**# FUNCTIONAL TESTINGS:**

1.Assertion testing

2.Gorilla testing

3.Sanity testing

4.Monkey testing

5.Smoke testing

6.Exploratory testing

7.Mutation testing

8.Benchmark testing

**1.ASSERTION TESTING:**

**Definition:**

An **assertion** is a **boolean expression**.it is a concept of **functional testing**. It is used to test a **logical expression**. An assertion is true if the logical expression that is being tested is true and there are no bugs in the program. Assertion testing can be used at any particular stage of the program.

EX: 3<4=true, assertion true and logical expression true then their is no bugs.

7>10=false, assertion false and logical expression false then bugs are their.

**Benefits of Assertions:**

The main advantage of having assertions is to identify defects in a program. The usefulness of assertions include:

* It is used to detect subtle errors which might go unnoticed.
* It is used to detect errors sooner after they occur.
* Make a statement about the effects of the code that is guaranteed to be true.

**Limitations:**

* Failing to report a bug that exists.
* Reporting an error when it does not exist.
* Can lead to other side effects
* Can Take time to execute if it contains errors and occupies memory as well.

**2.GORILLA TESTING:**

Gorilla testing is a software testing technique that repeatedly applies inputs on a module to ensure it is functioning correctly and that there are no bugs.

**3.SANITY TESTING:**

Sanity testing is performed on stable builds and it is also known as a variant of regression testing.Sanity testing was performed when we are receiving software build (with minor code changes) from the development team. It is a checkpoint to assess if testing for the build can proceed or not.

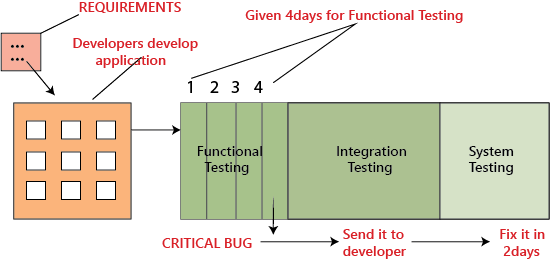
In other words, we can say that sanity testing is performed to make sure that all the defects have been solved and no added issues come into the presence because of these modifications.

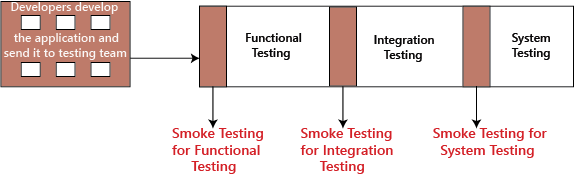
**4.MONKEY TSETING:**

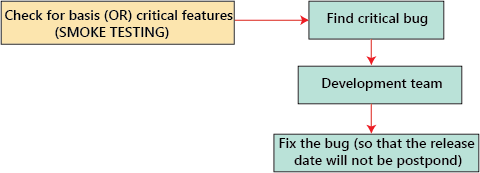
Monkey Testing is a software testing technique in which the tester enters any random inputs into the software application without predefined test cases and checks the behavior of the software application, whether it crashes or not. The purpose of Monkey testing is to find the bugs and errors in the software application using experimental techniques.

**5.SMOKE TESTING:**

Smoke Testing is a software testing process that determines whether the deployed software build is stable or not. Smoke testing is a confirmation for QA team to proceed with further software testing. It consists of a minimal set of tests run on each build to test software functionalities. Smoke testing is also known as “Build Verification Testing” or “Confidence Testing.”







**6.EXPLORATOY TESTING:**

exploratory testing?If requirement does not exist, then we do one round of exploratory testing.So, for this first, we will be exploring the application in all possible ways, understanding the flow of the application, preparing a test document and then testing the application, this approach is known as exploratory testing.

**7.MUTATION TESTING:**

Mutation Testing is a type of software testing in which certain statements of the source code are changed/mutated to check if the test cases are able to find errors in source code. The goal of Mutation Testing is ensuring the quality of test cases in terms of robustness that it should fail the mutated source code.

**8. Fuzz Testing** or Fuzzing is a software testing technique of putting invalid or random data called FUZZ into software system to discover coding errors and security loopholes. The purpose of fuzz testing is inserting data using automated or semi-automated techniques and testing the system for various exceptions like system crashing or failure of built-in code, etc.

**# NON-FUNCTIONAL TESTINGS:**

1.Load testing

2.Strees testing

3.Usability testing

4.Performance testing

5.Volume testing

6.Scalability testing

7.Security testing

**1.LOAD TSETING:**

Load Testing is a non-functional software testing process in which the performance of software application is tested under a specific expected load. It determines how the software application behaves while being accessed by multiple users simultaneously. The goal of Load Testing is to improve performance bottlenecks and to ensure stability and smooth functioning of software application before deployment.

**Load Testing Tools:**

1. Apache JMeter

2. Web Load

3. Neo Load

4. Load Ninja

5. HP Performance Tester

6. Load UI Pro

7. Load View

**2.STRESS TESTING:**

Stress testing (sometimes called torture testing) is a form of deliberately intense or thorough testing used to determine the stability of a given system, critical infrastructure or entity. Stress testing involves testing the application under varying load. Extremely large numbers of concurrent users try to log into the application. Database linked to the website shuts down when the website tries to reach it from the front end. Data in added in extremely large quantity in the database. Stress Testing is a type of software testing that verifies stability & reliability of software application. The goal of Stress testing is measuring software on its robustness and error handling capabilities under extremely heavy load conditions and ensuring that software doesn’t crash under crunch situations.

