,

|  |
| --- |
|  |
| SOFTWARE TESTING PLAN |

NAME: SUBHIKSHA.A

BRANCH: CSD ‘B’

REG NO.111722107051

TEAM-3

TABLE OF CONTENTS

1.INTRODUCTION

1.1 Purpose

1.2 Importance of test plan

1.3 Project objective

2.OVERALL DESCRIPTION

2.1 Product perspective

2.2 User class and characteristics

2.3 Test case

3.SYSTEM FEATURES

3.1 How to write a test plan

4.FUNCTIONAL REQUIREMENT

4.1 Resource planning

4.2 Test environment

4.3 Schedule and estimation

4.4 Test automation

5.NON-FUNCTIONAL REQUIREMENTS

5.1 Risks and issues

5.2 Problem scenario

5.3 Identifying test type

5.4 Exit criteria

**CONCLUSION**

1.INTRODUCTION

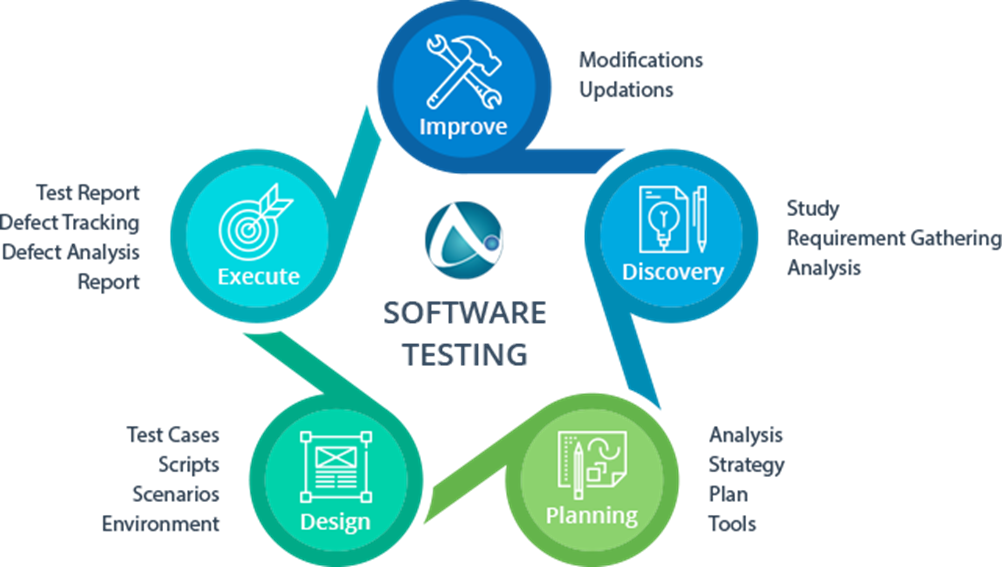
1.1 PURPOSE

When creating new software, it's important to put it through rigorous testing. This improves IT security and helps to identify any problems. As software developers, the aim is to make the process of testing as painless as possible. This ensures the software can be taken to market quickly. The more complex the software, however, the longer it takes to test. This article will take a look at the definition, objectives, and how to create a test plan.

1.2 IMPORTANCE OF TEST PLAN

A test plan is the foundation of every testing effort. It helps set out how the software will be checked, what specifically will be tested, and who will be performing the test. By creating a clear test plan all team members can follow, everyone can work together effectively.

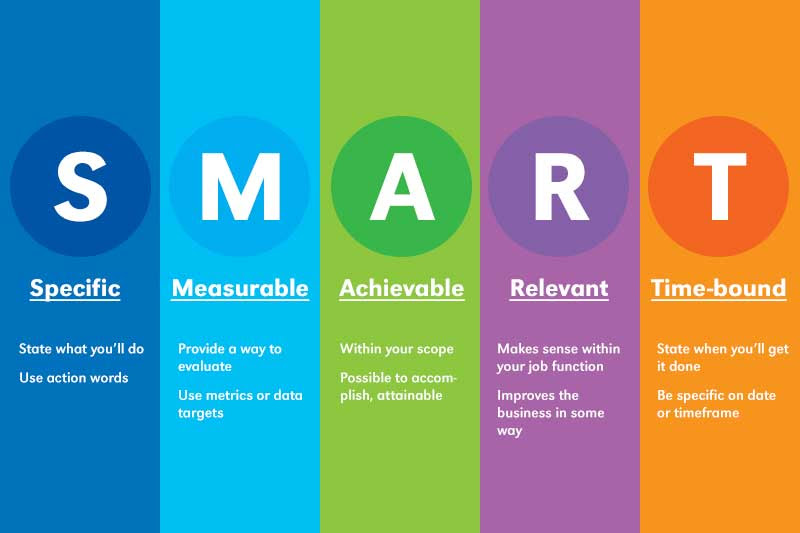
Whether you’re building an app or developing open-source software, a test plan is essential to delivering the final result.

A high-quality plan helps to identify risk areas, determine the order of testing activities, and allocate resources efficiently. The test plan becomes a useful reference document that can be referred back to throughout the product's development cycle.

