacy system and under test application on a cloud is validated. |
| * Cloud-based application testing over clouds: | To check the quality of a cloud-based application across different clouds this type of testing is performed. |

# Usability Testing

It is primarily used in user-centered interaction design in 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, whilst 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.

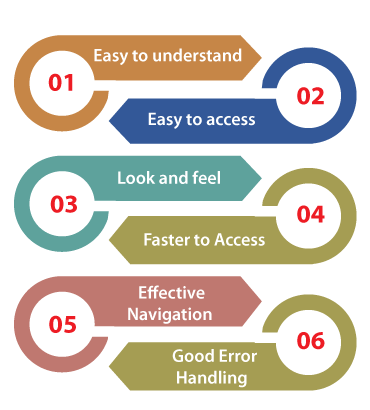
In other words, we can say that Usability testing is one of the distinct testing techniques that identify the defect in the end-user communication of software product. And that's why it is also known as **User Experience (UX) Testing**.

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.

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**

**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 errors 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.

We need usability testing because usability testing is to build a system with great user experience. Usability is not only used for [software development](https://www.javatpoint.com/software-development-life-cycle) 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

## Various Strategies of Usability Testing/Usability Testing Methods

* **A/B Testing**
* **Hallway Testing**
* **Laboratory Usability Testing**
* **Expert Review**
* **Automated Expert Review**
* **Synchronous Remote Usability Testing**
* **Asynchronous Remote Usability Testing**

### 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.

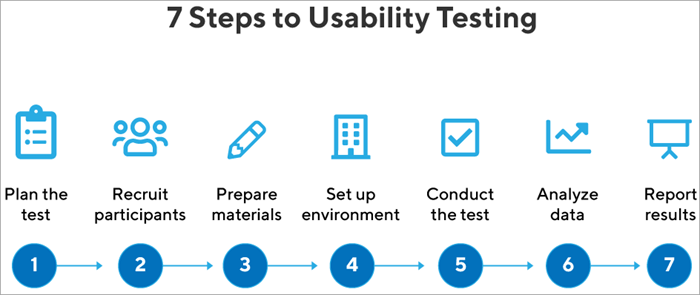
From time to time, remote testing is implemented using video conferencing. And this approach is less expensive in comparison with other types of usability testing approaches.

## Usability Testing Process

## 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.

Therefore, the process of usability testing completed into the following steps, as we can see in the following image:



#### #1) Planning

This stage is the first stage in software testing. The planning stage is when the objective of the project is defined. The documentation of the project is done in the planning stage. This is one of the most fundamental and critical stages in software testing.

In this stage, we plan the road map to proceed with the testing process. We plan for questions like what to test? how to test? who would be testing which features? which different scenarios should be considered for testing?. In the planning stage, we gather the business requirement and prepare the plan.

#### #2) Recruiting

In this phase, we recruit the facilitator and the participants according to the budget and complexity of the project.

#### #3) Execution

In this stage, the participants perform usability testing and perform the assigned task.

#### #4) Data Analysis

During the data analysis stage, the feedback received from usability testing is analyzed. Results are categorized and patterns are identified. The conclusion is generated from the feedback submitted to improve the product and thereby improve the business.

#### #5) Reporting

The feedback and findings received are shared with the customers and the development teams across the project. Discussion on the areas of improvement and its implementation is considered in this stage.

# 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).

Accessibility testing is another type of [software testing](https://www.javatpoint.com/software-testing-tutorial) 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.

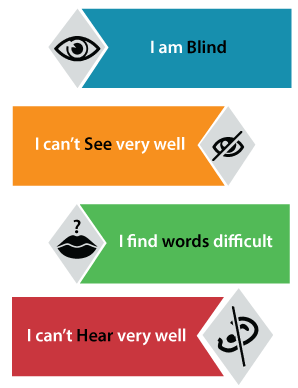
For accessibility testing, we have some assured rules and regulations, which need to be followed as well.