**3.USABILITY TSETING:**

Usability testing refers to evaluating a product or service by testing it with representative users. Typically, during a test, participants will try to complete typical tasks while observers watch, listen and takes notes.

to check the usability or ease of using a software product. Checking the user-friendliness, efficiency, and accuracy of the application is known as **Usability Testing.**

**Parameters:**

Efficiency

Memorability

Accuracy

Learnability

Satisfaction

Errors

**4.PERFORMANCE TESTING:**

Performance testing is in general a testing practice performed to determine how a system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.

Types of Performance Testing:

• Load

• Stress

• Spike

• Endurance

• Scalability

• Volume

**5.VOLUME TESTING:**

Volume Testing is a type of software testing which is carried out to test a software application with a certain amount of data. . In volume testing a huge volume of data is acted upon the software. It is basically performed to analyze the performance of the system by increasing the volume of data in the database. Volume testing is performed to study the impact on response time and behavior of the system when the volume of data is increased in the database.Volume Testing is also known as Flood Testing.

**6.SCALABILITY TESTING:**

scalability testing, which comes under the non-functional testing of software testing.It is used to check an application's performance by increasing or decreasing the load in particular scales known as scalability testing. It is executed at a hardware, software, or database level.

**7.SECURITY TESTING:**

The main goal of Security Testing is to identify the threats in the system and measure its potential vulnerabilities, so the threats can be encountered and the system does not stop functioning or can not be exploited. It also helps in detecting all possible security risks in the system and helps developers to fix the problems through coding.

types of security Vulnerability Scanning. ...

Security Scanning. ...

Penetration Testing. ...

Security Audit/ Review. ...

Ethical Hacking. ...

Risk Assessment. ...

Posture Assessment. ...

Authentication.

**8. BENCHMARK TESTING:**

A Benchmark in Performance Testing is a metric or a point of reference against which software products or services can be compared to assess the quality measures. In other words, Benchmark means a set standard that helps to determine the quality of a software product or service.

the user experience cannot be quantified in numbers, but the time a user spends on a webpage due to good UI can be quantified.

Benchmark Testing is not a term related to just software testing, but it also deals with Hardware Testing

**Soak Testing** is a type of performance testing where the software under load is tested to validate that it can endure heavy loads for the extended period.

**Soak Testing** is a type of software testing in which system is tested under huge load over a continuous availability period to check the behavior of the system under production use.   
Soak Testing tests that system can withstand a huge volume of the load for an extended period of time.

This testing is performed at the system level to find whether the system will stand up to a very high volume of usage or not. It also tests that what would happen outside the design expectations of the system.

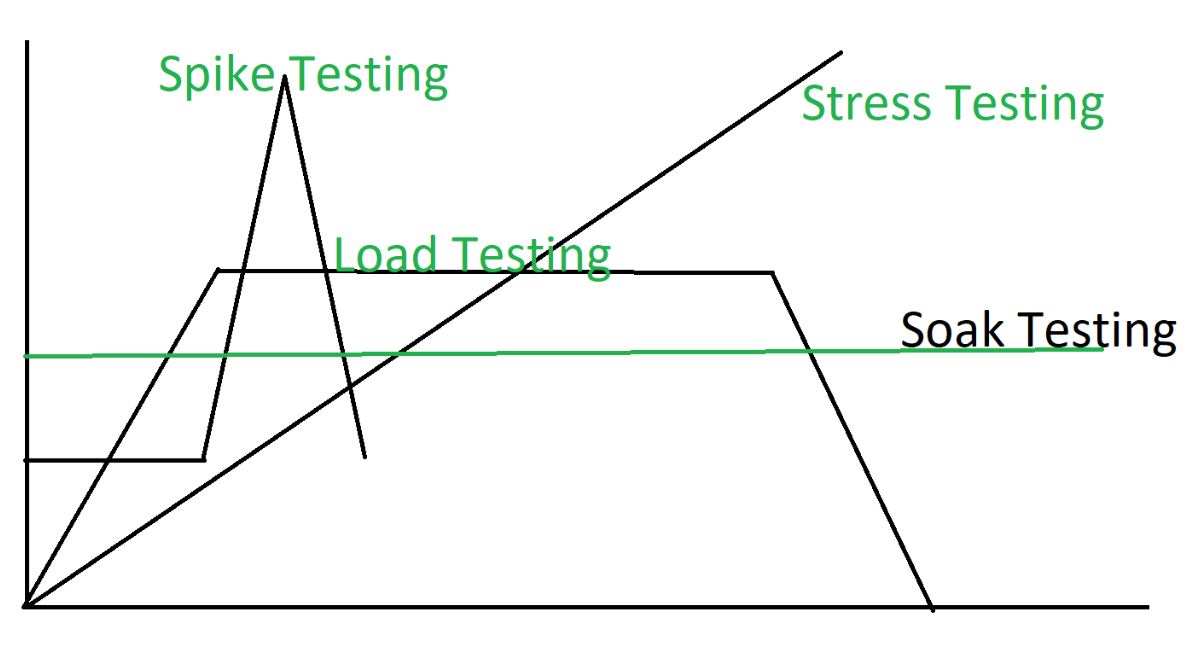
**Objective of Soak Testing:**   
The objective of Soak Testing is to:

* To check the system behavior under heavy load for long time.
* To predict the failure caused by the heavy load.
* To test the performance of the system.
* To make the system reliable and stable.

