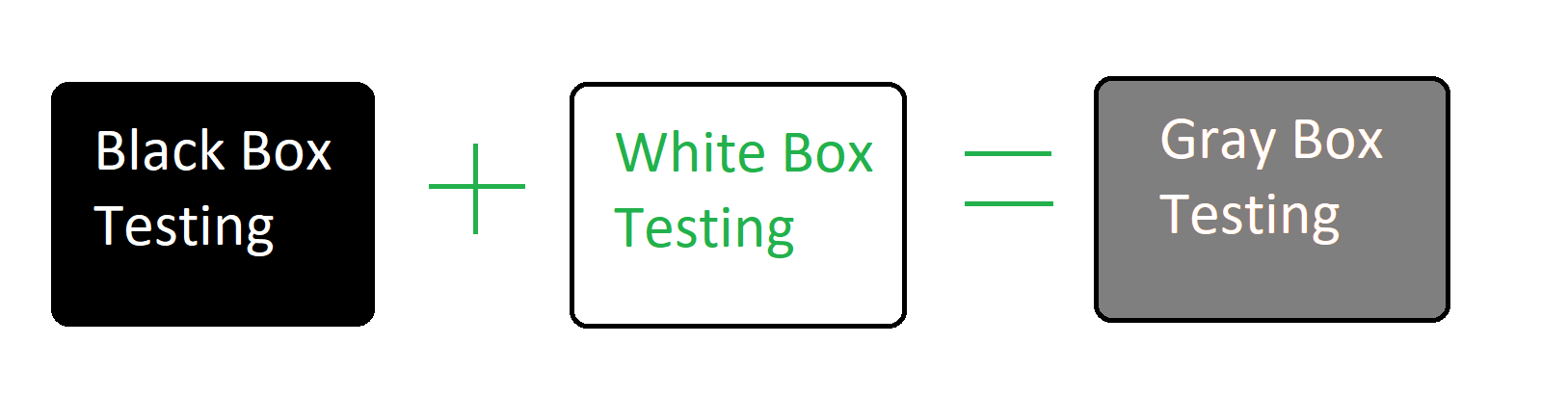
**Different Software Testing**

**Gray Box Testing**

* **Gray Box Testing** is a software testing technique which is a combination of [Black Box Testing](https://www.geeksforgeeks.org/software-engineering-black-box-testing/) technique and [White Box Testing](https://www.geeksforgeeks.org/software-engineering-white-box-testing/) technique. In Black Box Testing technique, tester is unknown to the internal structure of the item being tested and in White Box Testing the internal structure is known to tester. The internal structure is partially known in Gray Box Testing. This includes access to internal data structures and algorithms for purpose of designing the test cases.

Gray Box Testing is named so because the software program is like a semitransparent or grey box inside which tester can partially see. It commonly focuses on context-specific errors related to web systems. It is based on requirement test case generationbecause it has all the conditions presented before the program is tested.



**Objective of Gray Box Testing:**   
The objective of Gray Box Testing is:

1. To provide combined advantages of both black box testing and white box testing.
2. To combine the input of developers as well as testers.
3. To improve overall product quality.
4. To reduce the overhead of long process of functional and non-functional testings.
5. To provide enough free time to developers to fix defects.
6. To test from the user point of view rather than a designer point of view.

**Gray Box Testing Techniques:**

* **Pattern Testing:**   
  To perform the testing, previous defects are analyzed. It determines the cause of the failure by looking into the code. Analysis template includes reasons for the defect. This helps test cases designed as they are proactive in finding other failures before hitting production.
* **Orthogonal Array Testing:**   
  It is mainly a black box testing technique. In orthogonal array testing, test data have n numbers of permutations and combinations. Orthogonal array testing is preferred when maximum coverage is required when there are very few test cases and test data is large. This is very helpful in testing complex applications.
* **Regression Testing:**   
  Regression testing is testing the software after every change in the software to make sure that the changes or the new functionalities are not affecting the existing functioning of the system. Regression testing is also carried out to ensure that fixing any defect has not affected other functionality of the software.

**Advantages of Gray Box Testing:**

* Users and developers have clear goals while doing testing.
* Gray box testing is mostly done by the user perspective.
* Testers are not required to have high programming skills for this testing.
* Gray box testing is non-intrusive.
* Overall quality of the product is improved.
* In gray box testing, developers have more time for defect fixing.
* By doing gray box testing, benefits of both black box and white box testing is obtained.
* Gray box testing is unbiased. It avoids conflicts between a tester and a developer.
* Gray box testing is much more effective in integration testing.

