**What is System Testing? Types & Definition with Example**

**What is System Testing?**

System Testing is the testing of a complete and fully integrated software product. Usually software is only one element of a larger computer based system. Ultimately, software is interfaced with other software/hardware systems.System Testing is actually a series of different tests whose sole purpose is to exercise the full computer based system.

Two Category of Software Testing

* Black Box Testing
* White Box Testing

System test falls under the **black box testing** category of software testing.

**White box testing** is the testing of the internal workings or code of a software application. In contrast, black box or System Testing is the opposite. System test involves the external workings of the software from the user's perspective.

Please be patient. The Video will load in some time. If you still face issue viewing video click [here](https://www.guru99.com/faq.html#1)

**What do you verify in System Testing ?**

System Testing involves testing the software code for following

* **Testing the fully integrated applications**  including external peripherals in order to check how components interact with one another and with the system as a whole. This is also called End to End testing scenario..
* Verify thorough testing of every input in the application to check for desired outputs.
* Testing of the user's experience with the application. .

That is a very basic description of what is involved in system testing. You need to build  detailed test cases and test suites that test each aspect of the application as seen from the outside without looking at the actual source code.

**Software Testing Hierarchy**

As with almost any technical process, software testing has a prescribed order in which things should be done. The following is a list of software testing categories arranged in chronological order. These are the steps taken to fully test new software in preparation for marketing it:

* **Unit testing -** testing performed on each module or block of code during development. [Unit Testing](https://www.guru99.com/unit-testing-guide.html) is normally done by the programmer who writes the code.

* **Integration testing -** testing done before, during and after integration of a new module into the main software package. This involves testing of each individual code module. One piece of software can contain several modules which are often created by several different programmers. It is crucial to test each module's effect on the entire program model.

* **System testing -** testing done by a professional testing agent on the completed software product before it is introduced to the market.

* **Acceptance testing -** beta testing of the product done by the actual end users.

**Different Types of System Testing**

There are more than 50 types of System Testing. For an exhaustive list of software testing types click [here](https://www.guru99.com/types-of-software-testing.html). Below we have listed types of system testing a large software development company would typically use

1. **Usability Testing -** [Usability Testing](https://www.guru99.com/usability-testing-tutorial.html) mainly focuses on the user's ease to use the application, flexibility in handling controls and ability of the system to meet its objectives

1. **Load Testing -** [Load Testing](https://www.guru99.com/load-testing-tutorial.html) is necessary to know that a software solution will perform under real-life loads.

1. **Regression Testing-** - [Regression Testing](https://www.guru99.com/regression-testing.html) involves testing done to make sure none of the changes made over the course of the development process have caused new bugs. It also makes sure no old bugs appear from the addition of new software modules over time.

1. **Recovery Testing -** Recovery testing is done to demonstrate a software solution is reliable, trustworthy and can successfully recoup from possible crashes.

1. **Migration Testing -** Migration testing is done to ensure that the software can be moved from older system infrastructures to current system infrastructures without any issues.

1. **Functional Testing -** Also known as functional completeness testing,[Functional Testing](https://www.guru99.com/functional-testing.html) involves trying to think of any possible missing functions. Testers might make a list of additional functionalities that a product could have to improve it during functional testing.

1. **Hardware/Software Testing -** IBM refers to Hardware/Software testing as "HW/SW Testing". This is when the tester focuses his/her attention on the interactions between the hardware and software during system testing.

**What Types of System Testing Should Testers Use?**

There are over 50 different types of system testing. The specific types used by a tester depend on several variables. Those variables include:

* **Who the tester works for** - This is a major factor in determining the types of system testing a tester will use. Methods used by large companies are different than that used by medium and small companies.
* **Time available for testing** - Ultimately, all 50 testing types could be used. Time is often what limits us to using only the types that are most relevant for the software project.
* **Resources available to the tester** - Of course some testers will not have necessary resources to conduct a testing type. For example if you are a tester working for a large software development firm, you are likely to have expensive automated testing software not available to others.
* **Software Tester's Education** - There is a certain learning curve for each type of software testing available. To use some of the software involved, a tester has to learn how to use it.
* **Testing Budget** - Money becomes a factor not just for smaller companies and individual software developers but large companies as well.