**Failures detected by Soak Testing:**   
The failures or issues detected by the Soak Testing are:

* **Memory Leaks:**   
  Soak testing detects the serious memory leaks which can cause application crash or lead up to the crash of operating system.
* **Layer Connections Failure:**   
  Soak testing finds the failure of close connections between the layers of the system that can interrupt the modules of the system.
* **Database Connections Failure:**   
  Soak testing detects the failure of close database connections under some conditions that may crash the complete system.
* **Response Time Degradation:**   
  Soak testing finds the degradation of response time of the system as the system becomes less efficient and takes more time to response.

**Soak Testing Graphical Representation:**



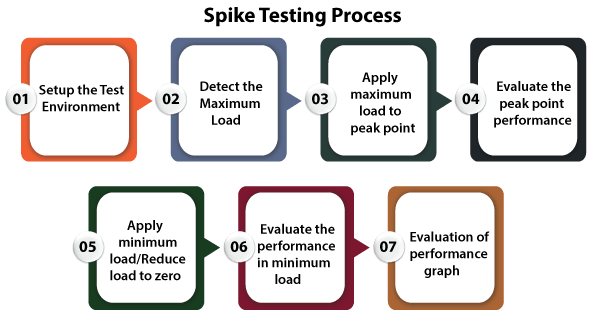
**Advantages of Soak Testing:** The advantages of Soak Testing are:

* Soak Testing improves the performance of the system.
* Soak Testing increases the resistance of the system.
* Soak Testing makes the system work under heavy load.
* It improves the behavior of the system under the heavy load for long time.

**Disadvantages of Soak Testing:**

* It is difficult to know or predict how long that the test will run.
* Utilization of the memory is high due to more number of users accessing the web application.
* It is a time consumption process and it is not recommended for the project which has strict deadlines.
* Manual soak testing often takes lot of time to complete the test and often provides wrong test results.
* If we run this technique in a live environment then it will lead to loss of data or data corruption.

**Spike testing** is **a type of performance testing in which an application receives a sudden and extreme increase or decrease in load**. The goal of spike testing is to determine the behavior of a software application when it receives extreme variations in traffic.



**Connection testing:**

An internet speed test **measures the connection speed and quality of your connected device to the internet**. It does so by running multiple consecutive tests that analyze different aspects of your internet connection, namely ping (latency), download speed, and upload speed.

**Production testing**

Testing in production, rather, refers to **the continuous testing of the application in the production environment, after a deployment**.

| **Overriding** | **Overloading** |
| --- | --- |
| Implements “runtime polymorphism” | Implements “compile time polymorphism” |
| The method call is determined at runtime based on the object type | The method call is determined at compile time |
| Occurs between superclass and subclass | Occurs between the methods in the same class |
| Have the same signature (name and method arguments) | Have the same name, but the parameters are different |
| On error, the effect will be visible at runtime | On error, it can be caught at compile time |

| **sr. No.** | **Key** | **String Buffer** | **String Builder** |
| --- | --- | --- | --- |
| 1 | Basic | StringBuffer was introduced with the initial release of Java | It was introduced in Java 5 |
| 2 | Synchronized | It is synchronized | It is not synchronized |
| 3 | Performance | It is thread-safe. So, multiple threads can’t access at the same time, therefore, it is slow. | It is not thread-safe hence faster than String Buffer |
| 4 | Mutable | It is mutable. We can modify string without creating an object | It is also mutable |
| 5 | Storage | Heap | Heap |

# **What is 'Software Testing'**

**Definition:**The process or method of finding error/s in a software application or program so that the application functions according to the end user's requirement is called software testing.

## Functional Testing

There are four main types of functional testing.

### 1) Unit Testing

[Unit testing](https://www.softwaretestinghelp.com/unit-testing/) is a type of software testing which is done on an individual unit or component to test its corrections. Typically, Unit testing is done by the developer at the application development phase.

**For example,** there is a simple calculator application. The developer can write the unit test to check if the user can enter two numbers and get the correct sum for addition functionality.

**a) White Box Testing**

[White box](https://www.softwaretestinghelp.com/white-box-testing-techniques-with-example/) testing is a test technique in which the internal structure or code of an application is visible and accessible to the tester.

**b) Gorilla Testing**

Gorilla testing is a test technique in which the tester and/or developer test the module of the application thoroughly in all aspects. **For example,** the tester is testing the pet insurance company’s website, which provides the service of buying an insurance policy, tag for the pet, Lifetime membership.

### 2) Integration Testing

[Integration testing](https://www.softwaretestinghelp.com/what-is-integration-testing/) is a type of software testing where two or more modules of an application are logically grouped together and tested as a whole.

**For example,** a user is buying a flight ticket from any airline website. Users can see flight details and payment information while buying a ticket, but flight details and payment processing are two different systems.

**a) Gray box testing**

As the name suggests, gray box testing is a combination of white-box testing and black-box testing. Testers have partial knowledge of the internal structure or code of an application.

### 3) System Testing

**a) End to End Testing**

It involves testing a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate. **For example,** a tester is testing a pet insurance website. End to End testing involves testing of buying an insurance policy, LPM, tag, adding another pet, updating credit card information on users’ accounts, updating user address information, receiving order confirmation emails and policy documents.

**b) Black Box Testing**

Blackbox testing is a software testing technique in which testing is performed without knowing the internal structure, design, or code of a system under test. Testers should focus only on the input and output of test objects.

**c) Smoke Testing**

Smoke testing is performed to verify that basic and critical functionality of the system under test is working fine at a very high level. **For example,** tester is testing pet insurance website. Buying an insurance policy, adding another pet, providing quotes are all basic and critical functionality of the application.

