Test Plan

In software development, a test plan defines your testing team's test strategy, goals, and scope, which ultimately work together to ensure that all your software components are testes sufficiently before a release.

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. This means defining the features or functions that need to be included in the release, considering any constraints and dependencies that can affect the release, and determining what type of release it is.

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?
- Are there any particularly sticky areas where you've seen regressions in the past?
- 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?
- What does being "done" actually look like for your team?

For example, what information would you require if your organization has just launched a new e-commerce site and wants to test it before it launches?

Whether it's talking with the developers to understand the scope of the project or working with a product manager to walkthrough new functionalities and user flow, defining the scope ensures that accurate information is being shared and that there is a common understanding of the product's goals, expectations, and features.

2. Schedule timelines

Specify release deadlines to help you decide your testing time and routine. Here are some pointers 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: Your development team might have a set schedule for finishing development work. Make sure you comprehend that timeframe so you can adjust the testing schedule.
- Add some extra wiggle room: It's common to encounter unexpected delays. Including extra time for unforeseen events can help you stick to your plan.
- Review and update the schedule frequently to ensure the test timetable is attainable.

3. Define test objectives

A test objective is a reason or purpose for designing and executing a test. These objectives ultimately help guide and define the scope of testing activities.

Examples of general test objectives include:

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

Examples of objectives for specific types of testing include:

- **Functional testing objectives:** Ensure the software works as it should. Examples of goals for this objective include: Validating user workflows, data processing, and verifying input/output parameters.
- **Performance testing objectives:** Ensure the software is efficient and can handle various loads. Examples of goals for this objective include: Verifying software reaction time, throughput, and scalability.
- **Security testing objectives:** Uncover program security flaws. Examples of goals for this objective include: Verifying authentication and authorization features and identifying potential threats.
- **Usability testing objectives:** Concentrate on ease of use and user experience. Examples of goals for this objective include: Validating software accessibility, verifying user flow, and identifying user-related issues.

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, be identified early enough to be included in the test plan, and be scheduled accordingly. There are different test deliverables at every phase of the software development lifecycle. 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 endeavor 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), 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).

Identify testing types

It is critical to identify when to perform what type of testing, what should be tested manually vs. automated, the scope of automated tests, how much work will be required to create new test cases, and who will be doing that work.

Depending on several factors, there may be various types of testing to include in your test plan.

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

Here are commonly used types of testing to consider including in your test plan:

Manual Testing	Automated Testing	Other
•Smoke testing •Exploratory testing •Usability testing of new features	•Unit testing •Regression testing for existing features •Integration testing	•Performance testing •Security testing •Accessibility testing

Document risks and issues

It's essential to document risks that may occur during testing and the effect of these risks. Risks can include:

- Strict deadlines
- Insufficient or inaccurate budget estimates
- Poor management
- Problems with the code
- Changes in the business environment
- Limited resources for testing
- Unexpected delays during testing

Document test logistics

Test logistics should answer the “Who, what, where, when, and how.” Documenting test logistics ensures that all human and system-related testing resources are available. For example, it may be important that your team identifies who is available to do testing and who will support them if needed during testing. Moreover, when resource planning, it can be helpful to identify alternative resources or build slack into your plan to ensure your project gets completed.

Establish test criteria

Test Criteria is a standard that regulates all activities within a testing project. The two main types of test criteria include suspension and exit criteria.

- **Suspension Criteria:** Establishes the conditions for suspending all tests.
- **Exit Criteria:** Exit criteria are established items or goals to complete that define the end of a test phase. The exit criteria of a test are the predetermined results that must be achieved to move on to the next testing phase. For example, 92% of all critical test cases must pass before a feature can be deemed suitable for release to your customers.

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. This may

require setting up a separate server to host the application and installing a database management system or other tools.

- **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:** Prepare the test material for the application's testing. 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.