**INTEGRATION Testing Tutorial: Big Bang, Top Down & Bottom Up**

**What is Integration Testing?**

In integration Testing, individual software modules are integrated logically and tested as a group.

A typical software project consists of multiple software modules, coded by different programmers.

 integration Testing focuses on checking data communication amongst these modules.

Hence it is also termed as **'I & T'** (Integration and Testing), **'String Testing'** and sometimes 'Thread Testing'.

**Why do Integration Testing?:**

Although each software module is unit tested, defects still exist for various reasons like

* A Module in general is designed by an individual software developer whose understanding and programming logic may differ from other programmers. integration Testing becomes necessary to verify the software modules work in unity
* At the time of module development, there are wide chances of change in requirements by the clients. These new requirements may not be unit tested and hence system integration Testing becomes necessary.
* Interfaces of the software modules with the database could be erroneous
* External Hardware interfaces, if any, could be erroneous
* Inadequate exception handling could cause issues.

Please be patient. The Video will load in some time. If you still face issue viewing video click [here](https://www.guru99.com/faq.html#1)

**Integration Test Case:**

Integration[Test Case](https://www.guru99.com/test-case.html)differs from other test cases in the sense it**focuses mainly on the interfaces & flow of data/information between the modules**. Here priority is to be given for the **integrating links** rather than the unit functions which are already tested.

Sample Integration Test Cases for the following scenario:Application has 3 modules say 'Login Page', 'Mail box' and 'Delete mails' and each of them are integrated logically.

Here do not concentrate much on the Login Page testing as it's already been done in [Unit Testing](https://www.guru99.com/unit-testing-guide.html). But check how it's linked to the Mail Box Page.

Similarly Mail Box: Check its integration to the Delete Mails Module.

| **Test Case ID** | **Test Case Objective** | **Test Case Description** | **Expected Result** |
| --- | --- | --- | --- |
| **1** | Check the interface link between the Login and Mailbox module | Enter login credentials and click on the Login button | To be directed to the Mail Box |
| **2** | Check the interface link between the Mailbox and Delete Mails Module | From Mail box select the an email and click delete button | Selected email should appear in the Deleted/Trash folder |

**Approaches/Methodologies/Strategies of Integration Testing:**

The Software Industry uses variety of strategies to execute Integration testing , viz.

* Big Bang Approach :
* Incremental Approach: which is further divided into following
  + Top Down Approach
  + Bottom Up Approach
  + Sandwich Approach - Combination of Top Down and Bottom Up

Below are the different strategies, the way they are executed and their limitations as well advantages.

**Big Bang Approach:**

Here all component are integrated together at **once**, and then tested.

**Advantages:**

* Convenient for small systems.

**Disadvantages:**

* Fault Localization is difficult.
* Given the sheer number of interfaces that need to be tested in this approach, some interfaces links to be tested could be missed easily.
* Since the integration testing can commence only after "all" the modules are designed, testing team will have less time for execution in the testing phase.
* Since all modules are tested at once, high risk critical modules are not isolated and tested on priority. Peripheral modules which deal with user interfaces are also not isolated and tested on priority.

**Incremental Approach:**

In this approach, testing is done by joining two or more modules that are ***logically related***. Then the other related modules are added and tested for the proper functioning. Process continues until all of the modules are joined and tested successfully.

This process is carried out by using dummy programs called **Stubs and Drivers**. Stubs and Drivers do not implement the entire programming logic of the software module but just simulate data communication with the calling module.

**Stub**: Is called by the Module under Test.

**Driver**: Calls the Module to be tested.

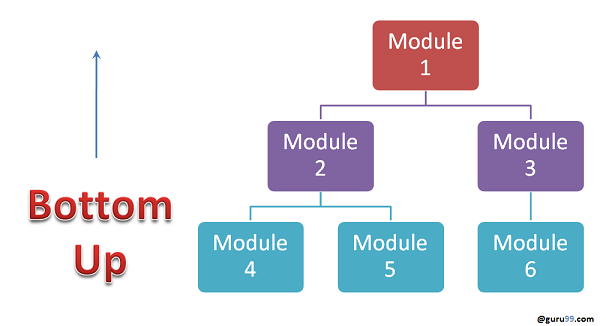
Incremental Approach in turn is carried out by two different Methods:

* **Bottom Up**
* **Top Down**

**Bottom up Integration**

In the bottom up strategy, each module at lower levels is tested with higher modules until all modules are tested. It takes help of Drivers for testing

**Diagrammatic Representation**:

[](https://www.guru99.com/images/bottom-up-integration-testing.png)**Advantages:**

* Fault localization is easier.
* No time  is wasted waiting for all modules to be developed unlike Big-bang approach

**Disadvantages:**

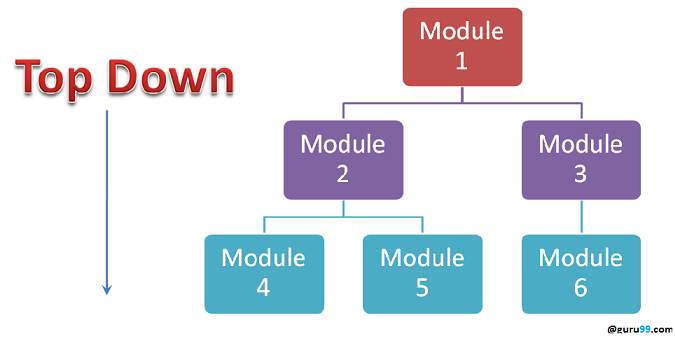
* Critical modules (at the top level of software architecture) which control the flow of application are tested last and may be prone to defects.
* Early prototype is not possible

**Top down Integration:**

In Top to down approach, testing takes place from top to down following the control flow of the software system.

Takes help of stubs for testing.

**Diagrammatic Representation:**

[](https://www.guru99.com/images/top-down-integration-testing.png)**Advantages:**

* Fault Localization is easier.
* Possibility to obtain an early prototype.
* Critical Modules are tested on priority; major design flaws could be found and fixed first.

**Disadvantages:**

* Needs many Stubs.
* Modules at lower level are tested inadequately.

**Integration Testing Procedure**

The integration test procedure irrespective of the test strategies (discussed above):

1. Prepare the Integration Tests Plan
2. Design the Test Scenarios, Cases, and Scripts.
3. Executing the test Cases followed by reporting the defects.
4. Tracking & re-testing the defects.
5. Steps 3 and 4 are repeated until the completion of Integration is successfully.

**Brief Description of Integration Test Plans:**

It includes following attributes:

* Methods/Approaches to test (as discussed above).
* Scopes and Out of Scopes Items of Integration Testing.
* Roles and Responsibilities.
* Pre-requisites for Integration testing.
* Testing environment.
* Risk and Mitigation Plans.

**Entry and Exit Criteria.**

Entry and Exit Criteria to Integration testing phase in any software development model

**Entry Criteria:**

* Unit Tested Components/Modules
* All High prioritized bugs fixed and closed
* All Modules to be code completed and integrated successfully.
* Integration tests Plan, test case, scenarios to be signed off and documented.
* Required[Test Environment](https://www.guru99.com/test-environment-software-testing.html)to be set up for Integration testing

**Exit Criteria:**

* Successful Testing of Integrated Application.
* Executed Test Cases are documented
* All High prioritized bugs fixed and closed
* Technical documents to be submitted followed by release Notes.

**Best Practices/ Guidelines for Integration Testing**

* First determine the Integration[Test Strategy](https://www.guru99.com/how-to-create-test-strategy-document.html)that could be adopted and later prepare the test cases and test data accordingly.
* Study the Architecture design of the Application and identify the Critical Modules. These need to be tested on priority.
* Obtain the interface designs from the Architectural team and create test cases to verify all of the interfaces in detail. Interface to database/external hardware/software application must be tested in detail.
* After the test cases, it's the test data which plays the critical role.
* Always have the mock data prepared, prior to executing. Do not select test data while executing the test cases.

# FUNCTIONAL Testing Tutorial: What is, Process, Types, & Examples

### What is Functional Testing?

Functional testing is a type of testing which verifies that each **function** of the software application operates in conformance with the requirement specification. This testing mainly involves black box testing and it is not concerned about the source code of the application.