**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.



## 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.

## 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.

### Automation method

Generally, the Automation technique is used for various testing methods. The [automation testing](https://www.javatpoint.com/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:

<https://wave.webaim.org/>

### 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](https://www.javatpoint.com/ibm-full-form) 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 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.

# Cross-browser testing

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.

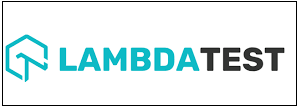
We have various cross-browser testing tools available in the market.

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

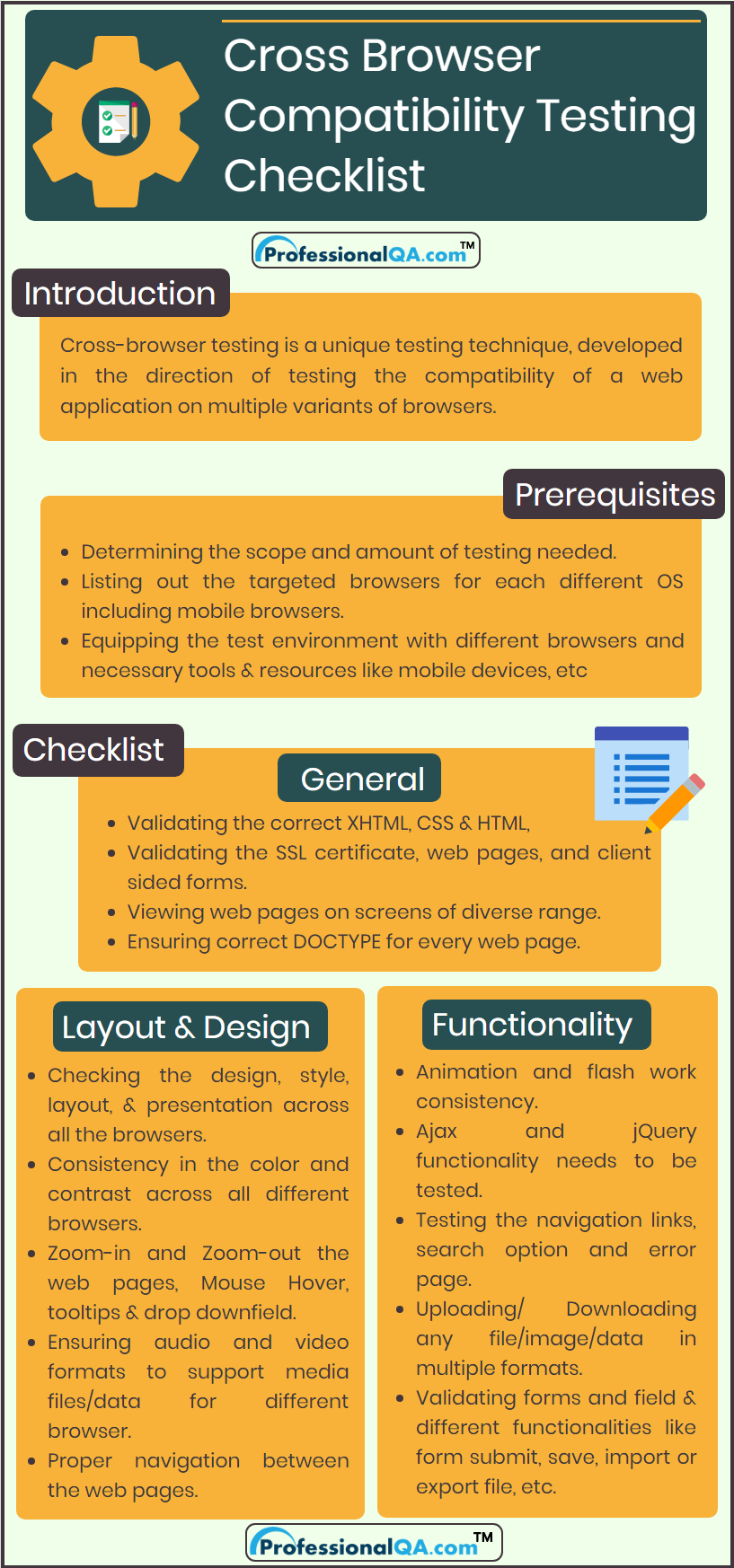
It is an online tool, which is used to test and report the cross-browser design modification and scripting issues on our website. It will compare the output of every browser repeatedly and check the conflicts in our pages, so that we can fix them quickly. It can test the various pages of our websites immediately. It will also create a report once the test has been completed.



