Software testing is an essential process that ensures the quality and reliability of software products before they are released to users

The primary goal of testing is to increase the probability that an application under test will behave correctly under all circumstances and will meet defined requirements. A program works correctly if:

* It produces the correct output, given a valid input
* It correctly and gracefully rejects an invalid input
* It doesn't hang or crash, given either a valid or invalid input
* It runs correctly for as long as expected
* It behaves as specified

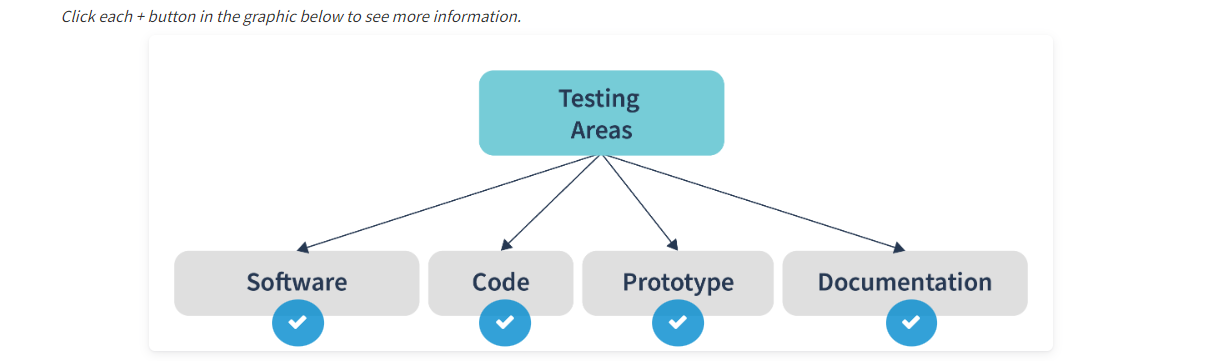
**Software testing is a process of analyzing software and accompanying documentation in order to identify defects and improve the quality of the product.**

**Exhaustive testing** Positive Testing. Negative testing

The only working approach used in those days was **debugging**, the process of detecting and eliminating errors in software code.

70s: • Testing ensures that the program meets the requirements.

• Testing identifies conditions under which the software performs incorrectly.

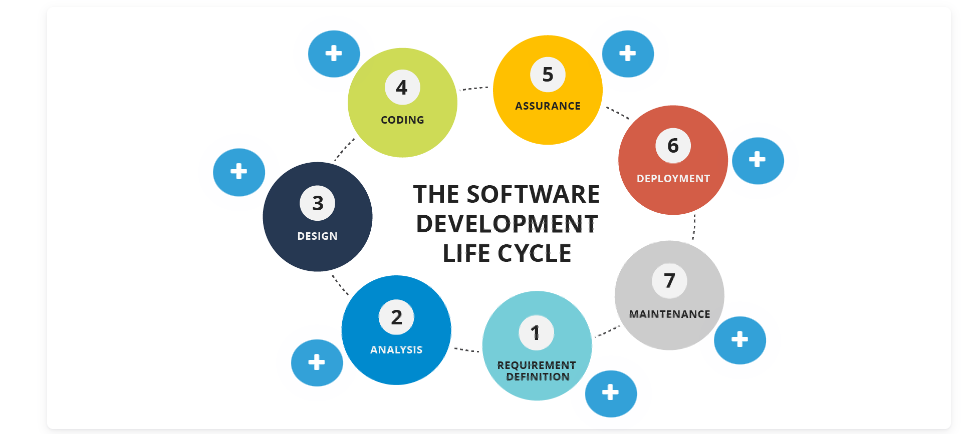
****

**Errors – mistake by human**

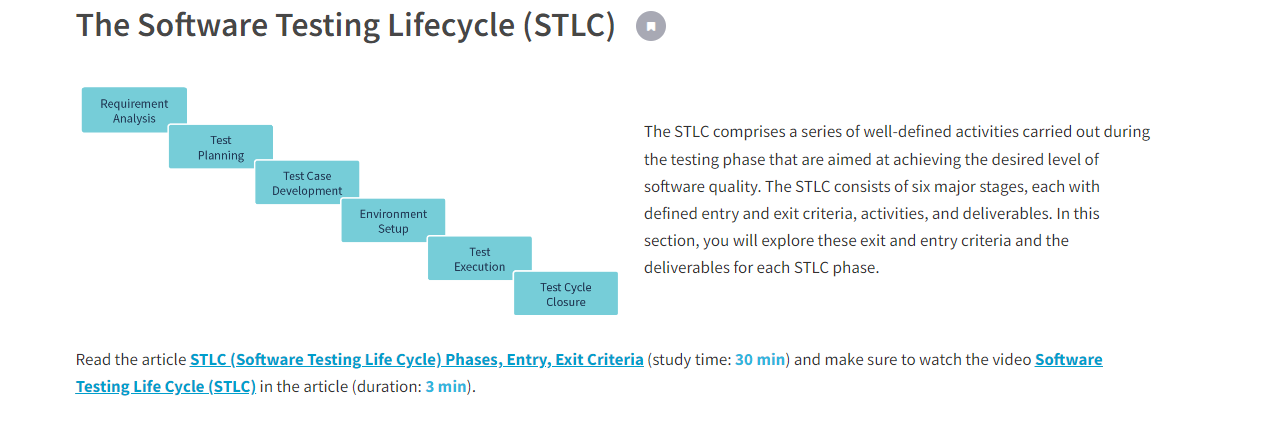
**Defects – issue caused in code because of error**

