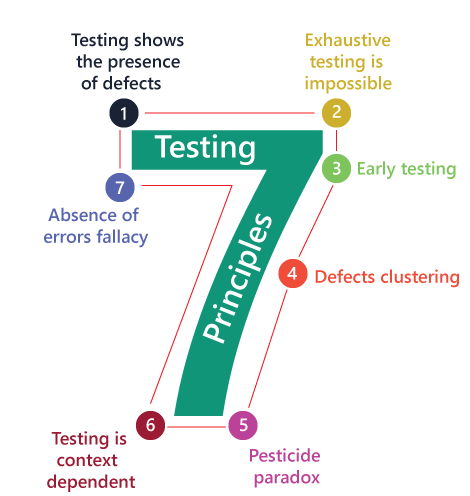
**Testing week-11**

* Principles of testing
* Need of testing
* Stages
* Testing process and activities
* Classification
* Testing strategies
* Levels of software testing
* Software testing types

(Integration testing, functional testing, end to-end testing)

# Principles of testing

1) This principle defined that if we are executing the same set of test cases again and again over a particular time, then these kinds of the test will not be able to find the new bugs in the software or the application.



1. **Testing shows the presence of defects**: The goal of software testing is to make the software fail. Software testing reduces the presence of defects. Software testing talks about the presence of defects and doesn’t talk about the absence of defects. Software testing can ensure that defects are present but it can not prove that software is defect-free. Even multiple testing can never ensure that software is 100% bug-free. Testing can reduce the number of defects but not remove all defects.
2. **Exhaustive testing is not possible**: It is the process of testing the functionality of the software in all possible inputs (valid or invalid) and pre-conditions is known as exhaustive testing. Exhaustive testing is impossible means the software can never test at every test case. It can test only some test cases and assume that the software is correct and it will produce the correct output in every test case. If the software will test every test case then it will take more cost, effort, etc., which is impractical.
3. **Early Testing**: To find the defect in the software, early test activity shall be started. The defect detected in the early phases of SDLC will be very less expensive. For better performance of software, software testing will start at the initial phase
4. **Defect clustering**: In a project, a small number of modules can contain most of the defects. Pareto Principle to software testing state that 80% of software defect comes from 20% of modules.
5. **Pesticide paradox**: Repeating the same test cases, again and again, will not find new bugs. So it is necessary to review the test cases and add or update test cases to find new bugs.
6. **Testing is context-dependent**: The testing approach depends on the context of the software developed. Different types of software need to perform different types of testing. For example, The testing of the e-commerce site is different from the testing of the Android application.
7. **Absence of errors fallacy:** If a built software is 99% bug-free but it does not follow the user requirement then it is unusable. It is not only necessary that software is 99% bug-free but it is also mandatory to fulfill all the customer requirements.

# Need of testing

* Software testing is the process of evaluating and verifying that a software product or application does what it is supposed to do. The benefits of testing include **preventing bugs, reducing development costs and improving performance**.
* A study conducted by National Institute of Standards and Technology (NIST) in 2002 reported that software bugs cost the U.S. economy 59.5 billion dollars annually.
* More than one-third of this cost could be avoided if better software testing was performed. Therefore testing is necessary as some errors can turn out to be expensive or dangerous.
* Every product needs to be checked to ensure there are no errors. If developers check their own product, there are chances that they might miss errors due to bad assumptions or blind spots. It is advisable to get the product checked by another individual who was not involved in product development.
* It is important to check the severity of the error and its consequences, as well.
* For software systems, some errors are important while others are not. You need to determine the impact of a software error.

# Stages of testing

* **Software Testing Life Cycle (STLC)** is a sequence of different activities performed during the software testing process.
* The STLC involves strategizing, planning, executing and completing test cycles.
* Each STLC phase is useful in its own way to achieve high-quality software releases.

