# **UNIT – IV: Software Testing**

[**UNIT – IV: Software Testing 1**](#_wf6ppvf1sdpr)

[Syllabus 1](#_43gn25tqpy0l)

[**What is Software Testing? 2**](#_6kobl09yjs54)

[**Importance of Software Testing 2**](#_fo6w3ntmtglw)

[**Need for Software Testing 3**](#_nunmxb7kott1)

[**What is a Test Case? 3**](#_vzs0cquroqa8)

[**Parameters of a Test Case 3**](#_vguadwpb40jw)

[**Test Case vs Test Scenario 4**](#_t7c938m1waka)

[When do we Write Test Cases? 5](#_zhov46piht83)

[Why Write Test Cases? 6](#_9eay1r2voy4m)

[Test Case Template 6](#_917sd9riseco)

[**What is Functional Testing? 8**](#_5cf7i9353wjb)

[Purpose of Functional Testing 8](#)

[What to Test in Functional Testing? 9](#)

[Functional Testing Process 9](#)

[Type of Functional Testing Techniques 10](#)

[Functional Testing vs Non-Functional Testing 11](#)

[Functional Testing Tools 12](#)

[Limitations of Functional Testing 13](#)

[**Structural Testing 13**](#_halh3eguaw7)

[Types of Structural Testing  
There are 4 types of Structural Testing: 14](#_wjaqjtl5h10g)

[Advantages of Structural Testing: 14](#_mqrhm1401eyl)

[Disadvantages of Structural Testing: 14](#_a7cgf94rqw26)

[Structural Testing Tools: 15](#_uyqd0nhl1t2o)

## Syllabus

Software Testing: Testing Process, Design of Test Cases, Types of Testing, Functional Testing, Structural Testing, Test Activities, Unit Testing, Integration Testing and System Testing, Debugging Activities. Software Maintenance: Management of Maintenance, Maintenance Process, Reverse Engineering, Software Re-engineering, Configuration Management, Documentation

# What is Software Testing?

Software Testing is a method to assess the functionality of the software program. The process checks whether the actual software matches the expected requirements and ensures the software is bug-free. The purpose of software testing is to identify the errors, faults, or missing requirements in contrast to actual requirements. It mainly aims at measuring the specification, functionality, and performance of a software program or application.

Software testing can be divided into two steps:

1. **Verification**: It refers to the set of tasks that ensure that the software correctly implements a specific function. It means “Are we building the product right?”.
2. **Validation**: It refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements. It means “Are we building the right product?”.

## Importance of Software Testing

* **Defects can be identified early:** Software testing is important because if there are any bugs they can be identified early and can be fixed before the delivery of the software.
* **Improves quality of software:** Software Testing uncovers the defects in the software, and fixing them improves the quality of the software.
* **Increased customer satisfaction:** Software testing ensures reliability, security, and high performance which results in saving time, costs, and customer satisfaction.
* **Helps with scalability:** Software testing type non-functional testing helps to identify the scalability issues and the point where an application might stop working.
* **Saves time and money:** After the application is launched it will be very difficult to trace and resolve the issues, as performing this activity will incur more costs and time. Thus, it is better to conduct software testing at regular intervals during software development.

## Need for Software Testing

Software bugs can cause potential monetary and human loss. There are many examples in history that clearly depict that without the testing phase in software development a lot of damage was incurred. Below are some examples:

* **1985:** Canada’s Therac-25 radiation therapy malfunctioned due to a software bug and resulted in lethal radiation doses to patients leaving 3 injured and 3 people dead.
* **1994:** China Airlines Airbus A300 crashed due to a software bug killing 264 people.
* **1996:** A software bug caused U.S. bank accounts of 823 customers to be credited with 920 million US dollars.
* **1999:** A software bug caused the failure of a $1.2 billion military satellite launch.
* **2015:** A software bug in fighter plane F-35 resulted in making it unable to detect targets correctly.
* **2015:** Bloomberg terminal in London crashed due to a software bug affecting 300,000 traders on the financial market and forcing the government to postpone the 3bn pound debt sale.
* Starbucks was forced to close more than 60% of its outlet in the U.S. and Canada due to a software failure in its POS system.
* Nissan cars were forced to recall 1 million cars from the market due to a software failure in the car’s airbag sensory detectors.

# What is a Test Case?

A test case is a defined format for software testing required to check if a particular application/software is working or not. A test case consists of a certain set of conditions that need to be checked to test an application or software i.e. in more simple terms when conditions are checked it checks if the resultant output meets with the expected output or not. A test case consists of various parameters such as Id, condition, steps, input, expected result, result, status, and remarks.