**Bug – crash resulting in running code with a defect within it**

**known as the software development lifecycle, or SDLC**

****

**The Software Testing Lifecycle (STLC)**

****

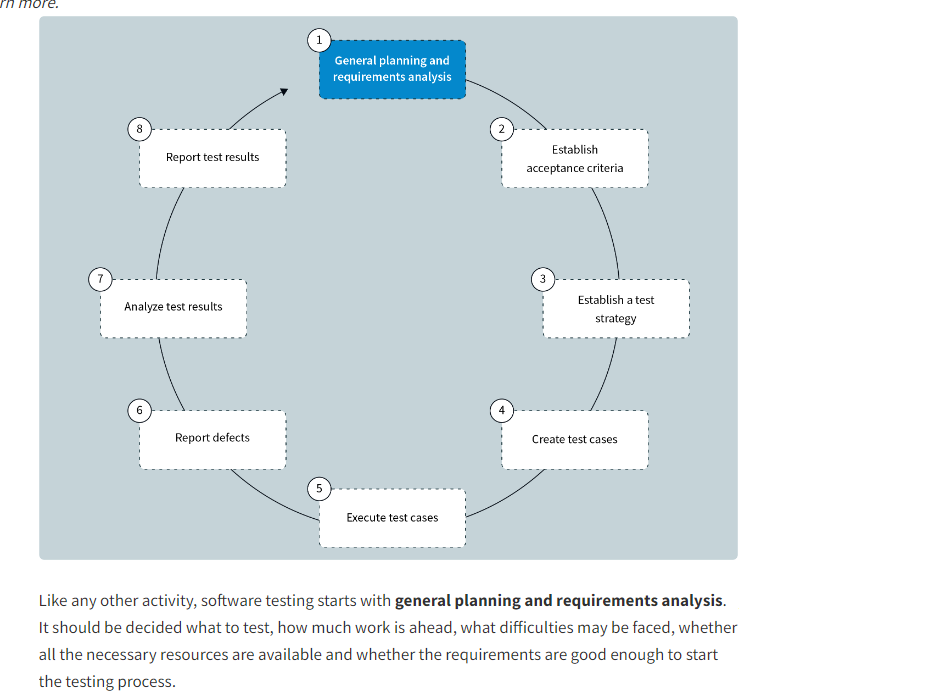
STLC involves both verification and validation activities

1. Requirement Analysis
2. Test Planning
3. Test case development
4. Test Environment setup
5. Test Execution
6. Test Cycle closure

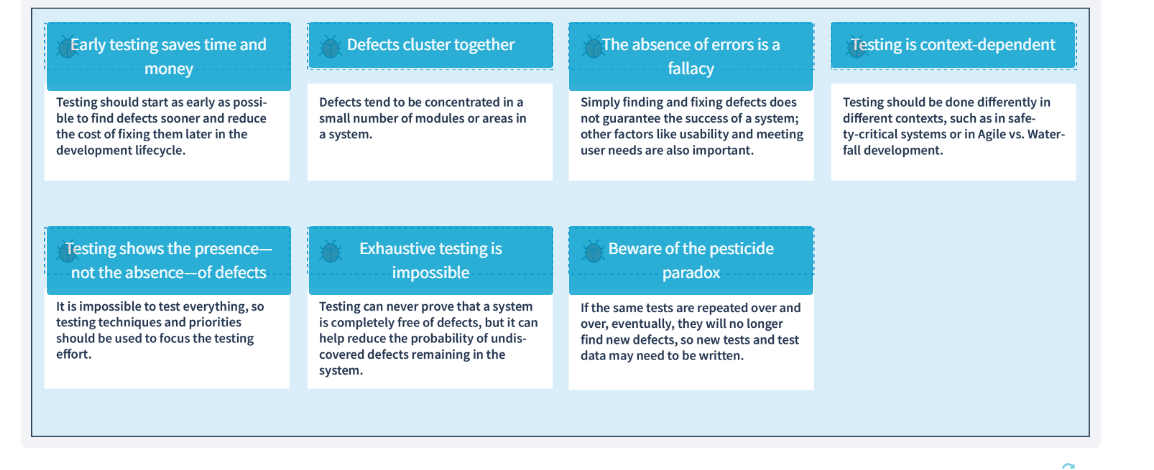
**What is Entry and Exit Criteria in STLC?**