1.3 PROJECT OBJECTIVE

The primary objective for a test plan is to produce documentation that describes how the tester will verify that the system works as intended. The document should describe what needs to be tested, how it will be tested, and who’s responsible for doing so.

By writing up a test plan, all team members can work in union and communicate their roles to one another. You should consider creating some SMART objectives for your test plan.



2.OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

A test plan is a document that sets out the scope, approach, and schedule of intended testing activities. The test plan may also list the resources the software tester needs to function effectively. The test plan usually includes the following information:

1. The overall objective of the testing effort.  
2. A detailed outline of how testing will be conducted (the test approach).  
3. The features, applications, or components to be tested.  
4. Detailed scheduling and resource allocation plans for testers and developers throughout all stages of testing.

2.2 USER CLASS AND CHARACTERISTICS

Before you begin creating your test plan, you’ll need to identify your intended consumers and make sure their needs are being met. This will improve the quality of your test plan tenfold.Here are the main things to ensure your test plan is:

* Concise. Your test plan should be no longer than one page with bullet points.
* Organized. Make sure all the information is logically grouped.
* Readable. The document should be easy to read, avoiding technical language where possible.
* Flexible. Your test plan should be adaptable and not fixed in stone. You want to create documentation that won't hold you back if new information comes along or changes need to be made.
* Accurate. Make sure all the information you've included is accurate.

2.3 TEST CASE

A test case is documentation created by the software tester that contains detailed information on what the test should accomplish. It’s an essential part of recording information about testing activities and results. Test cases are used in conjunction with test plans. A test case should include the following information.

1. A unique name or number to identify it.  
2. The features, applications, or components covered by the test case.  
3. Specific data values required for input fields and button controls to be tested.  
4. The predicted results of actions taken during testing (the expected outcome).  
5. A description of the actual results following each action taken during testing (the actual outcome).  
6. An indication of whether or not the test case was successful.  
7. Any errors discovered.

3.SYSTEM FEATURES

3.1 HOW TO WRITE A TEST PLAN

This might be the first job on your [software developer CV](https://www.wearedevelopers.com/magazine/quick-guide-how-to-write-a-software-developer-cv), and if that’s the case, you may need a cheat sheet to successfully write your initial test plan. Luckily, we have you covered. This section will provide you with 14 essential things to include in your software test plan as part of the [QA process](https://www.globalapptesting.com/blog/qa-process).

* Learn about the software

Before testing starts, it's important to learn everything you can about the software. Ask questions about how it was developed to learn about its intended purpose, how it works, and to garner information that might help you understand its functionality. By understanding your software properly, you can create test cases that are relevant and useful for testing your product.

* Define the scope of testing

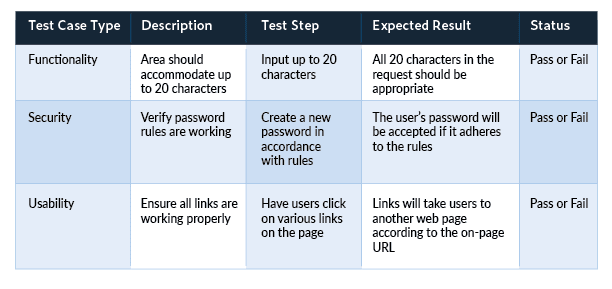
There’s no point in creating testing documents that are longer than the product itself. Before anything else, establish what exactly will be tested during the process, which modules or functions need to be covered in detail, and any other essential aspects you should know about.

* Create test cases

One of the main tasks when developing a software testing document is creating test cases. A test case is a document that describes the steps taken to carry out your testing.

It should include information such as:

* What needs to be tested
* How it will be tested
* Who will do the testing
* Expected results

Here’s a simple spreadsheet for setting up test cases:

* Develop a test strategy

The test strategy defines how you plan to implement testing. Your testers should all be working off the same game plan, so make sure every member of the team is aware of what they're supposed to be doing at any given time.

* Define the test objective

Each test case should be linked to a test objective. The objective ensures every action is relevant and contributes toward making your software more valuable for customers. Test objectives can include things like:

* Testing known features
* Testing newly implemented features
* Performing exploratory tests
* Ensuring stability throughout the product lifecycle
* Choose testing tools

You'll need to make sure you have the right [software testing solution](https://www.globalapptesting.com/blog/software-testing) to perform your testing activities. Some of these tools might be software-based, while others may require physical resources like test machines.

It's important to choose appropriate tools for each specific job and not to rely on a one-size-fits-all solution. Leave time in your planning document for 'bug fixing' sessions. These allow you to identify problems with the software early on before they become too problematic or expensive to fix. This makes them easier and cheaper to tackle. Check out any [app security measures](https://dzone.com/articles/security-measures-for-open-source-based-apps), use every feature, and seek out what doesn't work well.

* Define your test criteria

This should be part of the test case, but it's good to break it down separately. Test criteria are essentially your objectives broken down into smaller parts. They include specific information about how each objective will be met, which helps you track your testing progress.

