In software development, [a test plan](https://istqb-glossary.page/test-plan/) defines your testing team’s test strategy, goals, and scope, which ultimately work together to ensure that all your software components are tested sufficiently before a release.

Follow these six steps to create an efficient test plan:

1. Define the release scope
2. Schedule timelines
3. Define test objectives
4. Determine test deliverables
5. Design the test strategy
6. Plan test environment and test data

How to create a test plan

### **1. Define the release scope**

Before any test activity occurs, it’s important to define the scope of testing for your release.

Examples of questions to ask when defining the release scope include:

* Are there new features being released in this version?
* What are the risk areas?
* What type of release is it?
* Is this a maintenance release that includes bug fixes?
* Is this a minor feature release?
* Is this a major feature release?

### **2. Schedule timelines**

Specify release deadlines to help you decide your testing time and routine. Here are some points for determining timelines:

* Consult your project manager to understand the current release timeline.
* Look at past release times and schedules.
* Consider the timeframes for development

### **3. Define test objectives**

[A test objective](https://istqb-glossary.page/test-objective/)is a reason or purpose for designing and executing a test.

Examples of general test objectives include:

* Identifying and reporting defects
* Testing new features
* A certain level of test coverage

### **4. Determine test deliverables**

Test deliverables are the products of testing that help track testing progress.

Deliverables should meet your project’s and client’s needs.

 Here are important deliverables to focus on before, during, and after testing:

#### **Before testing**

* **Test plan document:** The scope, objectives, and approach of the testing are all outlined in the test plan.
* **Test suite:**Test cases illustrate how to run a test, including input data, expected output, and pass/fail criteria.
* **Test design and environment specifications:** The test environment outlines the hardware and software configurations used for testing.

#### **During testing**

* **Test log:** The test log records each test case’s results, including issues and resolutions.
* **Defect report:** A defect report lists testing issues by severity, priority, and reproducibility.
* **Test data:** According to the International Software Testing Qualifications Board ([ISTQB](https://glossary.istqb.org/en_US/term/test-data-1-3)), test data is data created or selected to satisfy the execution preconditions and input content required to execute one or more test cases.
* **Test summary report:**The test summary report lists the number of tests run, passed, and failed, as well as open defects.

#### **After testing**

* **Test completion report:** Covers the testing scope, product quality, and lessons discovered.
* **User acceptance test (UAT) report:**Points to any issues found and fixed.
* **Release notes:** List information about what the release includes. Examples include any new features for development, advancements, or fixes.

### **5. Design the test strategy**

Test strategy helps determine test cost, test effort, and which features will be in-scope (planned to be tested) versus out-of-scope (not planned to be tested).

Examples of factors to consider when choosing the right testing type to perform include:

* Test objectives
* Your project’s feature requirements
* The complexity of your product
* Your team’s experience levels
* Regulatory requirements
* Time and budget

### **6. Plan the test environment and test data**

Planning a test environment guarantees precise and robust testing. The test environment includes hardware, software, and network configurations for software testing. Follow these procedures to set up the test environment:

* **Determine your hardware and program requirements:**Select test environment devices and software, including operating systems, browsers, databases, and testing tools.
* **Install the required software:**Once prerequisites are established, install the necessary tools on the test environment.
* **Configure the network:**Make sure that firewall protocols, IP addresses, and DNS settings, among other network configurations, are identical between the test and production environments.
* [**Create the test data:**](https://www.datprof.com/solutions/what-is-test-data/#:~:text=Data%20can%20be%20created%201,from%20an%20existing%20production%20environment.)  Test data can be created manually with data from the production environment, retrieved from an existing production environment and database, or, created via automated Data Generation Tools.
* **Access the builds:**Ensure that the builds that the testers will be testing are accessible. One example is setting up a file-sharing or version control system to allow testers access to the most current builds.
* **Verify the test environment:**After setting it up, check that your test environment fulfils the requirements.

## **What is a Test Case?**

A **test case template** is a document that comes under one of the [test artifacts](https://www.softwaretestingmaterial.com/test-deliverables/), which allows testers to develop the test cases for a particular test scenario in order to verify whether the features of an application are working as intended or not.

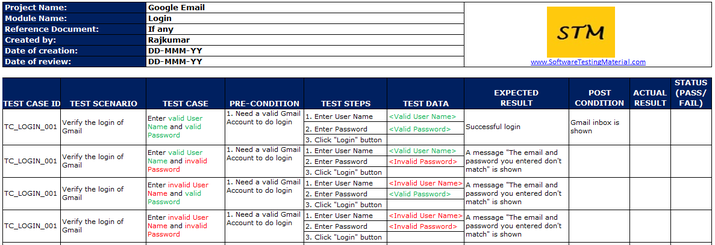
Test cases are the set of positive and negative executable steps of a test scenario which has a set of pre-conditions, test data, expected results, post-conditions, and actual results.

## **Who Writes Test Cases?**

It varies from company to company. Assuming that a team contains developers and dedicated testers then it might be something like a joint effort.