* **Entry Criteria:** Entry Criteria gives the prerequisite items that must be completed before testing can begin.
* **Exit Criteria:** Exit Criteria defines the items that must be completed before testing can be concluded

## Typical Work Tasks

****

## Main Principles

****

## Static and Dynamic Testing

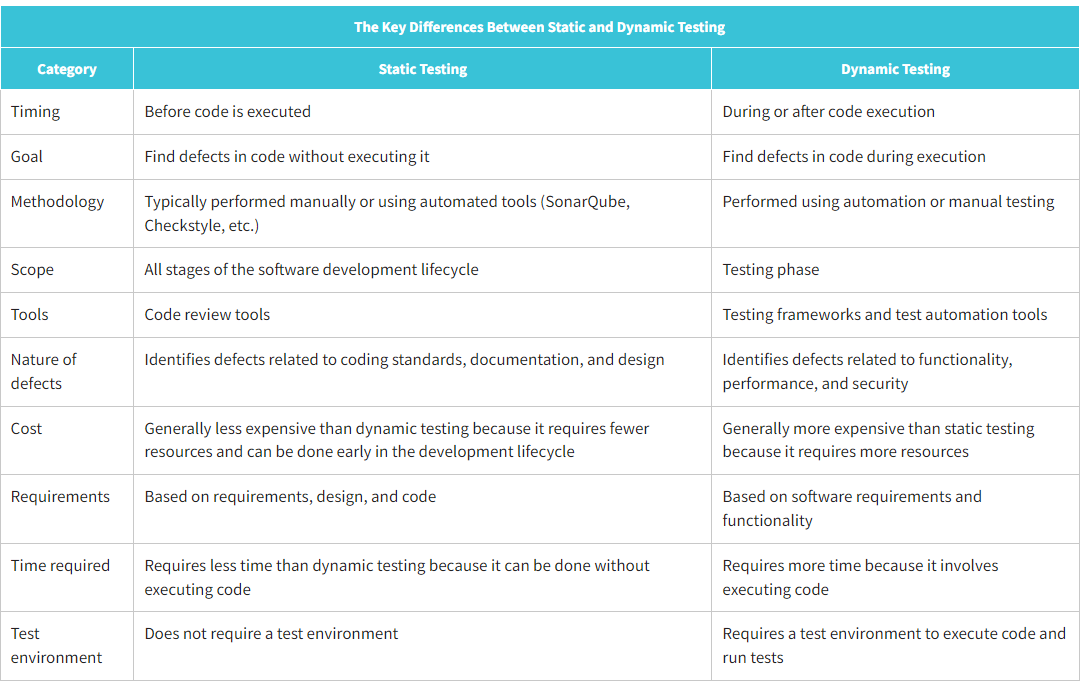
## Static testing aims to identify defects, errors, or vulnerabilities in software design or code structure during the early stages of the software development lifecycle, before the code is executed.

## Dynamic testing aims to identify defects, errors, or vulnerabilities in software behavior or functionality during the later stages of the software development lifecycle, after the code has been executed.

**Static testing** is a software testing approach performed during the early development stages. It checks defects in software without executing its code.

Є два види статік тестінг review(просто подивитись вут) Static analysis

**Dynamic testing** is a testing approach involving code execution to analyze software behavior. A software code under testing should be written, compiled, and launched to conduct dynamic testing. At the same time, with the help of dynamic testing, you can check software performance: processor load, memory usage, response time, etc.

****

## Requirements

## A requirement is a statement that specifies what needs to be implemented in a project. It outlines the necessary functionalities or features without providing any implementation details. A well-written requirement defines what needs to be achieved but does not dictate how it should be achieved. This allows the development team to have the flexibility to design and implement the solution in the most effective and efficient way possible.

Requirements specify **what**should be implemented. A requirement describes a need without describing any implementation details of a solution. A good requirement describes what, not how.

Types:

**Business requirements** express the purpose for which the product is being developed (why the product is needed at all, what benefits are expected from it, how the customer can make a profit with it). The output of the requirements definition at this level is the vision and scope, a document which is typically drawn up in plain text and spreadsheets. It does not include details on system implementation and other technical specifications, but it may well define priorities for the business tasks to be performed, risks.

**User requirements** describe the tasks that the user can perform with the system being developed (reaction of the system to user actions, user scenarios). As the system behavior is being described here, the requirements at this level can be used to estimate the scope of work, project cost, development time, etc. User requirements are formalized as use cases, user stories, and user scenarios

**Business rules** describe the specifics of the subject area (and/or those adopted directly by the customer) processes, constraints and other rules. These rules may relate to business processes, personnel rules, the details of software operation, etc

**Quality attributes** extend non-functional requirements and can be described as project-specific quality attributes (product properties that are not related to functionality, but are important for achieving product goals — performance, scalability, recoverability). There are many quality attributes, but only some subset is really important for any project.