## Parameters of a Test Case

1. **Module Name:** Subject or title that defines the functionality of the test.
2. **Test Case Id:** A unique identifier assigned to every single condition in a test case.
3. **Tester Name:** The name of the person who would be carrying out the test.
4. **Test scenario:** The test scenario provides a brief description to the tester, as in providing a small overview to know about what needs to be performed and the small features, and components of the test.
5. **Test Case Description:** The condition required to be checked for a given software. for eg. Check if only numbers validation is working or not for an age input box.
6. **Test Steps:** Steps to be performed for the checking of the condition.
7. **Prerequisite:** The conditions required to be fulfilled before the start of the test process.
8. **Test Priority:** As the name suggests gives the priority to the test cases as in which had to be performed first, or are more important and which could be performed later.
9. **Test Data:** The inputs to be taken while checking for the conditions.
10. **Test Expected Result:** The output which should be expected at the end of the test.
11. **Test parameters:** Parameters assigned to a particular test case.
12. **Actual Result:** The output that is displayed at the end.
13. **Environment Information:** The environment in which the test is being performed, such as the operating system, security information, the software name, software version, etc.
14. **Status:** The status of tests such as pass, fail, NA, etc.
15. **Comments:** Remarks on the test regarding the test for the betterment of the software.

## Test Case vs Test Scenario

Below are some of the points of difference between a test case and a test scenario:

| **Parameter** | **Test Case** | **Test Scenario** |
| --- | --- | --- |
| **Definition** | A test case is a defined format for software testing required to check if a particular application/software/module is working or not. Here we check for different conditions regarding the same. | The test Scenario provides a small description of what needs to be performed based on the use case. |
| **Level of detailing** | Test cases are more detailed with a number of parameters. | Test Scenario provides a small description, mostly one-line statements. |
| **Action Level** | Test cases are low-level actions. | Test scenarios are high-level actions. |
| **Derived from** | Test cases are mostly derived from test scenarios. | Test scenarios are derived from documents like BRS, SRS, etc. |
| **Objective** | It focuses on “What to test” and “How to test”. | It focuses more on ‘What to test”. |
| **Resources required** | Test cases require more resources for documentation and execution. | Fewer resources are required to write test scenarios. |
| **Inputs** | It includes all positive and negative inputs, expected results, navigation steps, etc. | They are one-liner statements. |
| **Time requirement** | It requires more time compared to test scenarios. | Test scenarios require less time. |
| **Maintenance** | They are hard to maintain. | They require less time to maintain. |

## When do we Write Test Cases?

Test cases are written in different situations:

* **Before development:**Test cases could be written before the actual coding as that would help to identify the requirement of the product/software and carry out the test later when the product/software once gets developed.
* **After development:** Test cases are also written directly after coming up with a product/software or after developing the feature but before the launching of a product/software as needed to test the working of that particular feature.
* **During development:** Test cases are sometimes written during the development time, parallelly. so whenever a part of the module/software gets developed it gets tested as well.

So, test cases are written in such cases, as test cases help in further development and make sure that we are meeting all the needed requirements.

## Why Write Test Cases?

Test cases are one of the most important aspects of software engineering, as they define the way in which the testing would be carried out. Test cases are carried out for a very simple reason, to check if the software actually works or not. There are many advantages of writing test cases:

* **To check whether the software meets customer expectations:**Test cases help to check if a particular module/software is meeting the specified requirement or not.
* **To check software consistency with conditions:** Test cases determine if a particular module/software work with a given set of conditions.
* **Narrow down software updates:** Test cases help to narrow down the software needs and required updates.
* **Better test coverage:** Test cases helps to make sure that all possible scenarios are covered and documented.
* **For consistency in test execution:** Test cases help to maintain consistency in test execution. A well-documented test case helps the tester to just have a look at the test case and start testing the application.
* **Helpful during maintenance:**Test cases are detailed which makes them helpful during the maintenance phase.

