Black Box testing:

A type of testing that is only concerned with the inputs and outputs of a system, the implementation is not being evaluated. Usually, tests are based on a set of predefined specifications for the system. There are 3 categories of BBT:

Functional Testing:

- This verifies that each function performs according to the specification. This is done by supplying a set of inputs to the function and testing if the correct outputs are produced.

Regression Testing:

- This testing is performed after something has changed or introduced, it makes sure that no old bugs have been reintroduced or no new bugs have been created.

Nonfunctional Testing:

- This is not concerned with the correct functionality of the system but its non-functional attributes. It checks attributes like performance, usability and scalability.

Pros:

- A tester needs very little programming experience.
- It is efficient when working with large systems.
- Test cases can be easily reproduced to investigate bugs.

Cons:

- Without clear specs, it can be difficult to implement.
- Internal functions can go untested.
- Can be difficult if inputs are complex.

White Box Testing

White box is similar to black box but the tester knows the system internals. Here the tester can test individual code snippets, algorithms and methods. This makes sure that the entire system works correctly and efficiently not just the parts the user interacts with.

- Can be slower to implement.
- Takes far more knowledge of coding and the individual system to implement.

Grey Box Testing

Grey box is a mixture of both it has the ease of black box while exposing some of the internal structure. This includes access to internal data structures and algorithms to design the test cases.

7 Principles of Testing

- Testing shows the presence of defects, not their absence.
    - We test to deliberately try and find defects to catch them before they are pushed to live. Trying to prove your code is flawless will mean you miss defects.
- Exhaustive testing is impossible.
    - No program can ever be fully tested. It would not be affordable to try.
- Early testing.
    - Catching defects early is much cheaper. It stops bugs affecting the development of other code.
- Defect clustering.
    - It was found that certain components usually contain the majority of bugs and issues.
- Pesticide paradox.
    - This is based on the theory that when you use pesticides repeatedly on crops, insects will eventually build up immunity. Code acts similarly so you need to change and review your tests periodically.
- Testing is context-dependent.
    - Tailor your tests to how the system will be used.
- Absence-of-errors fallacy.
    - Remember, just because there might be a low number of issues, it does not mean your software is shippable – meeting client expectations and requirements is just as important as ensuring quality.