Each and every functionality of the system is tested by providing appropriate input, verifying the output and comparing the actual results with the expected results. This testing involves checking of User Interface, APIs, Database, security, client/ server applications and functionality of the Application Under Test. The testing can be done either manually or using automation

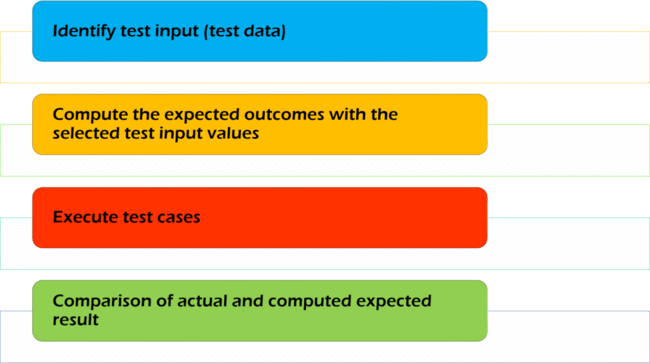
## What do you test in Functional Testing?

The prime objective of Functional testing is   checking the functionalities of the software system. It mainly concentrates on -

* **Mainline functions**:  Testing the main functions of an application
* **Basic Usability**: It involves basic usability testing of the system. It checks whether an user can freely navigate through the screens without any difficulties.
* **Accessibility**:  Checks the accessibility of the system for the user
* **Error Conditions**: Usage of testing techniques to check for error conditions.  It checks whether suitable error messages are displayed.

## Functional Testing Process:

In order to functionally test an application, following steps must be observed.

[](https://www.guru99.com/images/FunctionalTestingProcessv1.png)

Understand the Requirements

Identify test input (test data)

Compute the expected outcomes with the selected test input values

Execute test cases

Comparison of actual and computed expected result

## Functional Vs Non-Functional Testing:

|  |  |
| --- | --- |
| **Functional Testing** | **Non-Functional Testing** |
| Functional testing is performed using the functional specification provided by the client and verifies the system against the functional requirements. | Non-Functional testing  checks the  Performance, reliability, scalability and other non-functional aspects of the software system. |
| Functional testing is executed first | Non functional testing should be performed after functional testing |
| [Manual Testing](https://www.guru99.com/manual-testing.html) or automation tools can be used for functional testing | Using tools will be effective for this testing |
| Business requirements are the inputs to functional testing | Performance parameters like speed , scalability are inputs to non-functional testing. |
| Functional testing describes what the product does | Nonfunctional testing describes how good the product works |
| Easy to do Manual Testing | Tough to do Manual Testing |
| Types of Functional testing are   * [Unit Testing](https://www.guru99.com/unit-testing-guide.html) * Smoke Testing * Sanity Testing * [Integration Testing](https://www.guru99.com/integration-testing.html) * White box testing * Black Box testing * User Acceptance testing * [Regression Testing](https://www.guru99.com/regression-testing.html) | Types of Non functional testing are   * [Performance Testing](https://www.guru99.com/performance-testing.html) * Load Testing * Volume Testing * Stress Testing * Security Testing * Installation Testing * Penetration Testing * Compatibility Testing * Migration Testing |

## ****Functional testing tools:****

[](https://www.guru99.com/images/Functional_Testing_Tools.png)There are several tools available in the marker to perform functional testing. They are explained as follows:

* [Selenium](https://www.guru99.com/selenium-tutorial.html) - Popular Open Source Functional Testing Tool
* [QTP](https://www.guru99.com/quick-test-professional-qtp-tutorial.html) - Very user-friendly Functional Test tool by HP
* [JUnit](https://www.guru99.com/junit-tutorial.html)- Used mainly for[Java](https://www.guru99.com/java-tutorial.html)applications and this can be used in Unit and [System Testing](https://www.guru99.com/system-testing.html)
* [soapUI](https://www.guru99.com/soapui-tutorial.html) - This is an open source functional testing tool, mainly used for Web service testing. It supports multiple protocols such HTTP, SOAP and JDBC.
* Watir - This is functional testing tool for web applications. It supports tests executed at the web browser and uses ruby scripting language

**Conclusion:**

Functional testing is process of testing functionalities of the system and ensures that the system is working as per the functionalities specified in the business document. The goal of this testing is to check whether the system is functionally perfect!!!