## Characteristics of STLC:

* STLC is a fundamental part of [Software Development Life Cycle (SDLC)](https://practice.geeksforgeeks.org/problems/software-development-life-cycle) but STLC consists of only the testing phases.
* STLC starts as soon as requirements are defined or software requirement document is shared by stakeholders.
* STLC yields a step-by-step process to ensure quality software

## https://media.geeksforgeeks.org/wp-content/uploads/20190510015920/ppp10.pngPhases of STLC:

* **Requirement Analysis:**

Requirement Analysis is the first step of Software Testing Life Cycle (STLC). In this phase quality assurance team understands the requirements like what is to be tested. If anything is missing or not understandable then quality assurance team meets with the stakeholders to better understand the detail knowledge of requirement.

## Test Planning:

Test Planning is most efficient phase of software testing life cycle where all testing plans are defined. In this phase manager of the testing team calculates estimated effort and cost for the testing work. This phase gets started once the requirement gathering phase is completed.

## Test Case Development:

The test case development phase gets started once the test planning phase is completed. In this phase testing team note down the detailed test cases. Testing team also prepare the required test data for the testing. When the test cases are prepared then they are reviewed by quality assurance team.

## Test Environment Setup:

Test environment setup is the vital part of the STLC. Basically test environment decides the conditions on which software is tested. This is independent activity and can be started along with test case development. In this process the testing team is not involved. Either the **developer or the customer** creates the testing environment.

## Test Execution:

After the test case development and test environment setup ,test execution phase gets started. In this phase testing team start executing test cases based on prepared test cases in the earlier step.

## Test Closure:

This is the last stage of STLC in which the process of testing is analyzed.

# Testing process and activities

These are 11 steps [software testing](https://www.geeksforgeeks.org/software-testing-basics/) process is an experience based practical approach for solution to test assignment.

These are explained as following below.

## Step-1: Assess Development Plan and Status –

This initiative may be prerequisite to putting together Verification, Validation, and Testing Plan .During this step, testers **challenge completeness and correctness** of plan. Based on extensiveness and completeness of Project Plan testers can estimate quantity of resources they’re going to got to test implemented software solution.

## Step-2: Develop the Test Plan –

Forming plan for testing will follow an equivalent pattern as any software planning process. The structure of all plans should be an equivalent, but content will vary.

## Step-3: Test Software Requirements –

Incomplete, inaccurate, or inconsistent requirements cause most software failures. Testers, through verification, must determine that requirements are accurate, complete, and they do not conflict with another.

## Step-4: Test Software Design –

This step tests both external and internal design primarily through verification techniques. The testers are concerned that planning will achieve objectives of wants, also because design being effective and efficient on designated hardware.

## Step-5: Build Phase Testing –

Experience has shown that it’s significantly cheaper to spot defects during development phase, than through dynamic testing during test execution step.

## Step-6: Execute and Record Result –

This involves testing of code during dynamic state. The approach, methods, and tools are used in test plan are going to be validate that executable code actually meets stated software requirements, and therefore the structural specifications of design, coding are needed

## Step-7: Acceptance Test –

Acceptance testing enables users to gauge applicability and usefulness of software in performing their day-to-day job functions. This tests what user believes software should perform.

## Step-8: Report Test Results –

Test reporting is continuous process. It may be both oral and written. It is important that defects and concerns be reported to the appropriate parties as early as possible, so that corrections can be made at the lowest possible cost.

## Step-9: The Software Installation –

Once test team has confirmed that software is prepared for production use, power to execute that software during production environment should be tested. This tests interface to operating software, related software, and operating procedures.

## Step-10: Test Software Changes –

While this is often shown as Step 10, within context of performing maintenance after software is implemented, concept is additionally applicable to changes throughout implementation process. Whenever requirements changes, test plan must change, and impact of that change on software systems must be tested and evaluate.

## Step-11: Evaluate Test Effectiveness –

Testing improvement can best be achieved by evaluating effectiveness of testing at top of every software test assignment. While this assessment is primarily performed by testers, it should involve developers, users of software, and quality assurance professionals .