**Functional requirements** describe the behavior of the system, i.e., its activities (calculations, transformations, checks, processing, etc.) In the planning context, the functional requirements mainly influence the system design.

**Non-functional** **requirements** describe the properties of the system (usability, security, reliability, scalability, etc.) that it must have when implementing its behavior. Here you can find a more technical and detailed description of the quality attributes. In the design context, the non-functional requirements have a major influence on the system architecture

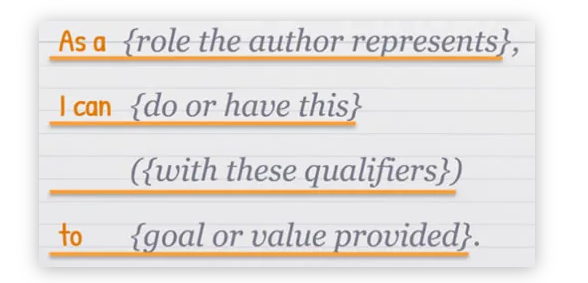
**Limitations or constraints** are factors that limit the choice of ways and means (including tools) to develop a product

**External interfaces** **requirements** describe the interaction of the system under development with other systems and the operating environment

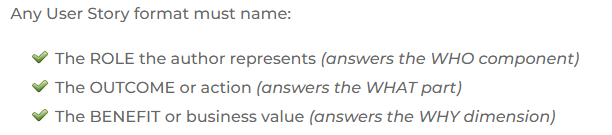
**Data requirements** describe the data structures (and the data itself) that are an integral part of the system under development. This often includes a description of the database and the features of its use.

**User Story**

**User story Складається з 3 - The card, the conversation, the criteria**

****

A User Story must answer the question: **WHO wants WHAT and WHY?**

****

You got the hang of the User Story format “As a <user>, I want <x> because <y>

A user story will be considered as done only after the **acceptance criteria** are met and tested

#### **Acceptance Criteria** AC) are conditions that software must satisfy to be accepted by a user, customer, or the consuming system (in the case of system-level functionality)

**User Acceptance Criteria** specify the [functional and non-functional requirements](https://www.businessanalysisexperts.com/functional-vs-non-functional-requirements-what-are-how-write/) that need to be met before a Story is considered complete by stakeholders

**User** **Acceptance Tests** (aka User Story Tests) confirm that the product meets all Acceptance Criteria and functions correctly

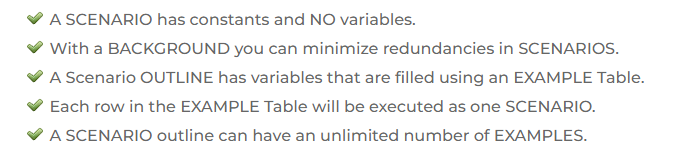
### **The Given-When-Then Format Is a Template to Guide Your Writing**

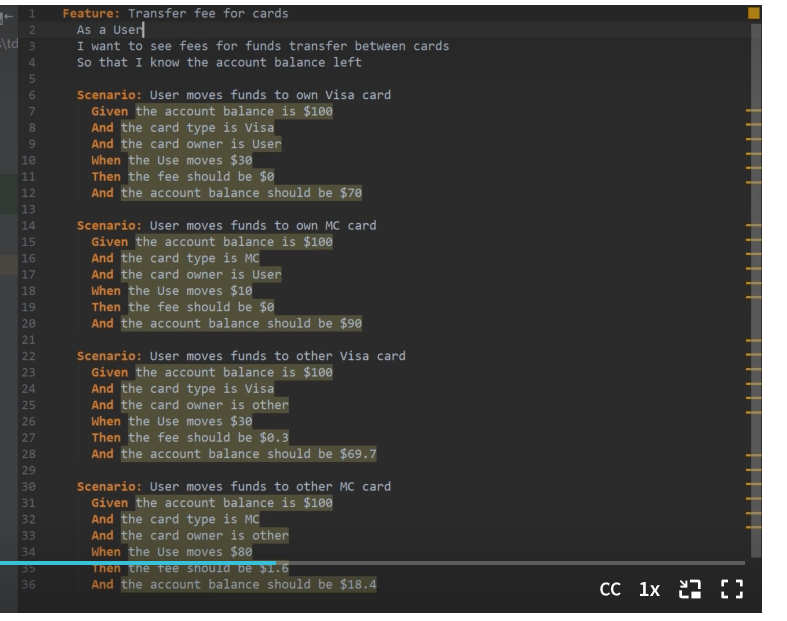
**GIVEN (setup data such as hardware or data)**

In the GIVEN clause, you should express any kind of setup conditions that are necessary for this test to succeed. This encompasses hardware conditions as well as data, files, or records that must have a specific condition for the test to run.