**d) Sanity Testing**

Sanity testing is performed on a system to verify that newly added functionality or bug fixes are working fine. Sanity testing is done on stable build. It is a subset of the regression test. **For example,** a tester is testing a pet insurance website. There is a change in the discount for buying a policy for second pet. Then sanity testing is only performed on buying insurance policy module.

**e) Happy path Testing**

The objective of Happy Path Testing is to test an application successfully on a positive flow. It does not look for negative or error conditions. The focus is only on valid and positive inputs through which the application generates the expected output.

**f) Monkey Testing**

[Monkey Testing](https://www.softwaretestinghelp.com/what-is-monkey-testing-in-software-testing/) is carried out by a tester, assuming that if the monkey uses the application, then how random input and values will be entered by the Monkey without any knowledge or understanding of the application.

### 4) Acceptance Testing

Acceptance testing is a type of testing where client/business/customer test the software with real time business scenarios.

The client accepts the software only when all the features and functionalities work as expected. This is the last phase of testing, after which the software goes into production. This is also called User Acceptance Testing (UAT).

**a) Alpha Testing**

Alpha testing is a type of acceptance testing performed by the team in an organization to find as many defects as possible before releasing software to customers.

**For example,** the pet insurance website is under UAT. UAT team will run real-time scenarios like buying an insurance policy, buying annual membership, changing the address, ownership transfer of the pet in a same way the user uses the real website.

**b) Beta Testing**

[Beta Testing](https://www.softwaretestinghelp.com/beta-testing/) is a type of software testing which is carried out by the clients/customers. It is performed in the **Real Environment**before releasing the product to the market for the actual end-users.

**c) Operational acceptance testing (OAT)**

Operational acceptance testing of the system is performed by operations or system administration staff in the production environment. The purpose of operational acceptance testing is to make sure that the system administrators can keep the system working properly for the users in a real-time environment.

**The focus of the OAT is on the following points:**

* Testing of backup and restore.
* Installing, uninstalling, upgrading software.
* The recovery process in case of natural disaster.
* User management.
* Maintenance of the software.

## Non-Functional Testing

There are four main types of functional testing.

### #1) Security Testing

It is a type of testing performed by a special team. Any hacking method can penetrate the system.

[Security Testing](https://www.softwaretestinghelp.com/how-to-test-application-security-web-and-desktop-application-security-testing-techniques/) is done to check how the software, application, or website is secure from internal and/or external threats. This testing includes how much software is secure from malicious programs, viruses and how secure & strong the authorization and authentication processes are.

It also checks how software behaves for any hacker’s attack & malicious programs and how software is maintained for data security after such a hacker attack.

**a) Penetration Testing**

Penetration Testing or Pen testing is the type of security testing performed as an authorized cyberattack on the system to find out the weak points of the system in terms of security.

Pen testing is performed by outside contractors, generally known as ethical hackers. That is why it is also known as ethical hacking. Contractors perform different operations like SQL injection, URL manipulation, Privilege Elevation, session expiry, and provide reports to the organization.

**Notes:** Do not perform the Pen testing on your laptop/computer. Always take written permission to do pen tests.

### 2) Performance Testing

[Performance testing](https://www.softwaretestinghelp.com/introduction-to-performance-testing-loadrunner-training-tutorial-part-1/) is testing of an application’s stability and response time by applying load.

The word stability means the ability of the application to withstand in the presence of load. Response time is how quickly an application is available to users. Performance testing is done with the help of tools. Loader.IO, JMeter, LoadRunner, etc. are good tools available in the market.

**a) Load testing**

[Load testing](https://www.softwaretestinghelp.com/load-testing/) is testing of an application’s stability and response time by applying load, which is equal to or less than the designed number of users for an application.

**For example,** your application handles 100 users at a time with a response time of 3 seconds, then load testing can be done by applying a load of the maximum of 100 or less than 100 users. The goal is to verify that the application is responding within 3 seconds for all the users.

**b) Stress Testing**

[Stress testing](https://www.softwaretestinghelp.com/stress-testing/) is testing an application’s stability and response time by applying load, which is more than the designed number of users for an application.

**For example,** your application handles 1000 users at a time with a response time of 4 seconds, then stress testing can be done by applying a load of more than 1000 users. Test the application with 1100,1200,1300 users and notice the response time. The goal is to verify the stability of an application under stress.

**c) Scalability Testing**

Scalability testing is testing an application’s stability and response time by applying load, which is more than the designed number of users for an application. **For example,** your application handles 1000 users at a time with a response time of 2 seconds, then scalability testing can be done by applying a load of more than 1000 users and gradually increasing the number of users to find out where exactly my application is crashing.

Let’s say my application is giving response time as follows:

* 1000 users -2 sec
* 1400 users -2 sec
* 4000 users -3 sec
* 5000 users -45 sec
* 5150 users- crash – This is the point that needs to identify in scalability testing

**d) Volume testing (flood testing)**

Volume testing is testing an application’s stability and response time by transferring a large volume of data to the database. Basically, it tests the capacity of the database to handle the data.

**e) Endurance Testing (Soak Testing)**

Endurance testing is testing an application’s stability and response time by applying load continuously for a longer period to verify that the application is working fine. **For example,** car companies soak testing to verify that users can drive cars continuously for hours without any problem.

### 3) Usability Testing

[Usability testing](https://www.softwaretestinghelp.com/usability-testing-guide/) is testing an application from the user’s perspective to check the look and feel and user-friendliness.