# Testing activities

The activities of testing can be divided into the following basic steps:

1. Planning and Control
2. Analysis and Design
3. Implementation and Execution
4. Evaluating exit criteria and Reporting
5. Test Closure activities

## Planning and Control

**Test Planning**: Test planning involves producing a document that describes an overall approach and test objectives. It involves reviewing the test basis, identifying the test conditions based on analysis of test items, writing test cases and Designing the test environment.

Completion or exit criteria must be specified so that we know when testing (at any stage) is complete.

**Control** This is the activity of comparing actual progress against the plan, and reporting the status, including deviations from the plan. It involves taking actions necessary to meet the mission and objectives of the project.

## Analysis and Design

Test analysis and Test Design has the following major tasks:

* To review the test basis. The test basis is the information on which test cases are based, such as requirements, design specifications, product risk analysis, architecture and interfaces
* To identify test conditions
* To design the tests
* To design the test environment set-up and identify the required infrastructure and tools

## Implementation and Execution

Test execution involves actually running the specified test on a computer system either manually or by using an automated test tool. It is a Fundamental Test Process in which actual work is done.

## Test implementation has the following major task:

* To develop and prioritize test cases by using techniques and create test data for those tests.
* To create test suites from the test cases for efficient test execution. Test suite is a collection of test cases that are used to test a software program
* To re-execute the tests that previously failed in order to confirm a fix.
* To log the outcome of the test execution. A test log is the status of the test case (pass/fail).
* To compare actual results with expected results.

## Evaluating Exit criteria and Reporting

Evaluating exit criteria is a process defining when to stop testing. It depends on coverage of code, functionality or risk. Basically it also depends on business risk, cost and time and vary from project to project. Exit criteria come into picture, when:

* Maximum test cases are executed with certain pass percentage
* Bug rate falls below certain level
* When we achieve the deadlines

## 5) Test Closure activities:

Test closure activities are done when software is ready to be delivered. The testing can be closed for the other reasons also like:

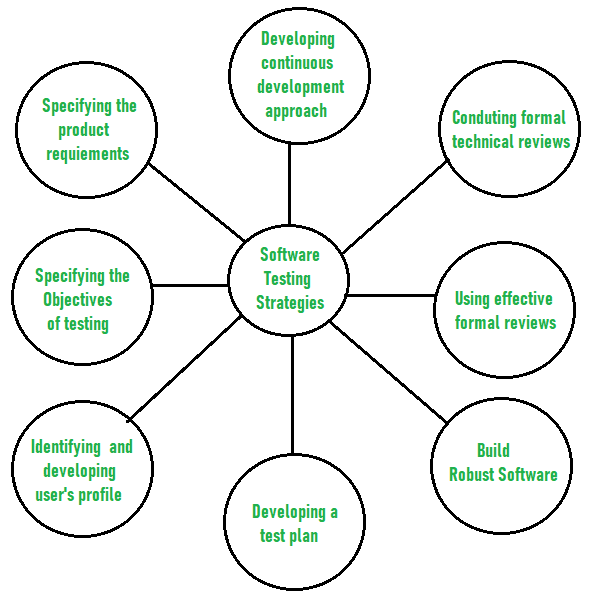
* When a project is cancelled
* When some target is achieved

## Test closure activities have the following major tasks:

* To check which planned deliverables are actually delivered and to ensure that all incident reports have been resolved

# Testing strategies

The main objective of software testing is to design the tests in such a way that it systematically finds different types of errors without taking much time and effort so that less time is required for the development of the software.



## The overall strategy for testing software includes:

1. **Before testing starts, it’s necessary to identify and specify the requirements of the product in a quantifiable manner.**

Different characteristics quality of the software is there such as maintainability that means the ability to update and modify, the probability that means to find and estimate any risk, and usability that means how it can easily be used by the customers or end-users. All these characteristic qualities should be specified in a particular order to obtain clear test results without any error.