**Disadvantages of gray box testing:**

* Defect association is difficult when gray testing is performed for distributed systems.
* Limited access to internal structure leads to limited access for code path traversal.
* Because source code cannot be accessed, doing complete white box testing is not possible.
* Gray box testing is not suitable for algorithm testing.
* Most of the test cases are difficult to design.

# Non-Functional Testing

**Non-functional Testing** is a type of [Software Testing](https://www.geeksforgeeks.org/software-testing-basics/) that is performed to verify the non-functional requirements of the application. It verifies whether the behavior of the system is as per the requirement or not. It tests all the aspects that are not tested in functional testing. Non-functional testing is a software testing technique that checks the non-functional attributes of the system. Non-functional testing is defined as a type of software testing to check non-functional aspects of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing. Non-functional testing is as important as functional testing.

### Objectives of Non-functional Testing

The objectives of non-functional testing are:

* **Increased usability:**To increase usability, efficiency, maintainability, and portability of the product.
* **Reduction in production risk:**To help in the reduction of production risk related to non-functional aspects of the product.
* **Reduction in cost:**To help in the reduction of costs related to non-functional aspects of the product.
* **Optimize installation:**To optimize the installation, execution, and monitoring way of the product.
* **Collect metrics:**To collect and produce measurements and metrics for internal research and development.
* **Enhance knowledge of product:**To improve and enhance knowledge of the product behavior and technologies in use.

### Non-Functional Testing Techniques

* **Performance testing:**[Performance testing](https://www.geeksforgeeks.org/performance-testing-software-testing/) is a type of testing to ensure that a software program or system meets specific performance goals, such as response time or throughput. For example, organizations perform performance tests in order to identify performance-related bottlenecks.
  + - **Load Testing:** [Load testing](https://www.geeksforgeeks.org/software-testing-load-testing/) is a type of testing to ensure that a software program or system can handle a large number of users or transactions. For example, Running multiple applications on the computer simultaneously.
    - **Stress Testing:** [Stress testing](https://www.geeksforgeeks.org/stress-testing-software-testing/) is a type of testing to ensure that a software program or system can handle an unusually high load. For example, extremely large numbers of concurrent users try to log into the application.
    - **Scalability Testing:** [Scalability testing](https://www.geeksforgeeks.org/software-testing-scalability-testing/) is a type of testing to ensure that a software program or system can be scaled up or down to meet changing needs. For example, to measure the application’s capability to scale up or scale out in terms of non-functional capability.

* + - **Stability testing:** Stability Testing is the testing process to determine the ability of the software product to perform its required functions under specified conditions for a stated period of time, or for a certain number of operations. In other words, it is a testing of the stability of a software product.
* **Usability testing:**[Usability testing](https://www.geeksforgeeks.org/usability-testing/) is a type of testing to ensure that a software program or system is easy to use. For example, on the e-commerce website, it can be tested whether the users can easily locate the Buy Now button or not.
* **Compatibility testing:**[Compatibility testing](https://www.geeksforgeeks.org/compatibility-testing-in-software-engineering/) is a type of testing to ensure that a software program or system is compatible with other software programs or systems. For example, in this, the tester checks that the software is compatible with other software, operating systems, etc.

### Benefits of Non-functional Testing

* **Improved performance:** Non-functional testing checks the performance of the system and determines the performance bottlenecks that can affect the performance.
* **Less time-consuming:** Non-functional testing is overall less time-consuming than the other testing process.
* **Improves user experience:** Non-functional testing like Usability testing checks how easily usable and user-friendly the software is for the users. Thus, focus on improving the overall user experience for the application.
* **More secure product:**As non-functional testing specifically includes security testing that checks the security bottlenecks of the application and how secure is the application against attacks from internal and external sources.

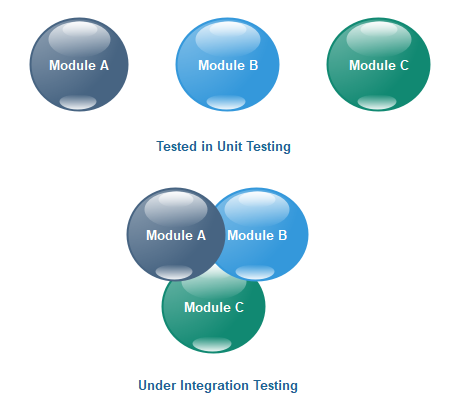
### Limitations of Non-functional Testing

* **Non-functional tests are performed repeatedly:**Whenever there is a change in the application, non-functional testing is performed again. Thus, it is more resource intensive.
* **Expensive in case of software update:**In case of software update, non-functional testing is performed again thus incurring extra charges to re-examine the software, and thus software becomes expensive.

# Integration testing

Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.