**For example,** there is a mobile app for stock trading, and a tester is performing usability testing. Testers can check the scenario like if the mobile app is easy to operate with one hand or not, scroll bar should be vertical, background color of the app should be black and price of and stock is displayed in red or green color.

The main idea of usability testing of this kind of app is that as soon as the user opens the app, the user should get a glance at the market.

**a) Exploratory testing**

Exploratory Testing is informal testing performed by the testing team. The objective of this testing is to explore the application and look for defects that exist in the application. Testers use the knowledge of the business domain to test the application. Test charters are used to guide the exploratory testing.

**b) Cross browser testing**

Cross browser testing is testing an application on different browsers, operating systems, mobile devices to see look and feel and performance.

Why do we need cross-browser testing? The answer is different users use different operating systems, different browsers, and different mobile devices. The goal of the company is to get a good user experience regardless of those devices.

Browser stack provides all the versions of all the browsers and all mobile devices to test the application. For learning purposes, it is good to take the free trial given by browser stack for a few days.

**c) Accessibility Testing**

The aim of [Accessibility Testing](https://www.softwaretestinghelp.com/what-is-web-accessibility-testing/) is to determine whether the software or application is accessible for disabled people or not.

Here, disability means deafness, color blindness, mentally disabled, blind, old age, and other disabled groups. Various checks are performed, such as font size for visually disabled, color and contrast for color blindness, etc.

### 4) Compatibility testing

This is a testing type in which it validates how software behaves and runs in a different environment, web servers, hardware, and network environment.

[Compatibility testing](https://www.softwaretestinghelp.com/software-compatibility-testing/) ensures that software can run on different configuration, different databases, different browsers, and their versions. The testing team performs compatibility testing.

### Other Types of Testing

**Ad-hoc Testing**

The name itself suggests that this testing is performed on an [ad-hoc](https://www.softwaretestinghelp.com/ad-hoc-testing/) basis, i.e., with no reference to the test case and also without any plan or documentation in place for this type of testing.

The objective of this testing is to find the defects and break the application by executing any flow of the application or any random functionality.

Ad-hoc testing is an informal way of finding defects and can be performed by anyone in the project. It is difficult to identify defects without a test case, but sometimes it is possible that defects found during ad-hoc testing might not have been identified using the existing test cases.

**Back-end Testing**

Whenever an input or data is entered on the front-end application, it is stored in the database and the testing of such database is known as Database Testing or Backend Testing.

There are different databases like SQL Server, MySQL, Oracle, etc. Database Testing involves testing of table structure, schema, stored procedure, data structure, and so on. In Back-end Testing, GUI is not involved, the testers are directly connected to the database with proper access and testers can easily verify data by running a few queries on the database.

There can be issues identified like data loss, deadlock, data corruption, etc during this back-end testing and these issues are critical to fixing before the system goes live into the production environment.

**Browser Compatibility Testing**

This is a sub-type of Compatibility Testing (which is explained below) and is performed by the testing team.

[Browser Compatibility Testing](https://www.softwaretestinghelp.com/how-is-cross-browser-testing-performed/) is performed for web applications and ensures that the software can run with a combination of different browsers and operating systems. This type of testing also validates whether a web application runs on all versions of all browsers or not.

**Backward Compatibility Testing**

It is a type of testing that validates whether the newly developed software or updated software works well with the older version of the environment or not.

Backward Compatibility Testing checks whether the new version of the software works properly with the file format created by an older version of the software. It also works well with data tables, data files, and data structures created by the older version of that software. If any of the software is updated, then it should work well on top of the previous version of that software.

**Black Box Testing**

Internal system design is not considered in this type of testing. Tests are based on the requirements and functionality.

**Boundary Value Testing**

This type of testing checks the behavior of the application at the boundary level.

[Boundary Value Testing](https://www.softwaretestinghelp.com/what-is-boundary-value-analysis-and-equivalence-partitioning/) is performed to check if defects exist at boundary values. Boundary Value Testing is used for testing a different range of numbers. There is an upper and lower boundary for each range and testing is performed on these boundary values.

If testing requires a test range of numbers from 1 to 500, then Boundary Value Testing is performed on values at 0, 1, 2, 499, 500, and 501.

**Branch Testing**

This is also known as Branch coverage or decision coverage testing. It is a type of white box testing performed at the unit test level. It is done to make sure that each possible path from the decision point is executed at least once for 100% of test coverage.

**Example:**

Read number A, B  
If (A>B) then  
Print(“A is greater”)  
Else  
Print(“B is greater”)

Here, there are two branches, one for if and the other for else. For 100% coverage, we need 2 test cases with different values of A and B.

Test case 1: A=10, B=5 It will cover the if branch.

Test case 2: A=7, B=15 It will cover the else branch.

**Comparison Testing**

Comparison of a product’s strengths and weaknesses with its previous versions or other similar products is termed Comparison Testing.

**Equivalence Partitioning**

It is a testing technique and a type of Black Box Testing. During this [Equivalence Partitioning](https://www.softwaretestinghelp.com/what-is-boundary-value-analysis-and-equivalence-partitioning/), a set of groups are selected and a few values or numbers are picked up for testing. It is understood that all values from that group generate the same output.

The aim of this testing is to remove redundant test cases within a specific group that generate the same output but not any defect.

Suppose the application accepts values between -10 and +10, then using equivalence partitioning, the values picked for testing are zero, one positive value, and one negative value. So the Equivalence Partitioning for this testing is  -10 to -1, 0, and 1 to 10.