## Specifying the objectives of testing in a clear and detailed manner.

Several objectives of testing are there such as effectiveness that means how effectively the software can achieve the target, any failure that means inability to fulfill the requirements and perform functions, and the cost of defects or errors that mean the cost required to fix the error. All these objectives should be clearly mentioned in the test plan.

## For the software, identifying the user’s category and developing a profile for each user.

Use cases describe the interactions and communication among different classes of users

and the system to achieve the target. So as to identify the actual requirement of the users and then testing the actual use of the product.

## Developing a test plan to give value and focus on rapid-cycle testing.

Rapid Cycle Testing is a type of test that improves quality by identifying and measuring the any changes that need to be required for improving the process of software. Therefore, a test plan is an important and effective document that helps the tester to perform rapid cycle testing.

## Robust software is developed that is designed to test itself.

The software should be capable of detecting or identifying different classes of errors. Moreover, software design should allow automated and regression testing which tests the software to find out if there is any adverse or side effect on the features of software due to any change in code or program.

## Before testing, using effective formal reviews as a filter.

Formal technical reviews is technique to identify the errors that are not discovered yet. The effective technical reviews conducted before testing reduces a significant amount of testing efforts and time duration required for testing software so that the overall development time of software is reduced.

## Conduct formal technical reviews to evaluate the nature, quality or ability of the test strategy and test cases.

The formal technical review helps in detecting any unfilled gap in the testing approach. Hence, it is necessary to evaluate the ability and quality of the test strategy and test cases by technical reviewers to improve the quality of software.

## For the testing process, developing a approach for the continuous development.

A test strategy that is already measured should be used for software testing to measure and control the quality during the development of software.

# Classification of testing

* Tests could be classified according to the type of activity measured, according to the method of constructing the tests, according to the purpose for which the test is designed, or according to the actual function that.
* Software testing is generally classified into two main broad categories: functional testing and non-functional testing.

**Levels of software testing**

There are different levels of testing :

1. [**Unit Testing**](https://www.geeksforgeeks.org/unit-testing-software-testing/) **:**

In this type of testing, errors are detected individually from every component or unit by individually testing the components ,to ensure that if they are fit for use by the developers. It is the smallest testable part of the software.

1. [**Integration Testing**](https://www.geeksforgeeks.org/software-engineering-integration-testing/) **:**

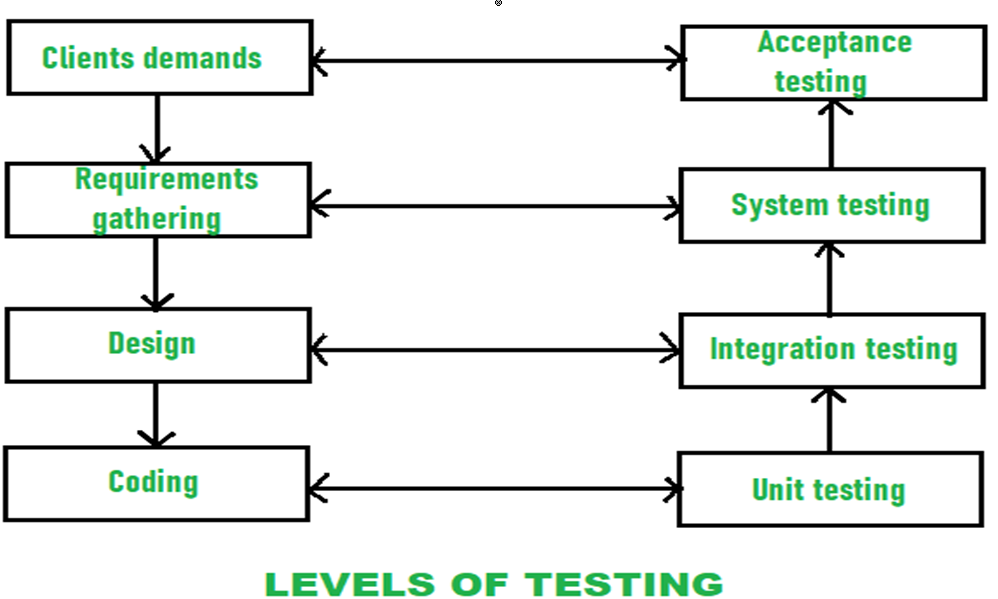
In this testing, two or more modules which are unit tested are integrated to test i.e. if these integrated modules work as per the expectation or not and interface errors are also detected.

