# **Black Box Testing – Understanding the Basics**

This article highlights the basics of black-box testing when to perform black-box testing and what are the benefits of using this form of the testing. It also gives insights into the various techniques used to perform black-box testing.

## **What is black box testing?**

The black box is a powerful technique to check the application under test from the user’s perspective. Black box testing is used to test the system against external factors responsible for software failures. This testing approach focuses on the input that goes into the software, and the output that is produced. The testing team does not cover the inside details such as code, server logic, and development method.

Black box testing is based on the requirements and checks the system to validate against predefined requirements.

****Various parameters checked in black box testing are:****

* Accurate actions performed by users
* System’s interaction with the inputs
* The response time of the system
* Use of data structures Issues in the user interface
* Usability issues
* Performance issues
* Abrupt application failure, unable to start or finish

## **IMG_257**

## **Black box testing example:**

A simple login screen of software or a web application will be tested for seamless user login. The login screen has two fields, username and password as an input and the output will be to enable access to the system.

A black box testing will not consider the specifications of the code, and it will test the valid username and password to login to the right account.

This form of testing technique will check the input and output.

* A user logged in when inputs a present username and correct password
* A user receives an error message when enters username and incorrect password

The black box testing is also known as an opaque, closed box, function-centric testing. It emphasizes on the behavior of the software. Black box testing checks scenarios where the system can break.

For example, a user might enter the password in the wrong format, and a user might not receive an error message on entering an incorrect password.

### ****When we do Black Box testing?****

* Unlike traditional white box testing, black box testing is beneficial for testing software usability.
* The overall functionality of the system under test
* Black box testing gives you a broader picture of the software.
* This testing approach sees an application from a user’s perspective.
* To test the software as a whole system rather than different modules.

## ****Various approaches to black-box testing****

There are a set of approaches for black-box testing.

****Manual**[UI Testing](https://reqtest.com/testing-blog/gui-testing-tutorial/)**: In this approach, a tester checks the system as a user. Check and verify the user data, error messages.

****Automated UI Testing****: In this approach, user interaction with the system is recorded to find errors and glitches. Testers can set record demand as per schedule.

****Documentation Testing****: In this approach, a tester purely checks the input and output of the software. Testers consider what system should perform rather than how. It is a manual approach to testing.

## ****What are Black Box testing techniques?****

There are various [test case design techniques](https://reqtest.com/testing-blog/test-case-design-techniques/) applied for black-box testing:

****Boundary Value Analysis****

It is the widely used black-box testing, which is also the basis for equivalence testing. Boundary value analysis tests the software with test cases with extreme values of test data. BVA is used to identify the flaws or errors that arise due to the limits of input data.

For example: Taking inputs for a test case data for an age section should accept a valid data of anything between 1-100. According to BVP analysis, the software will be tested against four test data as -1, 1, 100, and 101 to check the system’s response using the boundary values.

****Equivalence partitioning****

This test case designing techniques checks the input and output by dividing the input into equivalent classes. The data must be tested at least once to ensure maximum test coverage of data. It is the exhaustive form of testing, which also reduces the redundancy of inputs.

For example: Taking inputs for a test case data for the example mentioned above will have three classes from which one data will be tested.

Valid class: 1 to 100 (any number), Invalid class: -1 (checking the lowest of lowest), Invalid class: 101(highest of highest).

****[Also Read:**** [What is Boundary Value Analysis and Equivalence Partitioning?]](https://reqtest.com/testing-blog/what-is-boundary-value-analysis-and-equivalence-partitioning/)

****State Transition Testing****

This testing technique uses the inputs, outputs, and the state of the system during the testing phase. It checks the software against the sequence of transitions or events among the test data.

Based on the type of software that is tested, it checks for the behavioral changes of a system in a particular state or another state while maintaining the same inputs.

For example, A login page will let you input username and password until three attempts. Each incorrect password will be sent the user to the login page. After the third attempt, the user will be sent to an error page. This state transition method considers the various states of the system and the inputs to pass only the right sequence of the testing.

****Decision Table Testing****

This approach creates test cases based on various possibilities. It considers multiple test cases in a [decision table](https://reqtest.com/requirements-blog/a-guide-to-using-decision-tables/) format where each condition is checked and fulfilled, to pass the test and provide accurate output. It is preferred in case of various input combinations and multiple possibilities.

For example, A food delivery application will check various payment modes as input to place the order — decision making based on the table.

Case1: If the end-user has a card, then the system will not check for cash or coupon and will take action to place the order.

Case2: If the end-user has a coupon will not be checked for a card or cash and action will be taken.

Case3: if the end-user has cash, the action will be taken.

Case4: If the end-user doesn’t have anything, then action will not be taken.

****Graph-Based Testing:****

It is similar to a decision-based test case design approach where the relationship between links and input cases are considered.

****Error Guessing Technique:****

This method of designing test cases is about guessing the output and input to fix any errors that might be present in the system. It depends on the skills and judgment of the tester.

Comparison testing

This method uses the two different versions of the same software to compare and validate the results.

## ****How Is Black Box testing performed?****

Black box testing method is done as follows:

* The first step to black-box testing is to understand the requirement specifications of the application under test. An accurate and precise [SRS document](https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document) should be there.
* The next step is to evaluate the set of valid inputs and test scenarios to test the software.  The goal is to save time and [get good test coverage](https://reqtest.com/testing-blog/test-coverage-metrics/).
* Prepare the test cases to cover a maximum range of inputs.
* The test cases are run in the system to generate output, which is validated with the expected outcome to mark pass or fail.
* The failed steps are marked and sent to the development team to fix them.
* Retest the system using various testing techniques to verify its recurring nature or to pass it.

The black box testing can be easily used to check and validate the entire software development life cycle. It can be used at various stages such as unit, integration, acceptance, system, and regression to evaluate the product.

## ****What are the benefits of Black Box testing?****

* The tester doesn’t need any technical knowledge to test the system. It is essential to understand the user’s perspective.
* Testing is performed after development, and both the activities are independent of each other.
* It works for a more extensive coverage which is usually missed out by testers as they fail to see the bigger picture of the software.
* Test cases can be generated before development and right after specification.
* Black box testing methodology is close to agile.

#### ****Conclusion****

Black box testing helps to find the gaps in functionality, usability, and other features. This form of testing gives an overview of software performance and its output. It improves software quality and reduces the time to market. This form of testing mitigates the risk of software failures at the user’s end.