### Feature of Browser 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.

# **Cross-Browser Testing Checklist**



### 1. General:

* Validating the correct CSS & HTML.
* Validating the correct XHTML.
* Validating the SSL certificate for all the specified & targeted browsers.
* Validating the web pages and client sided forms and fields with and without the Javascript.
* Viewing web pages on screens of diverse range.
* Ensuring correct DOCTYPE for each and every web page.

### 2. Layout and Design:

* Checking the design, style, layout and presentation consistency across all the browsers.
* To check that if images are in their standard resolutions and proposed alignment on each browser or not.
* Font with its attributes such as colour, size formatting etc., which were actually implemented and used, is working identically for all the browsers.
* Checkboxes, radio buttons, forms are aligned correctly.
* Consistency in the colour and contrast across all different browsers.
* To check out the header and footer layout of the web pages.
* Ensuring website's responsive quality across each targeted browser.
* Checking the alignment of data content on the web pages.
* Proper spacing between sections, fields, paragraphs, images and data contents.
* Ensuring a good number of audio and video formats to support media files/data for each different browser.
* Horizontal and Vertical scroll bar appearance.
* Special characters requiring html character encoding.
* Zoom-in and Zoom-out the web pages.
* Mouse Hover and tool tips.
* Drop-down field.
* Proper navigation between the web pages.

### 3. Functionality:

* Whether plug-ins required for the web application is supported by all the browsers or not.
* Uploading or Downloading any file or image or data in multiple formats.
* Ajax and jQuery functionality needs to be tested.
* Testing the navigation links, search option and error page.
* Validating the forms and fields for each different browser.
* Validating the different functionalities like form submit, saving the data, importing or exporting file, etc.
* Animation and flash work consistency.
* Interaction with web pages through mouse, keyboard, etc. in a similar way for all the browsers.
* Validating the working of the scripts used in the website across all targeted browser.
* Performance of the web application on different browsers for multiple variants of load.

## What is Web Application Testing?

Web application testing, a software testing technique exclusively adopted to test the applications that are hosted on web in which the application interfaces and other functionalities are tested.

## Web Application Testing - Techniques:

1. Functionality Testing - The below are some of the checks that are performed but not limited to the below list:

* Verify there is no dead page or invalid redirects.
* First check all the validations on each field.
* Wrong inputs to perform negative testing.
* Verify the workflow of the system.
* Verify the data integrity.

2. Usability testing - To verify how the application is easy to use with.

* Test the navigation and controls.
* Content checking.
* Check for user intuition.

3. Interface testing - Performed to verify the interface and the dataflow from one system to other.

4. Compatibility testing- Compatibility testing is performed based on the context of the application.

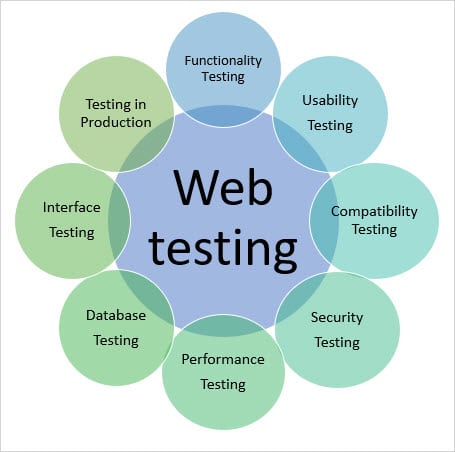
* Browser compatibility
* Operating system compatibility
* Compatible to various devices like notebook, mobile, etc.