* Developers write Unit Tests
* Developers & Testers write Integration Tests
* Testers write Acceptance Tests

## **General Test Case Template Format**

Find the test case template screenshot below:

## **Popular Test Case Management Tools**

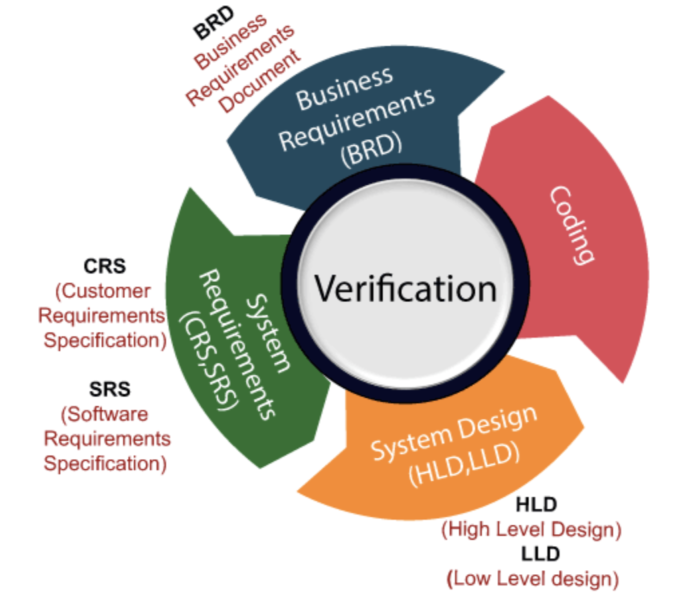
some of the popular tools for the test management process are

1. PractiTest
2. Test Rail
3. Testpad
4. Qase
5. Klaros
6. Test Collab
7. QMetry
8. Meliora Testlab
9. TestLodge
10. TestCaseLab

# **Verification and Validation in Software Testing**

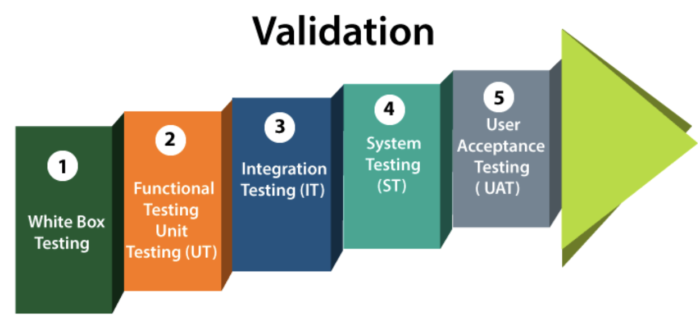
* **Verification** evaluates software artifacts (such as requirements, design, code, etc.) to ensure they meet the specified requirements and standards. It ensures the software is built according to the needs and design specifications.
* **Validation**evaluates software to meet the user’s needs and requirements. It ensures the software fits its intended purpose and meets the user’s expectations.

### **What is Verification Testing?**



* **Code reviews, walkthroughs, inspections, design, and specification analysis** are common components of verification testing.
* Some important documents to be reviewed in this stage are **requirements specification, design blueprints, ER diagrams, database table design,**[test cases](https://www.browserstack.com/guide/writing-good-test-cases)**,**[test scenarios](https://www.browserstack.com/guide/how-to-create-test-scenarios)**,**[traceability matrix](https://www.browserstack.com/guide/importance-of-traceability-matrix-in-testing)**,** etc.
* Verification tests ensure that all development elements (**software, hardware, documentation, and human resources**) adhere to organizational and team-specific standards and protocols.
* It checks to authenticate that the **system design and architecture** are accurately engineered and error-free.

### **What is Validation Testing?**



Unlike verification testing, which occurs at every stage in development, validation testing occurs at the end of a specific module or even after the software has been entirely built. Its primary intent is to ensure the final product matches the stakeholder and customer requirements.

* Most forms of QA fall under this category. All tests, from unit tests to User Acceptance Tests, are Validation tests. Some of the standard tests under this heading:
* [Unit Testing](https://www.browserstack.com/guide/unit-testing-a-detailed-guide)
* [Integration Testing](https://www.browserstack.com/guide/integration-testing)
* [Smoke Testing](https://www.browserstack.com/guide/smoke-testing)
* [Functional Testing](https://www.browserstack.com/guide/functional-testing)

All validation tests ensure that a system works as planned by running all its functions and tracking tangible, quantifiable results.

### **Difference between Verification and Validation in Software engineering**

|  |  |
| --- | --- |
| **Verification Testing** | **Validation Testing** |
| It is the static practice of studying and verifying the specific requirements of a particular stage in development. | It is the dynamic practice of testing the final product after development to ensure it meets customer requirements. |
| It does not require executing code. | It always requires executing code. |
| This involves only human verification of required assets. | This involves both human and machine-based checking and approval of software. |
| It is meant to detect bugs at the beginning of each development phase. | It is meant to detect all unnoticed bugs at the verification stage. |
| It comes before validation testing. | It follows verification testing. |
| It does not require any devices, platforms, browsers, or operating systems for its execution. | It is best executed by using real browsers, devices, and operating systems. |