**Example Testing**

Example testing is real-time testing. It includes real-time scenarios and scenarios are based on the experience of the testers.

This type of testing is also known as experience-based testing because it uses the tester’s knowledge of how the application has worked in the past, how to break the application, what kind of errors are common in this type of application.

**Graphical User Interface (GUI) Testing**

The objective of this GUI Testing is to validate the GUI as per the business requirement. The expected GUI of the application is mentioned in the Detailed Design Document and GUI mockup screens.

GUI Testing includes the size of the buttons and input fields present on the screen, alignment of all text, tables, and content in the tables.

It also validates the menu of the application. After selecting different menu and menu items, it validates that the page does not fluctuate, and the alignment remains the same after hovering the mouse on the menu or sub-menu.

**Incremental Integration Testing**

[Incremental Integration Testing](https://www.softwaretestinghelp.com/incremental-testing/) is a Bottom-up approach for testing, i.e continuous testing of an application when new functionality is added.

Application functionality and modules should be independent enough to test separately. This is done by programmers or by testers.

**Install/Uninstall Testing**

Installation testing is performed to check that the software application is installed properly and working as per expectation. Installation testing is a phase of testing before users interact with the actual application for the first time. Installation testing is also called “Implementation Testing”.

Uninstallation Testing is performed to confirm if all the components or elements of the software are removed from the system or not.

[Installation and Uninstallation Testing](https://www.softwaretestinghelp.com/software-installationuninstallation-testing/) is done on full, partial, or upgraded install/uninstall processes on different operating systems under different hardware or software environments.

**Mutation Testing**

[Mutation Testing](https://www.softwaretestinghelp.com/what-is-mutation-testing/) is a type of white box testing in which the source code of one program is changed and verifies whether the existing test cases can identify these defects in the system.

The change in the program source code is very minimal, so it does not impact the entire application, only the specific area having the impact and the related test cases should be able to identify those errors in the system.

**Negative Testing**

The mindset of the tester is to “Break the System/Application” and it is achieved through Negative Testing.

[Negative Testing technique](https://www.softwaretestinghelp.com/what-is-negative-testing/) is performed using incorrect data, invalid data, or input. It validates if the system throws an error of invalid input and behaves as expected.

It should not take much time to load any page or system and should be sustained during peak load. Different performance and load tools are used to do this testing.

**Recovery Testing**

It is a type of testing that validates how well the application or system recovers from crashes or disasters.

Recovery Testing determines if the system can continue its operation after a disaster. Assume that the application is receiving data through a network cable and suddenly that network cable has been unplugged.

Sometime later, plug in the network cable; then the system should start receiving data from where it lost the connection due to the network cable being unplugged.

**Regression Testing**

Regression testing is testing of unchanged features of the application to make sure that any bug fixes, adding new features, deleting, or updating existing features, are not impacting the working application.

To find out regression scope is an important part in  [Regression Testing](https://www.softwaretestinghelp.com/regression-testing-tools-and-methods/). To find out regression scope, Tester needs to find out the area of application where changes happened and the Impact of those changes on the entire application. It is difficult to cover the whole regression test suite in every release, so [Automation Testing Tools](https://www.softwaretestinghelp.com/automation-testing-tutorial-1/) are used in regression testing.

**Risk-Based Testing (RBT)**

For [Risk-Based Testing](https://www.softwaretestinghelp.com/risk-management-during-test-planning-risk-based-testing/), the functionalities or requirements are tested based on their priority. Risk-Based Testing includes testing of highly critical functionality, which has the highest impact on business and in which the probability of failure is very high.

Priority decisions are based on business needs, so once priority is set for all functionalities, then high priority functionality or test cases are executed first, followed by medium and then low priority functionalities.

Low priority functionality may be tested or not tested based on the available time. Risk-Based Testing is carried out if there is insufficient time available to test the entire software and the software needs to be implemented on time without any delay.

This approach is followed only by the discussion and approval of the client and senior management of the organization.

**Static Testing**

Static Testing is a type of testing which is done without the execution of any code. Reviews, walkthroughs, and inspections are different methods of performing static testing. Activities like reviewing of requirement documents, customer requirement specification, high level, and low-level design, code syntax, naming standards, etc. come under static testing.

Static Testing also applies to test cases, test plans, test scenarios. Static testing is done to prevent the defect rather than catching the defect at a later stage. That is why static testing is cost-effective.

**For example,** Tester is testing a pet insurance website. The logic for premium calculation is described in requirement documentation. As a part of static testing, testers can review the developer code for premium calculation and compare it with the requirement document to prevent the defect related to premium calculation.

**Vulnerability Testing**

The testing, which involves identifying weaknesses in the software, hardware, and network, is known as Vulnerability Testing. In malicious programs, the hacker can take control of the system, if it is vulnerable to such kinds of attacks, viruses, and worms.

