Black Box Testing | Complete UK Guide

What is black box testing?

Black box testing, which is sometimes referred to as ‘Behavioural Testing’, is a software testing method which approaches the software or application being tested from the point of view of the end user. This means that, like the end user, the black box tester will have no knowledge of the internal factors governing the performance of the software, such as the code and server logic. Instead, black box testing is based on the inputs fed into the software and the outputs which they generate, and checking whether the outputs generated match the predefined requirements expected by the end user. Black box testing can be carried out on any software system, from custom applications to search engines like Google and an operating system like Windows.

Steps involved in black box testing

Although the details of any black box testing will vary depending on the nature of the software and the specifications outlined by the developer – i.e. the expected outputs an end user should experience – there are certain generic steps involved in any black box testing process, and these are as follows:

- Examine the requirements and specifications of the system to be tested
- The tester will then choose the inputs to be used for a positive test case scenario and a negative test case scenario. In a positive test case scenario the inputs will be valid and the system being tested is being checked to see if it handles those inputs correctly. In a negative test case scenario the inputs will be invalid, and the system tested to verify that it is able to detect this fact.
- Having devised the inputs the tester will then determine what the expected output for each input will be.
- A software tester will then construct test cases which utilise the chosen inputs.
- The test cases which have been devised are then executed and the outputs generated are collected.
- The actual outputs deriving from the tests are then compared with the expected outputs.
- Any defects which are identified by this process are fixed and the tests carried out again.

The parameters of black box testing cover issues such as the actions which are expected to be carried out by users, the way in which the system in question interacts with the inputs, the amount of time which any response takes and general performance and usability issues. These issues can be as basic as an abrupt failure mid-way through a process or the system being unable to start or finish a process.

General types of black box testing

Generally speaking, black box testing can be broken down into three specific areas – functional testing, non-functional testing and regression testing:

Functional testing – this is based on checking whether the software is able to perform specific functions expected of it, and can be performed to test the basic functions – i.e. whether a user can log in to a system using the correct username and password and is prevented from doing so when entering incorrect details – the integration between the components of the system and the operation of the system as a whole.

Non-functional testing – where functional testing is concerned with finding out whether the software is capable of delivering specific outputs, non-functional testing is designed to examine exactly *how* the outputs are delivered. Non-functional black box testing will look at whether the software is easy to understand and use, whether it performs at the expected level when placed under the predicted levels of stress, and continues to do so when the stress hits the expected operational peak, and if the software is compatible with specified operating systems, devices, screen sizes and browsers. In short, non-functional testing is based on the premise that the software is capable of delivering the minimum functionality expected, and is now being tested to ensure the user experience is as specified by the developers. Non-functional testing will also look at the security aspect of the software, and whether it has been designed to resist common security threats.

Regression Testing – this type of black box testing is carried out on a newer version of an existing piece of software. It is intended to test whether the software has suffered a regression – i.e. a decline in the capabilities it offers – when an existing version is updated. The regression testing could be functional in nature, testing whether a specific feature in the software still works as specified, or non-functional, testing whether a specific operation is carried out as seamlessly and quickly as it was in the older version.

Specific types of black box testing

In order to reduce the workload and save time, the tester can divide particular inputs into groups and only test one specific example from each of these groups. A system might have been designed to ask for the user's age to be inputted, with different outcomes presented for users aged up to the age of 10, between 10 and 18, and over 18. Rather than run tests on every possible age the tester only needs to take one example from each of the three groups and check that this example delivers the specified outcome.

Boundary Value Analysis

A certain field may be intended only to accept values within specific boundaries, for example between 0 and 99. The tester will run tests which concentrate on the values 0 and 99, but also on -1 and 100, to check that they are accepted and rejected as they should be.

Decision Table Testing

Many systems deliver outputs based on a set of specific conditions. Systems such as this would include the pricing for specific items of variable types and sizes. The information entered in terms of information such as the size of a table and the type of wood it is made of would be expected to produce specific results in the form of the price quoted to the user.

Another example is the log-in page of an email account. To access the account a user would be expected to enter their email address and a password. If both are entered correctly then the user will move on to the account homepage. If either is entered incorrectly the user should see an error message specifying either ‘Incorrect Email’ or ‘Incorrect Password’. A decision table created to test this aspect of the software would include two conditions – the email address and the password – and four possible outcomes, depending upon whether either or both of the conditions were entered incorrectly:

- Email address true/Password true = Account homepage
- Email address true/Password false = ‘Incorrect password’ message
- Email address false/password true = ‘Incorrect email’ message
- Email address false/Password false = ‘Incorrect email

In this example all possible combinations of password and email address have been included, together with the possible outcomes. In more complex systems the number of options might grow but the principle of testing all possible combinations would remain the same.

State Transition Testing

State transition testing is designed to test situations in which the system is meant to transition from one state to another. A common example is the login mechanism of a system which, after three failed attempts (due to incorrectly entered username or password details) is meant to transition to an error page and lock the account.

State transition testing will be designed to ensure that if the correct details are entered on any of the first, second or third attempts the user is directed to the homepage, but if incorrect details are entered three times the user is directed to an error message, and the account locked. The test would consist of the information being entered 4 times:

- First attempt valid = redirection to the homepage
- First attempt invalid/second attempt valid = redirection to the homepage
- First attempt invalid/second attempt invalid/third attempt valid = redirection to the homepage
- First attempt invalid/second attempt invalid/third attempt invalid = redirection to an error message and the account being locked

Error Guessing

This technique depends, to a degree, upon the experience and expertise of the tester. It is based on testing for mistakes which are identified as being commonplace in similar systems. There isn’t a strict formula for devising an error test since it depends upon the tester’s instinct with regard to any errors the developer might have made. Examples include testing to see what happens if a blank space is entered into a text field, if invalid parameters are entered and if the user can input executable code, something which will have security implications.

The pros and cons of black box testing

The pros

- Tests can be carried out by testers who don't have technical knowledge of the software, although they must have a firm grasp of the user's point of view
- Black box testing is likely to produce unbiased results since the testing team and development team work independently
- It's possible to save time by generating test cases immediately after specification and before development
- Black box tests have relatively low complexity as they are based on standard user behaviour

The cons

- The tests themselves are more difficult to automate than those which test aspects such as the code of the software
- The user paths being tested have to be prioritised as it is unlikely that any set of tests will be able to identify every single feasible user path
- When a test fails, the tester – lacking the technical programming or IT knowledge of a developer – will struggle to identify the root cause of the problem
- Some of the tests may simply be repeating testing already carried out by the software designer and/or developer

Alternatives to black box testing

The alternatives to black box testing are known as white box testing and grey box testing. White box testing is concerned with the internal workings of the software, including aspects such as the source code, network protocols and IP addresses. Unlike black box testing, white box testing is designed to analyse the internal structure of the software being tested rather than its functionality, and to do so from the point of view of a developer rather than an end user. Between black box testing and white box testing is grey box testing, which combines aspects of both approaches. A grey box tester will have some understanding of the internal coding of the software, and the tests themselves will be designed to uncover faults in both the initial code and the functionality as experienced by a user.