**WHEN (action or event that initiates the Scenario)**  
The WHEN clause defines any actions or events that initiate the Scenario. It is the interaction between a Stakeholder and the system itself.

**THEN (defines the result or outcome)**  
Finally, the THEN clause defines the conditions that determine whether or not the test is successful. If the conditions in this clause are met, the software works correctly; otherwise, it fails. This outcome can be a calculated value or any verifiable result.





**Software testing methods are the different techniques and strategies used to evaluate the functionality, quality, and performance of software products. These methods are used to detect defects and errors in software products and ensure that they meet the specified requirements and standards. These methods include black box, white box, and grey box testing.**

**Black box testing** (functional or specification-based testing) focuses on the software behavior and validation of functional requirements. Testers only check that the software does what it's supposed to do. Coding knowledge isn't necessary, and testers work at the user-interface level.

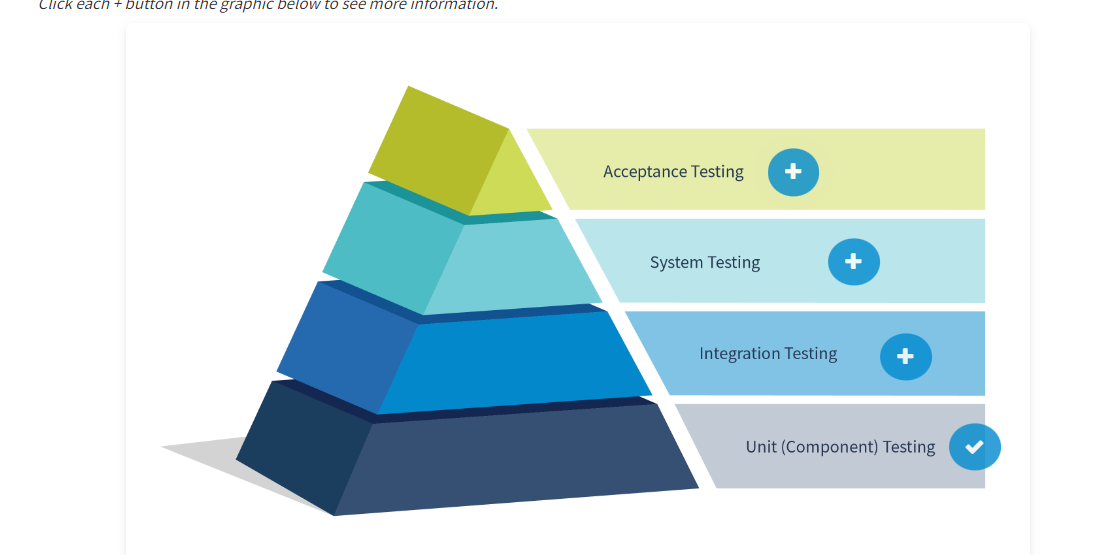
**White box testing** (also called structural testing) focuses on analyzing the internal structure of a component or system. This method uses coding experience as part of the test procedure. Testers go deep into the code when a product fails to find the cause. Most often, white box testing is performed by developers.

**Grey box testing** is a combination of white box and black box testing. Grey box testing focuses on software testing with a partial knowledge of its internal structure. To perform it, there is no need for a tester to access the source code. Tests are based on a knowledge of the algorithm, architecture, internal states, or other high-level descriptions of program behavior.

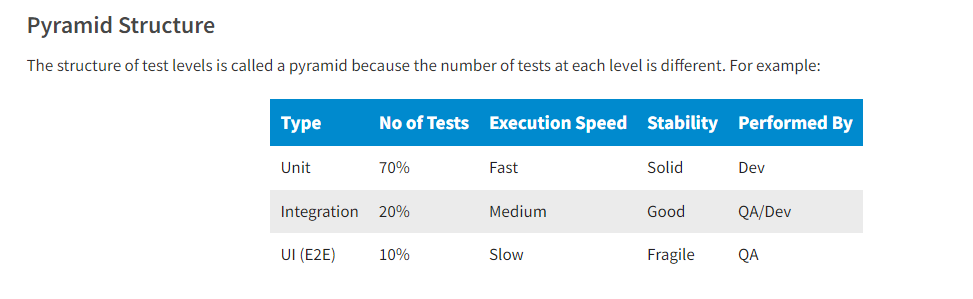
**Verification** evaluates a software product at various stages of the development cycle and involves reviewing and analyzing the product's design, code, and documentation.

**Validation** involves testing a software product against a set of test cases to ensure it behaves as expected and meets the customer's needs.

**A critical part of the validation process is user acceptance testing (UAT)** a testing method in which the end users or customers test the software product to ensure it meets their needs and expectations. UAT aims to validate that a software product meets the specified requirements and is fit for its intended use.