5. Performance testing - Performed to verify the server response time and throughput under various load conditions.

* Load testing - It is the simplest form of testing conducted to understand the behaviour of the system under a specific load. Load testing will result in measuring important business critical transactions and load on the database, application server, etc. are also monitored.
* Stress testing - It is performed to find the upper limit capacity of the system and also to determine how the system performs if the current load goes well above the expected maximum.
* Soak testing - Soak Testing also known as endurance testing, is performed to determine the system parameters under continuous expected load. During soak tests the parameters such as memory utilization is monitored to detect memory leaks or other performance issues. The main aim is to discover the system's performance under sustained use.
* Spike testing - Spike testing is performed by increasing the number of users suddenly by a very large amount and measuring the performance of the system. The main aim is to determine whether the system will be able to sustain the work load.

6. Security testing - Performed to verify if the application is secured on web as data theft and unauthorized access are more common issues and below are some of the techniques to verify the security level of the system.

* Injection
* Broken Authentication and Session Management
* Cross-Site Scripting (XSS)
* Insecure Direct Object References
* Security Misconfiguration
* Sensitive Data Exposure
* Missing Function Level Access Control
* Cross-Site Request Forgery (CSRF)
* Using Components with Known Vulnerabilities
* Unvalidated Redirects and Forwards



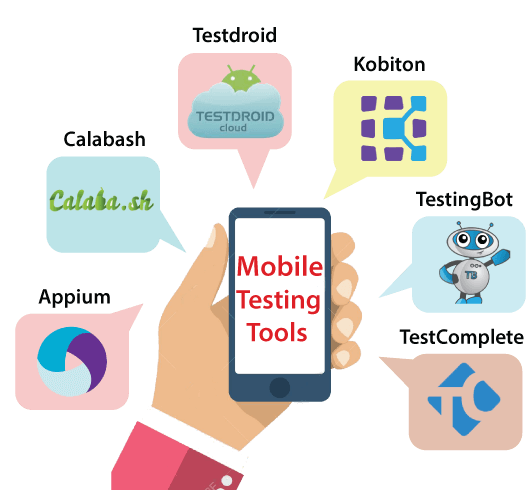
# 

# 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 was established by 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](https://www.javatpoint.com/mac-full-form), 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](https://www.javatpoint.com/android-tutorial) and iOS applications. For the mobile application, we can write and run automated acceptance tests. It is developed and maintained by [Xamarin](https://www.javatpoint.com/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](https://www.javatpoint.com/manual-testing) and [API](https://www.javatpoint.com/api-full-form) 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](https://www.javatpoint.com/creating-first-ios-application).



### 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 was established by SmartBear Company. It ensures the delivery of high-quality software that enhances test coverage. It supports multiple platforms such as Windows, [iOS](https://www.javatpoint.com/ios-full-form), 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](https://www.javatpoint.com/gps-full-form), Gyroscopes, and Accelerometers.

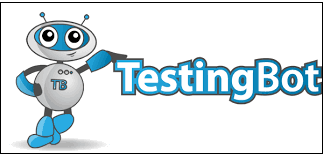


### Features of TestComplete tool

* 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.



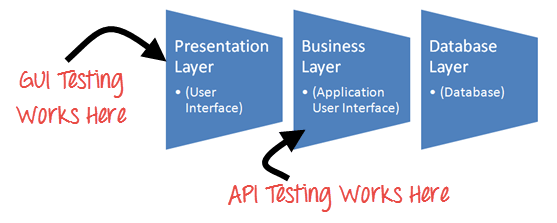
### Features of TestingBot tool

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.