[Unit testing](https://www.javatpoint.com/unit-testing) uses modules for testing purpose, and these modules are combined and tested in integration testing. The Software is developed with a number of software modules that are coded by different coders or programmers. The goal of integration testing is to check the correctness of communication among all the modules.



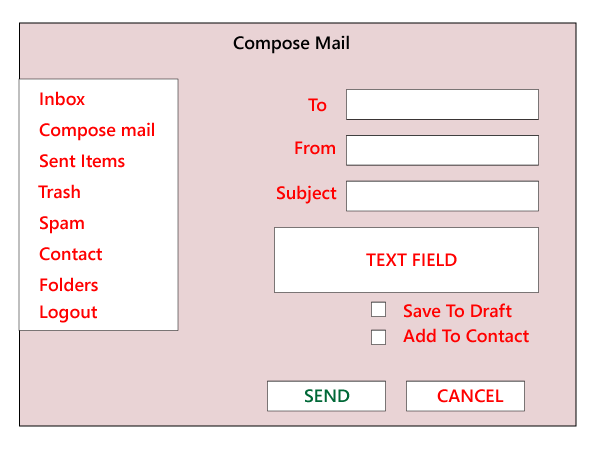
Once all the components or modules are working independently, then we need to check the data flow between the dependent modules is known as **integration testing**.

## **Example of integration testing**

Let us assume that we have a **Gmail** application where we perform the integration testing.

First, we will do **functional testing** on **the login page**, which includes the various components such as **username, password, submit, and cancel** button. Then only we can perform integration testing.

The different integration scenarios are as follows:



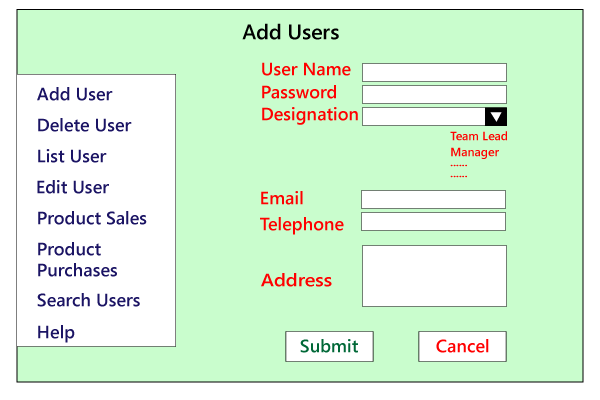
**Scenarios1:**

* First, we login as **P** users and click on the **Compose** mail and performing the functional testing for the specific components.
* Now we click on the **Send** and also check for **Save Drafts**.
* After that, we send a **mail** to **Q** and verify in the **Send Items** folder of P to check if the send mail is there.
* Now, we will **log out** as P and login as Q and move to the **Inbox** and verify that if the mail has reached.

**Secanrios2:** We also perform the integration testing on **Spam** folders. If the particular contact has been marked as spam, then any mail sent by that user should go to the spam folder and not in the inbox.

#### **Note: We will perform functional testing for all features, such as to send items, inbox, and so on.**

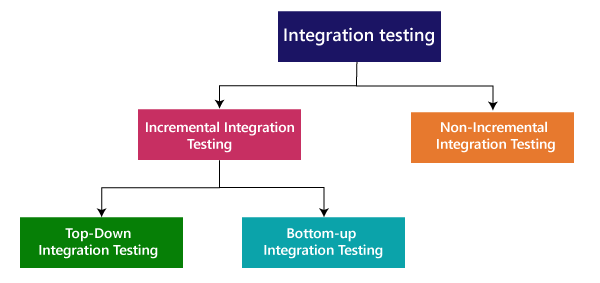
As we can see in the below image, we will perform the [**functional testing**](https://www.javatpoint.com/functional-testing) for all the **text fields and every feature**. Then we will perform **integration testing** for the related functions. We first test the **add user, list of users, delete user, edit user,** and then **search user**.



## **Types of Integration Testing**

Integration testing can be classified into two parts:

* **Incremental integration testing**
* **Non-incremental integration testing**

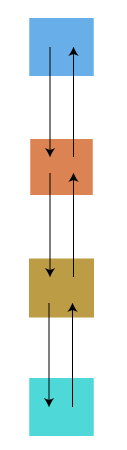


### **Incremental Approach**

In the Incremental Approach, modules are added in ascending order one by one or according to need. The selected modules must be logically related. Generally, two or more than two modules are added and tested to determine the correctness of functions. The process continues until the successful testing of all the modules.

**OR**

In this type of testing, there is a strong relationship between the dependent modules. Suppose we take two or more modules and verify that the data flow between them is working fine. If it is, then add more modules and test again.