****

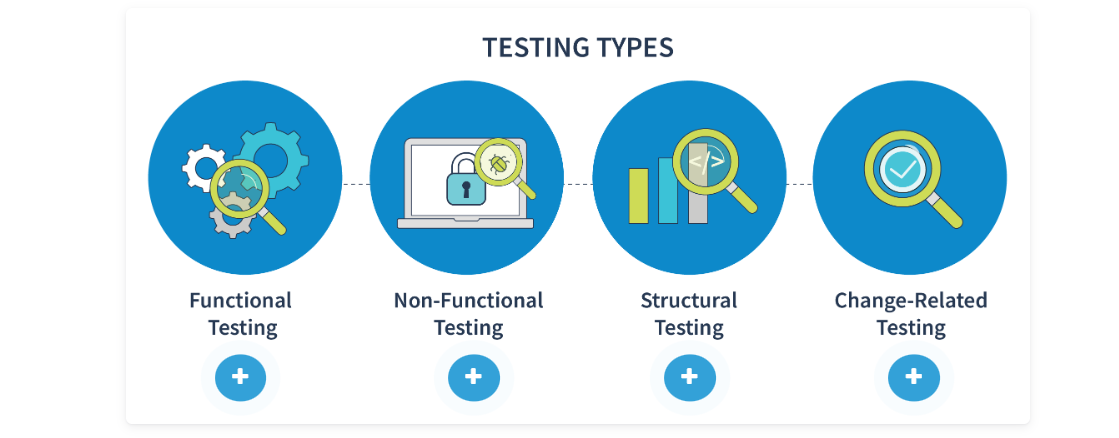
* Alpha Testing is performed by the Testers within the organization whereas Beta Testing is performed by the end users.

****

The main **software testing objectives** evaluate functional, non-functional, structural, or architectural characteristics of software or the effects of changes in software.

#### Software Testing Types

A **testing type** is a group of test activities aimed at testing specific characteristics of a software system (or part of it) based on specific test objectives. Each type involves certain procedures and actions that a tester performs to check the quality of software. The four most common types of testing are:

****

**Functional Testing**

Functional testing evaluates the functions a system should perform. It tests WHAT a system should do.

**Non-Functional Testing**

Non-functional testing evaluates aspects of software such as usability, performance, reliability, and security. It tests HOW WELL a system behaves.

**Structural Testing**

Structural testing evaluates a system's internal structure (code, architecture, workflows, and data) or how it is implemented.

**Change-Related Testing**

Change-related testing evaluates whether changes corrected a defect or whether a functionality was implemented without causing unforeseen adverse consequences.