# **API Testing**

**API TESTING** is a software testing type that validates Application Programming Interfaces (APIs). The purpose of API Testing is to check the functionality, reliability, performance, and security of the programming interfaces. In API Testing, instead of using standard user inputs(keyboard) and outputs, you use software to send calls to the API, get output, and note down the system’s response. API tests are very different from GUI Tests and won’t concentrate on the look and feel of an application. It mainly concentrates on the business logic layer of the software architecture.



API automation Testing requires an application that can be interacted via an API. In order to test an API, you will need to

* Use Testing Tool to drive the API
* Write your own code to test the API

## **How to Test API**

API automation testing should cover at least following testing methods apart from usual SDLC process

* **Discovery testing:** The test group should manually execute the set of calls documented in the API like verifying that a specific resource exposed by the API can be listed, created and deleted as appropriate
* **Usability testing:** This testing verifies whether the API is functional and user-friendly. And does API integrates well with another platform as well
* **Security testing:** This testing includes what type of authentication is required and whether sensitive data is encrypted over HTTP or both
* **Automated testing:** API testing should culminate in the creation of a set of scripts or a tool that can be used to execute the API regularly
* **Documentation:** The test team has to make sure that the documentation is adequate and provides enough information to interact with the API. Documentation should be a part of the final deliverable

## **Best Practices of API Testing:**

* API Test cases should be grouped by test category
* On top of each test, you should include the declarations of the APIs being called.
* Parameters selection should be explicitly mentioned in the test case itself
* Prioritize API function calls so that it will be easy for testers to test
* Each test case should be as self-contained and independent from dependencies as possible
* Avoid “test chaining” in your development
* Special care must be taken while handling one-time call functions like – Delete, CloseWindow, etc…
* Call sequencing should be performed and well planned
* To ensure complete test coverage, create API test cases for all possible input combinations of the API.

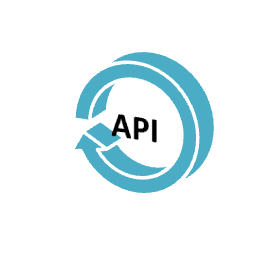
## **Types of Bugs that API testing detects**

* Fails to handle error conditions gracefully
* Unused flags
* Missing or duplicate functionality
* Reliability Issues. Difficulty in connecting and getting a response from API.
* Security Issues
* Multi-threading issues
* Performance Issues. API response time is very high.
* Improper errors/warning to a caller
* Incorrect handling of valid argument values
* Response Data is not structured correctly (JSON or XML)

## **API Testing Approach**

**API Testing Approach** is a predefined strategy or a method that the QA team will perform in order to conduct the API testing after the build is ready. This testing does not include the source code. The API testing approach helps to better understand the functionalities, testing techniques, input parameters and the execution of test cases.

Following points helps the user to do API Testing approach:



1. Understanding the functionality of the API program and clearly define the scope of the program
2. Apply testing techniques such as equivalence classes, boundary value analysis, and error guessing and write test cases for the API
3. Input Parameters for the API need to be planned and defined appropriately
4. Execute the test cases and compare expected and actual results.

## **Test Cases for API Testing:**

Test cases of API testing are based on

* **Return value based on input condition:** it is relatively easy to test, as input can be defined and results can be authenticated
* **Does not return anything:** When there is no return value, a behavior of API on the system to be checked
* **Trigger some other API/event/interrupt:** If an output of an API triggers some event or interrupt, then those events and interrupt listeners should be tracked
* **Update data structure:** Updating data structure will have some outcome or effect on the system, and that should be authenticated
* **Modify certain resources:** If API call modifies some resources then it should be validated by accessing respective resources

## **Types of Output of an API**

An output of API could be

1. Any type of data
2. Status (say Pass or Fail)
3. Call another API function.

Let’s look at an example of each of the above Types in this api testing tutorial

**Any Type of Data**

Example: There is an API function which should add two integer numbers.

Long add(int a, int b)

The numbers have to be given as input parameters. The output should be a summation of two integer numbers. This output needs to be verified with an expected outcome.

Calling needs to be done such as

add (1234, 5656)

Exceptions have to be handled if the number is exceeding the integer limit.