## Test Case Template

* Let’s look at a basic test case template for the login functionality.
* The Test case template contains the header section which has a set of parameters that provides information about the test case such as the tester’s name, test case description, Prerequisite, etc.
* The body section contains the actual test case content, such as test Id, test steps, test input, expected result, etc.
* Below is the table that shows the basic template of a test case:

| **Fields** | **Description** |
| --- | --- |
| Test Case ID | Each test case should have a unique ID. |
| Test Case Description | Each test case should have a proper description to let testers know what the test case is about. |
| Pre-Conditions | Conditions that are required to be satisfied before executing the test case. |
| Test Steps | Mention all test steps in detail and in the order to be executed from the end-user’s perspective. |
| Test Data | Test data could be used as input for the test cases. |
| Expected Result | The result expected after executing the test cases. |
| Post Condition | Conditions need to be fulfilled when the test cases was successfully executed. |
| Actual Result | The result which system shows once the test case was executed. |
| Status | Set status as Pass or Fail on the expected result against the actual result. |
| Project Name | Name of the project to which test case belongs to. |
| Module Name | Name of the module to which test case belongs to. |
| Reference Document | Mention the path of the reference document. |
| Created By | Name of the tester who created the test cases. |
| Date of Creation | Date of creation of test cases. |
| Reviewed By | Name of the tester who reviews the test case. |
| Date of Review | When the test cases were reviewed. |
| Executed By | Name of the tester who executed the test case. |
| Date of Execution | Date when the test cases were executed. |
| **Comments** | Include comments which help the team to understand the test cases. |

In the given template below it’s clearly identifiable that the section from module name to test scenario is the header section while the table that lies below the test scenario (from test case id to comments) is the body of the test case template.

# What is Functional Testing?

Functional testing is basically defined as a type of testing that verifies that each function of the software application works in conformance with the requirement and specification. This testing is not concerned with the source code of the application. Each functionality of the software application is tested by providing appropriate test input, expecting the output, and comparing the actual output with the expected output. This testing focuses on checking the user interface, APIs, database, security, client or server application, and functionality of the Application Under Test. Functional testing can be manual or automated.

## Purpose of Functional Testing

Functional testing mainly involves black box testing and can be done manually or using automation. The purpose of functional testing is to:

* **Test each function of the application:** Functional testing tests each function of the application by providing the appropriate input and verifying the output against the functional requirements of the application.
* **Test primary entry function:** In functional testing, the tester tests each entry function of the application to check all the entry and exit points.
* **Test flow of the GUI screen:** In functional testing, the flow of the GUI screen is checked so that the user can navigate throughout the application.

## What to Test in Functional Testing?

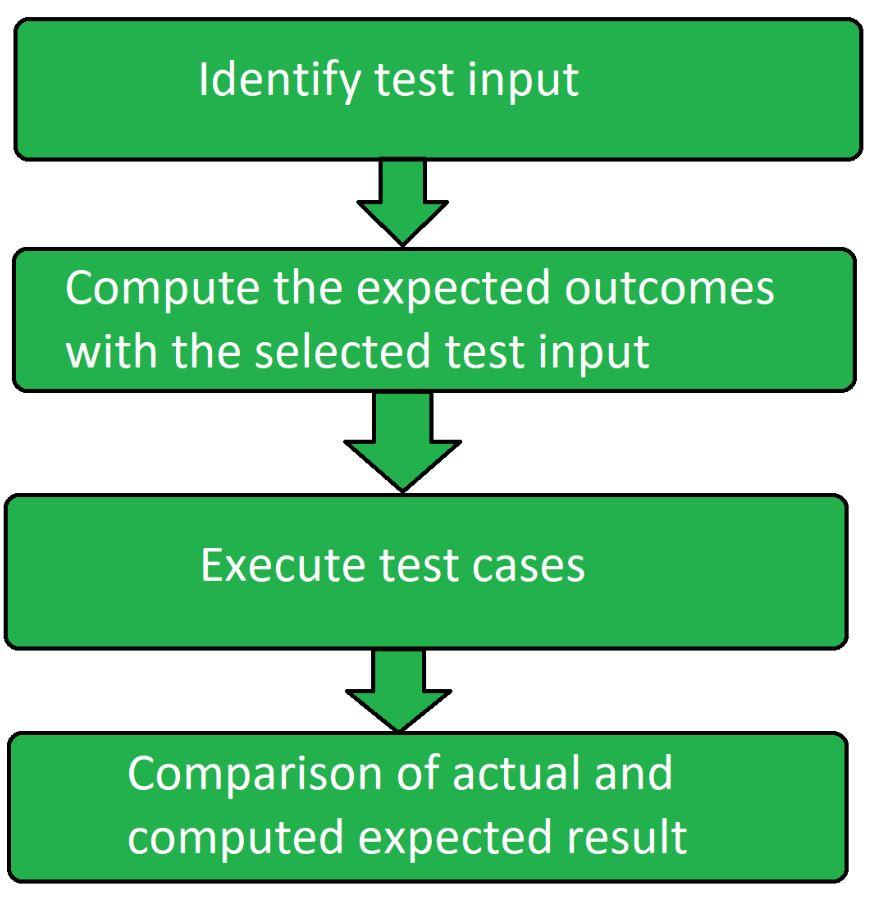
The goal of functional testing is to check the functionalities of the application under test. It concentrates on:

* **Basic Usability:**Functional testing involves basic usability testing to check whether the user can freely navigate through the screens without any difficulty.
* **Mainline functions:** This involves testing the main feature and functions of the application.
* **Accessibility:** This involves testing the accessibility of the system for the user.
* **Error Conditions:** Functional testing involves checking whether the appropriate error messages are being displayed or not in case of error conditions.

## Functional Testing Process

Functional testing involves the following steps:

1. **Identify test input:** This step involves identifying the functionality that needs to be tested. This can vary from testing the usability functions, and main functions to error conditions.
2. **Compute expected outcomes:** Create input data based on the specifications of the function and determine the output based on these specifications.
3. **Execute test cases:** This step involves executing the designed test cases and recording the output.
4. **Compare the actual and expected output:**  In this step, the actual output obtained after executing the test cases is compared with the expected output to determine the amount of deviation in the results. This step reveals if the system is working as expected or not.

## Type of Functional Testing Techniques

1. **Unit Testing:**Unit testing is the type of functional testing technique where the individual units or modules of the application are tested. It ensures that each module is working correctly.
2. **Integration Testing:** In Integration testing, combined individual units are tested as a group and expose the faults in the interaction between the integrated units.
3. **Smoke Testing:** Smoke testing is a type of functional testing technique where the basic functionality or feature of the application is tested as it ensures that the most important function works properly.
4. **User Acceptance Testing:** User acceptance testing is done by the client to certify that the system meets the requirements and works as intended. It is the final phase of testing before the product release.
5. **Interface Testing:** Interface testing is a type of software testing technique that checks the proper interaction between two different software systems.
6. **Usability Testing:**Usability testing is done to measure how easy and user-friendly a software application is.
7. **System Testing:**System testing is a type of software testing that is performed on the complete integrated system to evaluate the compliance of the system with the corresponding requirements.
8. **Regression Testing:** Regression testing is done to make sure that the code changes should not affect the existing functionality and the features of the application. It concentrates on whether all parts are working or not.
9. **Sanity Testing:** Sanity testing is a subset of regression testing and is done to make sure that the code changes introduced are working as expected.
10. **White box Testing:** White box testing is a type of software testing that allows the tester to verify the internal workings of the software system. This includes analyzing the code, infrastructure, and integrations with the external system.
11. **Black box Testing:**Black box testing is a type of software testing where the functionality of the software system is tested without looking at the internal working or structures of the software system.
12. **Database Testing:**Database testing is a type of software testing that checks the schema, tables, etc of the database under test.
13. **Adhoc Testing:** Adhoc testing also known as monkey testing or random testing is a type of software testing that does not follow any documentation or test plan to perform testing.
14. **Recovery Testing:**Recovery testing is a type of software testing that verifies the software’s ability to recover from the failures like hardware failures, software failures, crashes, etc.
15. **Static Testing:**Static testing is a type of software testing which is performed to check the defects in software without actually executing the code of the software application.
16. **Greybox Testing:** Grey box testing is a type of software testing that includes black box and white box testing.
17. **Component Testing:** Component testing also known as program testing or module testing is a type of software testing that is done after the unit testing. In this, the test objects can be tested independently as a component without integrating with other components.

## Functional Testing vs Non-Functional Testing

Below are the differences between functional testing and non-functional testing:

| **Parameters** | **Functional Testing** | **Non-functional Testing** |
| --- | --- | --- |
| **Definition** | Functional testing verifies the operations and actions of an application. | Non-functional verifies the behavior of an application. |
| **Testing based on** | It is based on the requirements of the customer. | It is based on the expectations of the customer. |
| **Objective** | The objective is to validate software actions. | The objective is to performance of the software system |
| **Requirements** | Functional testing is carried out using the functional specification. | Non-functional testing is carried out using the performance specifications. |
| **Functionality** | It describes what the product does. | It describes how the product works. |
| **Example** | * Unit testing. * Integration testing. * Sanity testing * Smoke testing. * Regression testing. | * Performance testing. * Load testing. * Stress testing. * Volume testing. * Usability testing. |

## Functional Testing Tools

Below are the tools for functional testing:

1.**Selenium:** Selenium is an open-source umbrella project for a range of tools and libraries developed with the aim to support browser automation.