1. [**System Testing**](https://www.geeksforgeeks.org/system-testing/) **:**

In system testing, complete and integrated softwares are tested i.e. all the system elements forming the system is tested as a whole to meet the requirements of the system.

1. [**Acceptance Testing**](https://www.geeksforgeeks.org/acceptance-testing-software-testing/) **:**

It is a kind of testing conducted to ensure whether the requirement of the users are fulfilled prior to its delivery and the software works correctly in the user’s working environment.

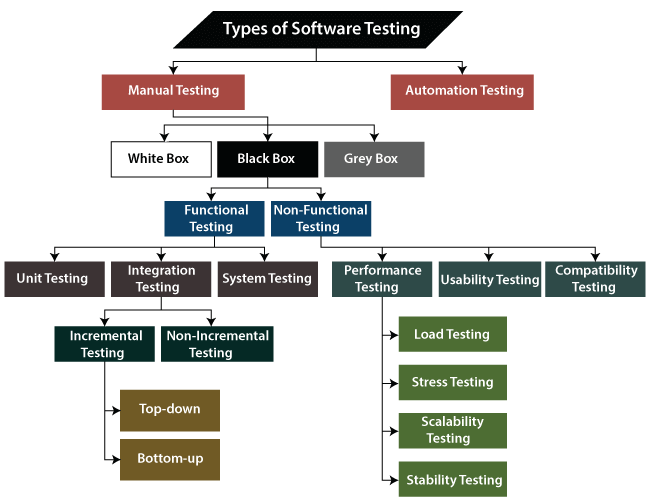


While performing the software testing, following [Testing principles](https://www.geeksforgeeks.org/types-software-testing/) must be applied by every software engineer:

* Planning of tests that how tests will be conducted should be done long before the beginning of the test.
* The Pareto principle can be applied to software testing- 80% of all errors identified during testing will likely be traceable to 20% of all program modules.
* Testing should begin “in the small” and progress toward testing “in the large”.
* Exhaustive testing which simply means to test all the possible combinations of data is not possible.
* Testing conducted should be most effective and for this purpose, an independent third party is required.

**Software testing types**

* 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.



* Software testing is the execution of the software to find defects.
* The purpose of having a testing type is to confirm the **AUT** (Application Under Test).
* To start testing, we should have a **requirement, application-ready, necessary resources available**.

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**.
* We do not require any precise knowledge of any testing tool to execute the manual test cases.

**Classification of Manual Testing**

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

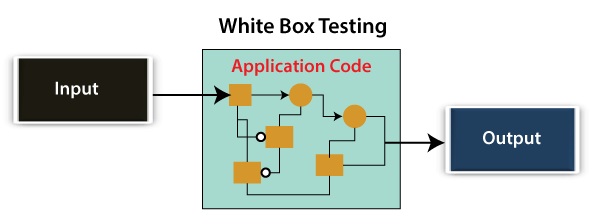
**o White Box Testing**

**o Black Box Testing**

**o Grey Box Testing**

**White Box Testing**

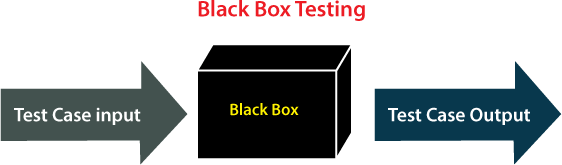
* In white-box testing, the developer will inspect every line of code before handing it over to the testing team or the concerned test engineers.
* Subsequently, the code is noticeable for developers throughout testing; that's why this process is known as **WBT (White Box Testing)**.
* In other words, 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

* 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.
* Then, the developers will fix those defects, do one round of White box testing, and send it to the testing team.
* Here, fixing the bugs means the defect is resolved, and the particular feature is working according to the given requirement.
* 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

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

* + **Unit Testing**
  + **Integration Testing**
  + **System Testing**

1. **Unit Testing**

* Unit testing is the first level of functional testing in order to test any software. In this, the test engineer will test the module of an application independently is called **unit testing**.
* The primary objective of executing the unit testing is to confirm the unit components with their performance.