We need to check if those systems undergo Vulnerability Testing before production. It may identify critical defects and flaws in security.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  | | --- | --- | --- | | **S.NO.** | **Behavior Driven Development** | **Test Driven Development** | | 01. | Behavior Driven Development is a development technique which focuses more on a software application’s behavior. | Test Driven Development is a development technique which focuses more on the implementation of a feature of a software application/product. | | 02. | In BDD the participants are Developers, Customer, QAs. | In TDD the participants are developers. | |  |  |  | | 04. | Its main focus is on system requirements. | Its main focus is on unit test. | | 05. | In BDD the starting point is a scenario. | In TDD the starting point is a test case. | | 06. | It is a team methodology. | It is a development practice. | | 07. | Here language used to write behavior/scenarios is simple English language. | Here language is used is similar to the one used for feature development like programming language. | | 08. | In BDD collaboration is required between all the stakeholders. | In TDD collaboration is required only between the developers. | | 09. | It is a good approach for project development which are driven by user actions. | It is a good approach for projects which involve API and third-party tools. | | 10. | Some of the tools used are  Cucumber, Dave, JBehave, Spec Flow,  Concordian, BeanSpec etc. | Some of the tools used are  JBehave, JDave, Cucumber, Spec Flow, BeanSpec, FitNesse etc. | | |

**TDD is a development practice while BDD is a team methodology**. In TDD, the developers write the tests while in BDD the automated specifications are created by users or testers (with developers wiring them to the code under test.) For small, co-located, developer-centric teams, TDD and BDD are effectively the same.

Disadvantages of BDD

Because communications between the user and the developer are essential, if the user if not available, it will be difficult to work past ambiguities and questions generated by the user stories. The second disadvantage is the need to dedicate a team of developers to work with the client.

**TDD disadvantages**

* Forget to run tests frequently.
* Write too many tests at once.
* Write tests that are too large.
* Write tests that are overly trivial.
* Write tests for trivial code.

**1) Different key words inside the jerkin format**

**Gherkin** is not necessarily used to write automated tests. Gherkin is primarily used to write **structured** tests which can later be used as project documentation. The property of being structured gives us the ability to automate them. This automation is done by **Cucumber/SpecFlow**.

**Feature**: LogIn Action Test

Description: This feature will test a LogIn and LogOut functionality

**Scenario:** Successful Login with Valid Credentials

**Given** User is on Home Page

**When** User Navigate to LogIn Page

**And** User enters UserName and Password

**Then** Message displayed Login Successfully

list of keywords that Gherkin supports:

* **Feature**
* **Rule**
* **Background**
* **Scenario**
* **Scenario Outline**
* **Given**
* **When**
* **Then**
* **And**
* **But**
* **' \* '**

## Feature: Keyword

Each Gherkin file begins with a **Feature** keyword. Feature defines the logical test functionality you will test in this feature file. For e.g, if you are testing a payment gateway your Feature will become Payment Gateway or if you are testing the LogIn functionality then the Feature will become Login. The idea of having a feature file is to put down a summary of what you will be testing. This will serve as the documentation for your tests as well as a good point to start for a new team member.

**Feature**: LogIn Action Test

Description: This feature will test a LogIn and LogOut functionality.

**Rule: Keyword**

**Rule** keyword was introduced in the latest version 6 of cucumber to represent the implementation of a business rule. **Rule** is meant to cover one or more scenarios that explain the particular rule. To explain the usage of the **Rule** keyword, follow the below example.

Example:-Consider the feature of OTP generation, in which one of the rules is that OTP is not generated if the account user has opted out of two-factor authentication for transactions below INR 2000.

**Feature**: OTP generation test

**Rule**: OTP is not generated if the account user has opted out of two-factor authentication for transactions below INR 2000

**Scenario**: For an amount lesser than INR 2000, if the account user has opted out of the two-factor authentication, an OTP must not be generated

**Scenario Outline**: If the account user has not opted out of two-factor authentication, an OTP must be generated regardless of the amount.

## Background: Keyword

**Background** keyword is used to define steps that are common to all the tests in the feature file. For example, to purchase a product, you need to do the following steps:

* Navigate to Home Page
* Click on the LogIn link
* Enter UserName and Password
* Click on Submit button

After these steps only you will be able to add a product to your cart/basket and able to perform the payment. Now as we are in a feature file where we will be testing only the Add to Cart functionality, these tests become common for all tests. So instead of writing them again and again for all tests, we can move it under the background keyword. This is how it will look like:

**Feature**: Add to Cart

This feature will test functionality of adding different products to the User basket from different flow

**Background**: User is Logged In

**Scenario**: Search a product and add the first result/product to the User basket

**Given** User searched for Lenovo Laptop

**When** Add the first laptop that appears in the search result to the basket

**Then** User basket should display with 1 item

## Scenario: Keyword

Each Feature will contain a number of tests to test the feature. Each test is called a **Scenario** and is described using the Scenario: keyword.

**Scenario**: Search a product and add the first result/product to the User basket

Or

**Scenario**: Successful LogIn with Valid Credentials

A scenario is equivalent to a test in our regular development process. Each scenario/test can be basically broken down into three parts:

* **Precondition** to the test, which represent with (Given) keyword
* **Test step** execution, which represent with (When) keyword
* **Verification** of the output with expected result, which represent with (Then)

## Scenario Outline: Keyword

In situations where one wants to execute the same **Scenario** with various combinations of values or arguments, one could use the **Scenario Outline**. It facilitates the testing of the same scenario with multiple arguments.

Examp**le:-** This application determines if delivery is possible for a particular city or not. The rule of the feature is that only the capital cities of a country are eligible for regular delivery of spare parts. For all other cities, customers must select premium as the delivery type.

**Feature**: Delivery Application - delivery type selection

**Rule**: Only capital cities of a country are eligible for regular delivery of automobile spare parts. For all other cities, customers must select premium as the delivery type.