* It is used to automate web browsers.
* It provides a single interface that lets the tester write test scripts in languages like Ruby, Java, NodeJS, etc.
* It provides a playback tool for authoring functional tests across most modern web browsers.

2. **QTP:** QTP tool now can UFT is a tool designed to perform automated functional testing without the need to monitor the system in intervals.

* It can be used to test web, desktop applications, and client-server.
* It is based on the VB scripting language.
* It is one of the widely used automation tools in the testing industry.

3. **JUnit:** JUnit is a unit-testing open-source framework for the Java programming language. It is used by Java developers to write and execute automated test cases.

* It can be used along with the Selenium WebDriver to automate tests for web applications.
* It provides several annotations to identify test methods.
* It has test runners to run tests.

4. **SoapUI:**It is one of the leading tools for SOAP and web service testing. It allows for easy and rapid creation and execution of functional, regression, and load tests.

* It has an easy to use graphical interface.
* It provides a code-free test environment where one can create and execute complex test cases with drag-and-drop options.
* It lets us dynamically analyze how well SOAP and REST service contract is covered by the functional tests.

5. **Cucumber:** Cucumber is an open-source testing tool written in Ruby language.

* This tool focuses on end-user experience.
* Quick and easy setup and execution.
* This tool allows for easy reuse of code in tests due to the style of writing the tests.

## Limitations of Functional Testing

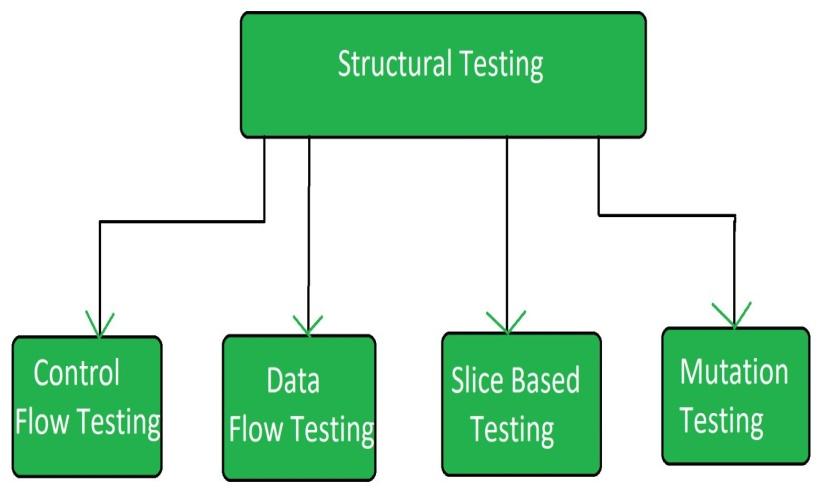
* **Missed critical errors:** There are chances while executing functional tests that critical and logical errors are missed.
* **Redundant testing:** There are high chances of performing redundant testing.
* **Incomplete requirements:** If the requirement is not complete then performing this testing becomes difficult.

# Structural Testing

Structural testing is a type of software testing which uses the internal design of the software for testing or in other words the software testing which is performed by the team which knows the development phase of the software, is known as structural testing.

Structural testing is basically related to the internal design and implementation of the software i.e. it involves the development team members in the testing team. It basically tests different aspects of the software according to its types. Structural testing is just the opposite of behavioral testing.

## **Types of Structural Testing** There are 4 types of Structural Testing:



1. **Control Flow Testing:**A structural method, utilizes the program's flow model, assessing code logic and structure for desired outcomes.
2. **Data Flow Testing:**Data Flow Testing inspects data anomalies by examining variable-value associations, detecting uninitialized variable usage or unused initialized variables.
3. **Slice Based Testing:**Weiser and Gallagher's slicing, proposed for software maintenance, partitions the program into slices to facilitate debugging, maintenance, and understanding.
4. **Mutation Testing:**Mutation testing alters software slightly to assess existing tests' quality and to create new effective tests, uncovering weaknesses in test data.

## Advantages of Structural Testing:

* It provides thorough testing of the software.
* It helps in finding out defects at an early stage.
* It helps in elimination of dead code.
* It is not time consuming as it is mostly automated.

## Disadvantages of Structural Testing:

* It requires knowledge of the code to perform tests.
* It requires training in the tool used for testing.
* Sometimes it is expensive.

## Structural Testing Tools:

* JBehave
* Cucumber
* Junit
* Cfix