## 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 analysis the data flow between them if they are working fine or not.
* If these modules are working fine, then we can add one more module and test again. And we can continue with the same process to get better results.
* In other words, we can say that incrementally adding up the modules and test the data flow between the modules is known as **Incremental integration testing**.

## Types of Incremental Integration Testing

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

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

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**.

## 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**.

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

## Types of Non-functional Testing

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

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

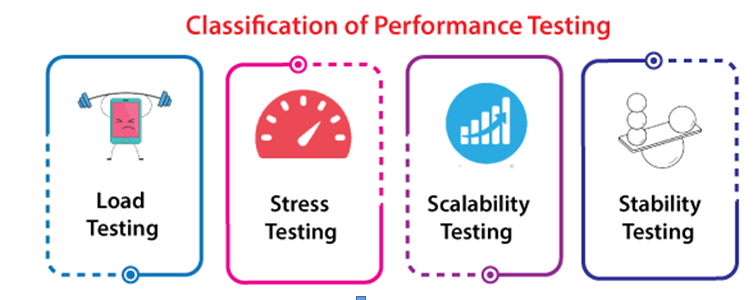
1. **Performance Testing**

* 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**

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

## Stress Testing

* It is used to analyze the user-friendliness and robustness of the software
* Primarily, stress testing is used for critical software, but it can also be used for all types of software applications.

## Scalability Testing

* To analysis, the **application's performance by enhancing** or **reducing the load** in particular balances is known as **scalability testing**.
* In scalability testing, we can also check the **system, processes, or database's ability** to meet an upward need. And in this, the **Test Cases** are designed and implemented efficiently.

## Stability Testing

* Stability testing is a procedure where we evaluate the application's performance by applying the load for a precise time.
* It mainly checks the constancy problems of the application and the efficiency of a developed

product. In this type of testing, we can rapidly find the system's defect even in a stressful situation.

1. **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.

1. **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.

**Grey Box Testing**

* Another part of **manual testing** is **Grey box testing**. It is a **collaboration of black box and white box testing**.
* Grey box testing is performed by a person who knows coding as well as testing.



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

**Automation Testing**

* The most significant part of Software testing is Automation testing.
* It uses specific tools to automate manual design test cases without any human interference.
* Automation testing is the best way to enhance the efficiency, productivity, and coverage of Software testing.
* It is used to re-run the test scenarios, which were executed manually, quickly, and repeatedly.
* In other words, we can say that whenever we are testing an application by using some tools is known as **automation testing**.

**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**.

**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 documented 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.
* Regression testing is the most suitable testing for automation tools. As per the project type and accessibility of resources, regression testing can be similar to **Retesting**.
* In other words, we can say that whenever there is a new release for some project, then we can perform Regression Testing, and due to a new feature may affect the old features in the earlier releases.

**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**.

**Exploratory Testing**

* Whenever the requirement is missing, early iteration is required, and the testing team has experienced testers when we have a critical application. New test engineer entered into the team then we go for the **exploratory testing**.
* To execute the exploratory testing, we will first go through the application in all possible ways, make a test document, understand the flow of the application, and then test the application.

**Adhoc Testing**

* Testing the application randomly as soon as the build, is in the checked sequence is known as **Adhoc testing**.
* It is also called **Monkey testing and Gorilla testing**. In Adhoc testing, we will check the application in contradiction of the client's requirements, that's why it is also known as **negative testing**.
* When the end-user using the application casually, and he/she may detect a bug. Still, the specialized test engineer uses the software thoroughly, so he/she may not identify a similar detection.

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

be used globally.