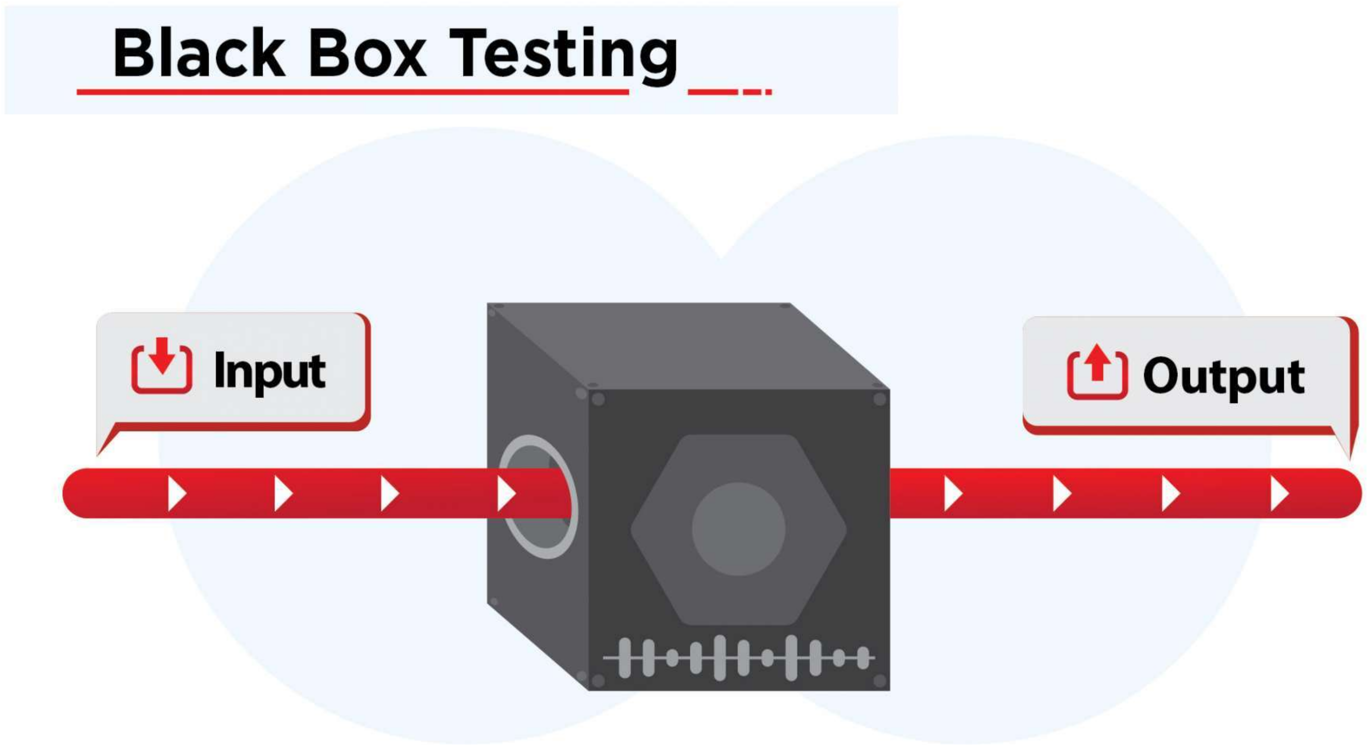
We have already learned about the factors that influence how to choose the test techniques. Now let's dive into the actual test techniques. At a high level, the test techniques can be categorized into two groups - ***Black-box techniques*** and ***White-box techniques***. In this article, we will discuss the Black-box testing techniques. We will cover each method in detail as a separate article, but let's give you an overview of them here!

What is the Black box testing technique?
How to do Black box testing?
Black-box technique and its typical characteristics
Types of Black-box testing techniques

What is Black Box Testing Technique?

The black box is a testing type where the tester doesn't know the internal design of the component or system which undergoes testing. So if we don't know the internal design, how do we ensure that our test cases can still find defects and provide excellent coverage? It is where Black Box testing techniques come into the picture. These are scientific and time tested techniques that will help us get maximum coverage in minimum test cases.



How to do Black Box Testing

We already know what Black box testing is, and we are going to learn it's techniques (*which is an efficient way of creating test cases for a black box testing type*). But before we dive in there, it's essential to know a few critical characteristics of Black box testing. Additionally, it's equally important to understand what it takes to execute a black box test type successfully.