**Smoke Testing**

**Smoke Testing** is a software testing process that determines whether the deployed software build is stable or not.

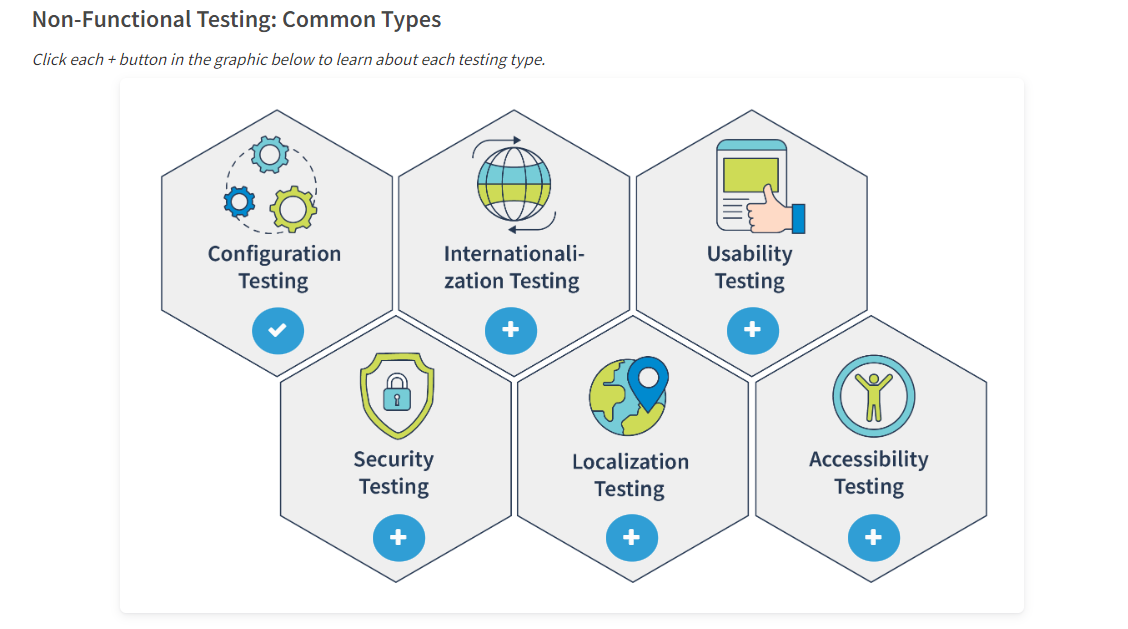
## Functional Testing

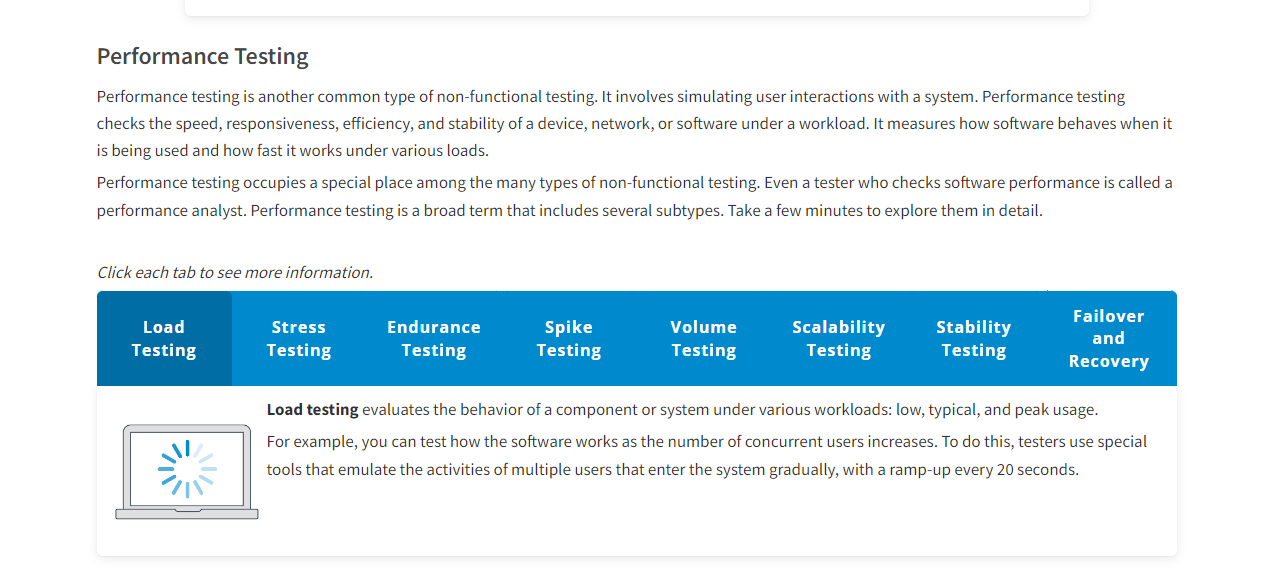
**Positive** testing, also known as "happy path testing," checks whether a software system behaves as expected in normal, expected scenarios. It evaluates whether the system produces the desired results when it receives valid inputs and follows the expected workflows.

**Negative** testing evaluates whether a software system can handle unexpected or invalid inputs and scenarios. It checks whether the system produces the correct error messages and fails gracefully and makes sure it does not produce unintended results.

**Non-functional**

Non-functional testing checks the quality characteristics of an entire system or of individual components of a system.

****

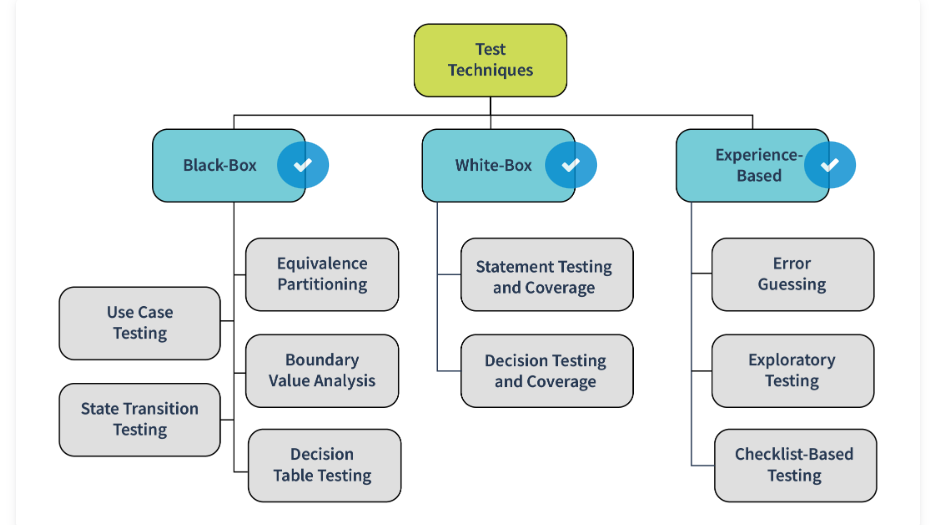
****

## Test Plan

A test plan is a document that outlines the approach, objectives, scope, and schedule of testing activities for a particular software project or product