**Scenario Outline**: The delivery type of automobile spare parts is dependent on the destination city of delivery.

**Given** User is raising a delivery request for automobile spare parts

**When** User enters the **<"city">**, a **<"delivery type">** is set as a system default

**Then** User submits delivery request

**Examples**:

| city | delivery type|

| Athens | Regular |

| Delhi | Regular |

| Kolkata| Premium |

| Houston| Premium |

## Given: Keyword

**Given** defines a precondition to the test. For e.g. In the shopping website, assume that the Login page link is only present on the Home Page, so the precondition for clicking the Login link is that the user is at the Home Page. If user is not at the Home Page, user would not be able to enter Username & Password. This precondition can be expressed in Gherkin like this:

**Scenario**: Successful Login with Valid Credentials

**Given** User is on Home Page

**When** User Navigate to Login Page

## When: Keyword

**When** keyword defines the test action that will be executed. By test action we mean the user input action.

**Scenario**: Successful Login with Valid Credentials

**Given** User is on Home Page

**When** User Navigate to LogIn Page

Here user is performing some action using When keyword, clicking on the LogIn link. We can see that when defines the action taken by the user. It's the event that will cause the actual change in state of the application.

## Then: Keyword

**Then** keyword defines the Outcome of previous steps. We can understand it best by looking at the test above and adding a Then step there.

**Feature**: LogIn Action Test

Description: This feature will test a LogIn and LogOut functionality

**Scenario**: Successful Login with Valid Credentials       **Given** User is on Home Page

**When** User Navigate to LogIn Page

**And** User enters UserName and Password

**Then** Message displayed LogIn Successfully

Here we can see that **Then** is the outcome of the steps above. The reader of this test would easily be able to relate to Then step and would understand that when the above conditions are fulfilled then the Then step will be executed.

## And: Keyword

**And** keyword is used to add conditions to your steps. Let's look at it by modifying our example a little

**Feature**: LogIn Action Test

Description: This feature will test a LogIn and LogOut functionality

**Scenario**: Successful Login with Valid Credentials

**Given** User is on Home Page

**When** User Navigate to LogIn Page

**And** User enters UserName and Password

**Then** Message displayed Login Successfully

Or

**Feature**: LogIn Action Test

Description: This feature will test a LogIn and LogOut functionality

**Scenario**: Successful Login with Valid Credentials

**Given** User is on Home Page

**And** LogIn Link displayed

**When** User Navigate to LogIn Page

**And** User enters UserName and Password

**Then** Message displayed Login Successfully       **And** LogOut Link displayed

Here you would see that And is being used to add more details to the Given step, it's simply adding more conditions. We have just added three conditions. Use it when you have specified more than one condition. And is used to add more conditions to Given, When and Then statements.

## But: Keyword

**But** keyword is used to add negative type comments. It is not a hard & fast rule to use but only for negative conditions. It makes sense to use But when you will try to add a condition which is opposite to the premise your test is trying to set. Take a look at the example below:

**Feature**: LogIn Action Test

Description: This feature will test a LogIn and LogOut functionality

**Scenario**: Unsuccessful Login with InValid Credentials

**Given** User is on Home Page

**When** User Navigate to LogIn Page

**And** User enters UserName and Password

**But** The user credentials are wrong

**Then** Message displayed Wrong UserName & Password

Here you can see how adding **But** has helped define a negative test, in this test we will try to test failure conditions. Where a wrong credentials are a failure condition.

## Keyword

This keyword is very special. This keyword defies the whole purpose of having Given, When, Then and all the other keywords. Basically Cucumber doesn't care about what Keyword you use to define test steps, all it cares about what code it needs to execute for each step. That code is called a **step definition** and we will discuss about it in the next section. At this time just remember that all the keywords can be replaced by the \* **keyword** and your test will just work fine. Let's see with example, we had this test earlier:

**Feature**: LogIn Action Test

Description: This feature will test a LogIn and LogOut functionality

**Scenario**: Successful Login with Valid Credentials

**Given** User is on Home Page

**When** User Navigate to LogIn Page

**And** User enters UserName and Password       Then Message displayed Login Successfully

**Using \* Keyword**

**Feature**: LogIn Action Test

Description: This feature will test a LogIn and LogOut functionality

**Scenario**: Successful Login with Valid Credentials

\* User is on Home Page

\* User Navigate to LogIn Page

\* User enters UserName and Password

\* Message displayed Login Successful

## What is Cucumber Feature File?

A ***Feature File*** is an entry point to the Cucumber tests. This is a file where you will describe your tests in Descriptive language (Like English). It is an essential part of Cucumber, as it serves as an automation test script as well as live documents.

### ***Keywords***

Now moving forward we have just defined a test. You will notice colored parts of the tests (Feature, Scenario, Given, When, And and Then). These are keywords defined by ***Gherkin***. Gherkin has more keywords and we will discuss those in the following tutorials. But to start off we can quickly explain some of the keywords in one line. Note this is not a complete listing of Keywords:

***Feature: Defines what feature you will be testing in the tests below***

***Given: Tells the pre-condition of the test***

***And: Defines additional conditions of the test***

***Then: States the post condition. You can say that it is the expected result of the test.***

### ***Gherkin***

A language above is called ***Gherkin*** and it implements the principles of ***Business readable domain specific language***(BRDSL). Domain-specific language gives you the ability to describe your application behavior without getting into details of implementation. What does that mean? If we go back to our tutorial in [***TDD***](https://toolsqa.com/cucumber/test-driven-development-tdd/) we saw that we wrote test code before writing any application code. In a way, we described what is the expected behavior of our application in terms of tests. On TDD those tests were pure Java tests, in your case, those might be a C++ or C# tests. But the basic idea is that those are core technical tests.