- ***Understand the Requirements**: The first step to execute a black box testing is to understand the system requirements thoroughly. As the internal system design is not known to us, the system requirements act as the source of truth to create effective test cases. Any understanding gap must be clarified, and the tester should ensure that specifications are detailed enough to write a black-box test case successfully.*
- ***Effective Test case Creation**: Once the requirements are understood, the next step is to create test cases. The biggest challenge is to write test cases that are effective enough to find defects (Remember - We don't know the internal design !). We will discuss the actual techniques in detail later, but keep in mind that it's essential to ensure that our test cases cover below*.
  - ***Happy Paths** - These are the Critical business scenarios where we give positive inputs. Additionally, happy paths involve the most commonly used steps. For instance, a successful login if I provide a valid user id and password*.
  - ***Alternate Paths** - These are critical business scenarios that involve positive inputs. However, the steps are not the most commonly used. E.g., I should be able to automatically log in if my session doesn't expire without inputting user id and password again!*
  - ***Negative Paths** - These are business scenarios where we give negative inputs, and we expect the system to throw error message (Of course, a user-friendly one !)*
- ***Adaptive Execution** - We all do execution, and we compare the actual result with expected results, so what is adaptive about it? Well, the black box testing is guesswork- a Scientific one, though! So, what happens when you find some critical defects in a particular area but don't find any in other areas? In such cases, we need to adapt our execution to focus more on the buggy areas - some developers had a bad day! Additionally, we review our test coverage and techniques where we are not finding bugs*.
- ***Defects Logging and Closure** - The last step is to log the defects, and retest it once it's fixed. Everyone does that, but what you can do better is to test scenarios around the defect fix. Often a fix leads to another defect, and it's best to catch it while retesting the defect*.

Black-Box Technique and its Common Characteristics

Before we dive into actual techniques, let's discuss some of the common characteristics of Black box techniques.

- *Test cases (including test conditions and test data) derive from Requirements, User Stories, User cases, and user specifications.*
- *Test cases are used to detect gaps between the requirements and the actual implementation.*
- *Measurement of Coverage happens in terms of requirements and the test cases written for these requirements*.
- *Black box testing techniques apply to all test levels (Component Testing, Component Integration testing, System Testing, and Unit Testing)*.

As you can see, we are not looking at design and code coverage (*Why ? Because it's a black box !*).

Types of Black-Box Testing Techniques

Let's have a quick overview of black-box testing techniques. We will cover these in detail in subsequent articles. However, this should give you an overall understanding.

***Equivalence Partitioning**: We use this technique when we have an input range to enter. E.g., A User registration form where we need to enter a mobile number. Equivalence partitioning works on the logic that you divide the input conditions into valid and invalid partitions. All the input values in that partition should behave the same. So let's say that we have to partition the mobile number input field. As we know, the mobile number should be ten digits. So the partitions will be*

> 10 Digits Mobile number (Invalid Partition)

= 10 Digits Mobile number (Valid Partition)

< 10 Digits Mobile number (Invalid Partition)

***Boundary Value Analysis**: Equivalence partitioning helped reduced test conditions, and increase coverage, but it was not enough. Realization surfaced that faults often occur at the boundary of equivalence classes. Boundary value analysis verifies the borders of the equivalence classes. On every border, testing of the exact boundary value and both nearest adjacent values happens (inside and outside the equivalence class). For the same, e.g., of the mobile number input field, the boundary conditions will be :*

9 Digit mobile number
10 Digit mobile number
11 Digit mobile number

***Decision Table Testing:** We use it when we need to execute complex business rules. Usually, this is in the form of a table where input combinations and corresponding outputs are listed. These combinations are used to create test cases.*

Conditions	**1st Input**	**2nd Input**	**3rd Input**
Salaried?	Y	N	Y
Monthly Income > 25000	N	Y	Y
IT Returns available?	Y	N	Y
Loan Approval	N	N	Y

***State Transition Testing**: We use this testing when the output of the system can change even when you give the same input. It means that the system can have different states despite the same inputs. As an example, consider that you withdraw $1000 from the ATM, and the cash comes out. Later you again try to withdraw $1000, but this time you are refused the money. So the input is the same in both the cases ($1000), but the state has changed. After the first transaction, the account doesn't have enough money, so the state of account has changed. Usually, we create a state diagram, and derivation of the test cases happens out of that.*

***Use Case Testing**: Use cases are defined in terms of System behavior by an actor (User of the system). A use case describes the process flow from the actor's point of view, and the use cases derive test cases. Consider a flight booking system. It will have 3 actors (users). Additionally, test cases writing will be basis what these actors can do in the system.*

Customer who will book the tickets
Agent who will book the ticket on behalf of the customer
Admin (Like Makemytrip personnel) who can provide support for any issues

All these black-box techniques ensure that we are writing test conditions that are more likely to catch defects. Consequently, keeping the number of test cases at a minimal requirement. In the absence of these techniques, either the test case combinations will increase (*thereby increasing cost and time*), or we will miss on ensuring the right coverage.

To conclude, the purpose of this article was to give you a high-level overview of the Black Box techniques. We will discuss each of these testing techniques in detail in subsequent articles, where we will deep dive with the help of detailed explanations and examples!