**For example:** Suppose we have a Flipkart application, we will perform incremental integration testing, and the flow of the application would like this:

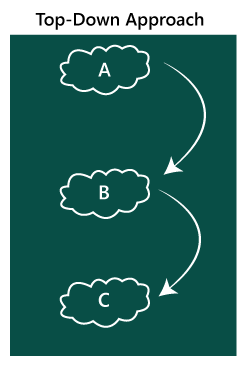
Flipkart→ Login→ Home → Search→ Add cart→Payment → Logout

Incremental integration testing is carried out by further methods:

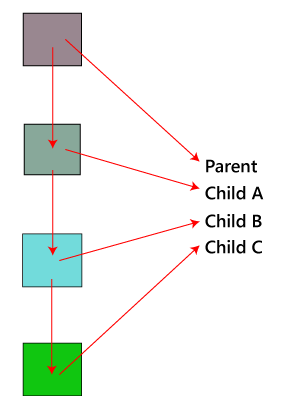
* Top-Down approach
* Bottom-Up approach

### **Top-Down Approach**

The top-down testing strategy deals with the process in which higher level modules are tested with lower level modules until the successful completion of testing of all the modules. Major design flaws can be detected and fixed early because critical modules tested first. In this type of method, we will add the modules incrementally or one by one and check the data flow in the same order.



In the top-down approach, we will be ensuring that the module we are adding is the **child of the previous one like Child C is a child of Child B** and so on as we can see in the below image:



**Advantages:**

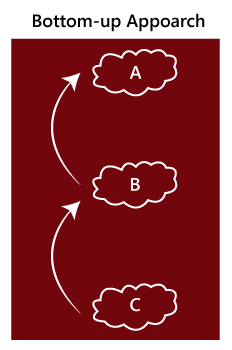
* Identification of defect is difficult.
* An early prototype is possible.

**Disadvantages:**

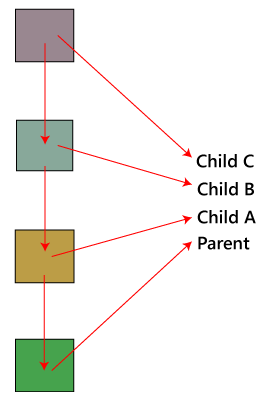
* Due to the high number of stubs, it gets quite complicated.
* Lower level modules are tested inadequately.
* Critical Modules are tested first so that fewer chances of defects.

### **Bottom-Up Method**

The bottom to up testing strategy deals with the process in which lower level modules are tested with higher level modules until the successful completion of testing of all the modules. Top level critical modules are tested at last, so it may cause a defect. Or we can say that we will be adding the modules from **bottom to the top** and check the data flow in the same order.



In the bottom-up method, we will ensure that the modules we are adding **are the parent of the previous one** as we can see in the below image:



**Advantages**

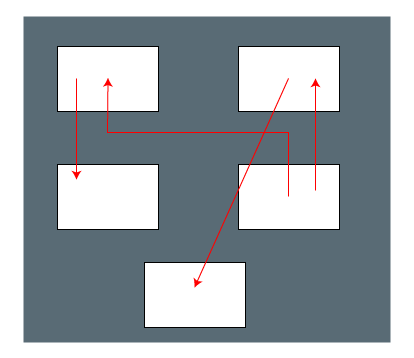
* Identification of defect is easy.
* Do not need to wait for the development of all the modules as it saves time.

**Disadvantages**

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

### **Non- incremental integration testing**

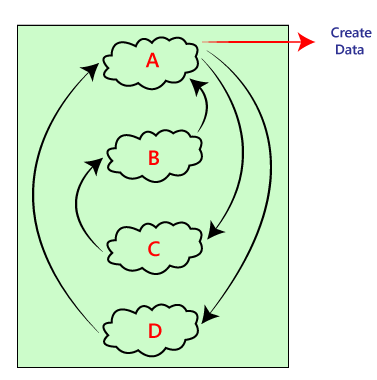
We will go for this method, when the data flow is very complex and when it is difficult to find who is a parent and who is a child. And in such case, we will create the data in any module bang on all other existing modules and check if the data is present. Hence, it is also known as the **Big bang method**.



### **Big Bang Method**

In this approach, testing is done via integration of all modules at once. It is convenient for small software systems, if used for large software systems identification of defects is difficult.

Since this testing can be done after completion of all modules due to that testing team has less time for execution of this process so that internally linked interfaces and high-risk critical modules can be missed easily.



**Advantages:**

* It is convenient for small size software systems.

**Disadvantages:**

* Identification of defects is difficult because finding the error where it came from is a problem, and we don't know the source of the bug.
* Small modules missed easily.
* Time provided for testing is very less.
* We may miss to test some of the interfaces.