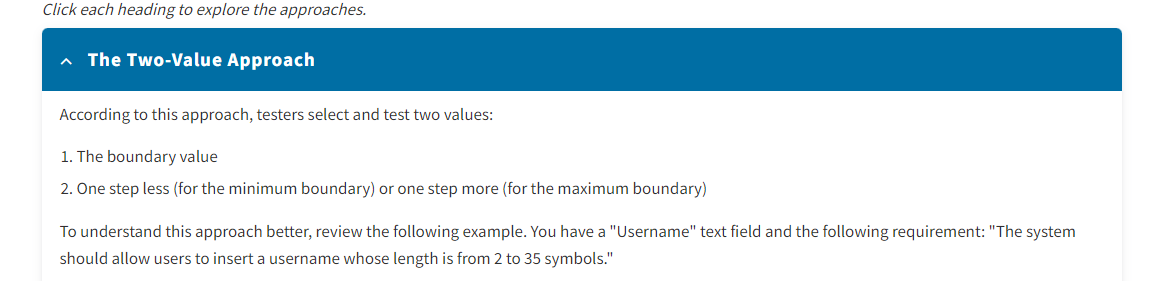
A **checklist** is a high-level list of items to be noted, checked, or remembered or a set of rules or criteria against which a product is verified

**Test techniques**

****

#### **Equivalence Partitioning -** The equivalent partitioning technique will help you reduce the number of required tests, as you only need to pick a few sample values from each equivalence class to write test documentation (checklists or test cases) and conduct a test without the risk of lowering product quality

#### **Boundary Value Analysis -**

****

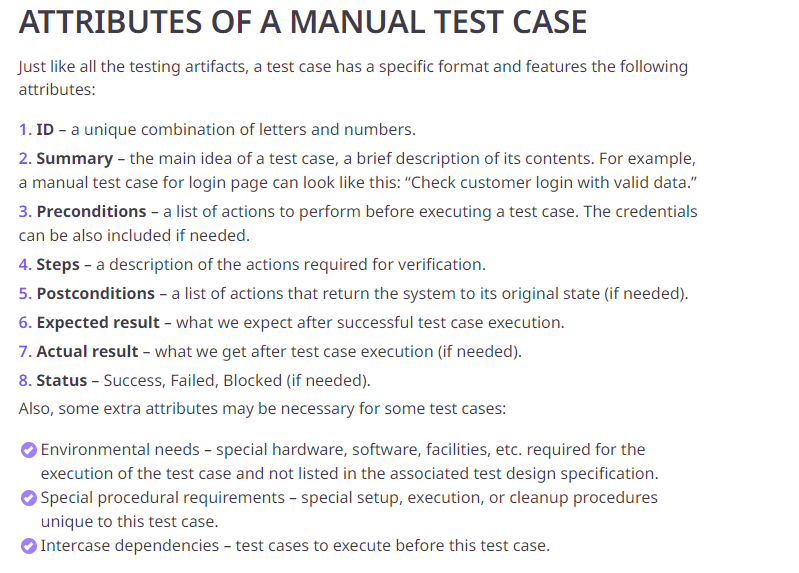
#### **Experience-Based Test Techniques**

Experience-based test techniques are a type of software testing technique that relies on the knowledge, skills, and experience of testers to identify defects and potential issues in a software application.

#### TEST CASE

The main purpose of a test case is to ensure that different features within an application work as expected.

A **test case** is a set of preconditions, inputs, actions (where applicable), expected results, and postconditions developed based on test conditions

****

**TEST SUITE**

A **test suite** is a set of test scripts or procedures to be executed in a specific test run. Test suites help categorize test cases into groups and arrange them systematically. These collections typically contain test cases that are associated with each other or grouped under multiple sub-suites.

Defect

 A**defect** is an imperfection or deficiency in a working product that causes it not to meet requirements or specifications

### Priority – наскільки бистро над фіксанути Priority, as the name suggests, is about prioritizing a defect based on business needs and the severity of the defect. Priority signifies the importance or urgency of fixing a defect.

Severity – Наскільки серйозна проблема тіпа важка і тд

Severity defines the extent to which a particular defect could create an impact on the application or system.

Defect workflow refers to the process of identifying, documenting, tracking, and resolving defects or issues in a software development project. It involves a systematic approach to managing defects and ensuring they are fixed promptly and efficiently, minimizing their impact on project timelines and budgets.

A defect tracking system (DTS) is an essential tool for any software development team committed to delivering high-quality software that meets the needs of end users.

A **test result report** (TRR) is a document that summarizes the results of testing and provides enough information to compare the current situation with the test plan and make the necessary managerial decisions.