**Status (say Pass or Fail)**

Consider the below API function –

1. Lock()
2. Unlock()
3. Delete()

They return any value such as True (in case of success) or false (In case of error) as an output.

A more accurate [Test Case](https://www.guru99.com/test-case.html) would be, can call the functions in any of the scripts and later check for changes either in the database or the Application GUI.

**Calling of another API / Event**

****

In this case, we call one of the API functions which in turn will call another function.

For example – First API function can be used for deleting a specified record in the table and this function, in turn, calls another function to REFRESH the database.

## **Set-up of API Test environment**

* API Testing is different than other software testing types as GUI is not available, and yet you are required to setup initial environment that invokes API with a required set of parameters and then finally examines the test result.
* Hence, Setting up a testing environment for API automation testing seems a little complex.
* Database and server should be configured as per the application requirements.
* Once the installation is done, the API Function should be called to check whether that API is working.

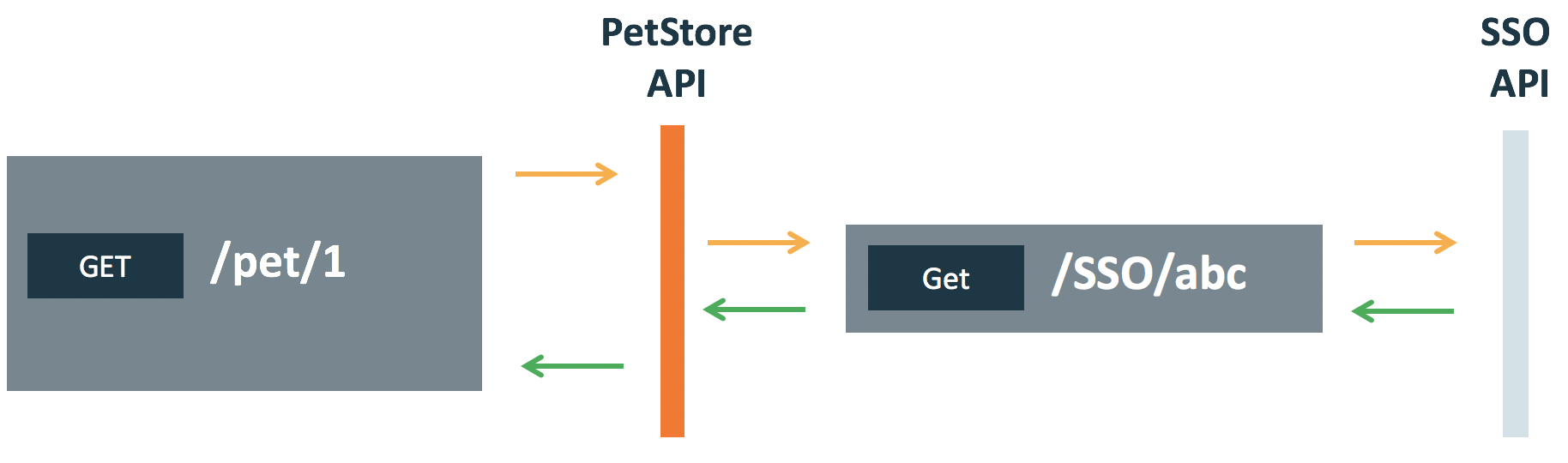
## Types of API Testing:

## Unit TestingScreen-Shot-2019-03-07-at-11-30-38-AM.png

While the testing world can be filled with misnomers, the easiest way to think about a "unit test" and APIs is testing a single endpoint, with a single request, looking for a single response or set of responses. Many times, this type of testing can be done manually via the command line and something like a cURL command or with lightweight tools like SoapUI.

## Integration Testing

Integration testing is the most often used form of API testing, as APIs are at the center of most integrations between internal or third-party services



## End-to-End Testing

End-to-End testing can help us validate the flow of data and information between a few different API connections.