Suspension criteria are criteria that need to be met before testing can stop. For example, you may want to suspend testing if a certain number of bugs have been found or if the software is unable to run due to performance issues.

Exit criteria are criteria that need to be met before testing can finish. For example, the test case should finish once each objective has been met and all bugs have been resolved.

4. FUNCTIONAL REQUIREMENTS

4.1 RESOURCE PLANNING

In your software testing document, include a resource plan that lists the number of people required for the testing process. This should detail what each person's role is and any training they'll require to fulfill it effectively.



4.2 TEST ENVIRONMENT

In your test plan, include information about the environment where testing will take place, such as:

* Test hardware required for product testing.
* Sizing requirements for software and servers.
* Platforms supported by the product.
* Other essential information related to the environment that might affect your testing process.

4.3 SCHEDULE AND ESTIMATION

In the article [Test estimation](https://www.guru99.com/an-expert-view-on-test-estimation.html), you already used some techniques to estimate the effort to complete the project. Now you should include that estimation as well as the schedule to the Test Planning

In the Test Estimation phase, suppose you break out the whole project into small tasks and add the estimation for each task as below

|  |  |  |
| --- | --- | --- |
| Task | Members | Estimate effort |
| Create the test specification | Test Designer | 170 man- hour |
| Perform Test Execution | Tester, Test Administrator | 80 man- hour |
| Test Report | Tester | 10 man- hour |
| Test Delivery |  | 20 man- hour |
| Total |  | 280 man- hour |

4.4 TEST AUTOMATION

If your software is particularly complex and requires a vast number of test cases, you may want to consider [software test automation](https://www.globalapptesting.com/blog/everything-there-is-to-know-about-automated-testing).

5.NON-FUNCTIONAL REQUIREMENTS

5.1 RISKS AND ISSUES

Risk is future’s uncertain event with a probability of occurrence and a potential for loss. When the risk actually happens, it becomes the ‘issue’.

In the article [Risk Analysis and Solution](https://www.guru99.com/how-precaution-becomes-cure-risk-analysis-and-solutions-in-test-management.html), you have already learned about the ‘Risk’ analysis in detail and identified potential risks in the project.

In the QA Test Plan, you will document those risks

|  |  |
| --- | --- |
| Risk | Mitigation |
| Team member lack the required skills for website testing. | Plan training course to skill up your members |
| The project schedule is too tight; it’s hard to complete this project on time | Set Test Priority for each of the test activity. |
| Test Manager has poor management skill | Plan leadership training for manager |
| A lack of cooperation negatively affects your employees’ productivity | Encourage each team member in his task, and inspire them to greater efforts. |
| Wrong budget estimate and cost overruns | Establish the scope before beginning work, pay a lot of attention to project planning and constantly track and measure the progress |

5.2 PROBLEM SCENARIO

The customer wants you to test his API. But the project budget does not permit to do so. In such a case what will you do?

Well, in such case you need to convince the customer that [Api Testing](https://www.guru99.com/api-testing.html) is extra work and will consume significant resources. Give him data supporting your facts. Tell him if Api Testing is included in-scope the budget will increase by XYZ amount.

The customer agrees and accordingly the new scopes, out of scope items are

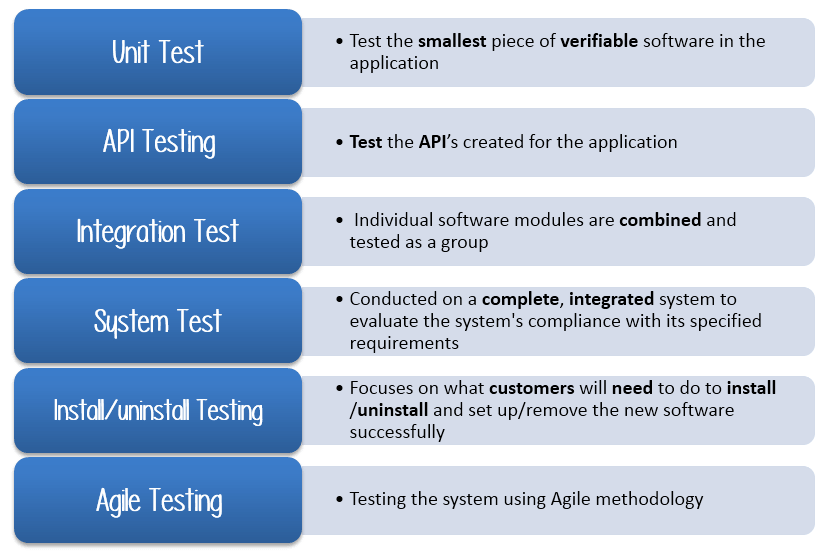
* In-scope items:[Functional Testing](https://www.guru99.com/functional-testing.html), Api Testing
* Out of scope items: [Database Testing](https://www.guru99.com/data-testing.html), hardware & any other external interfaces

5.3 IDENTIFYING TEST TYPE

A Testing Type is a standard test procedure that gives an expected test outcome.

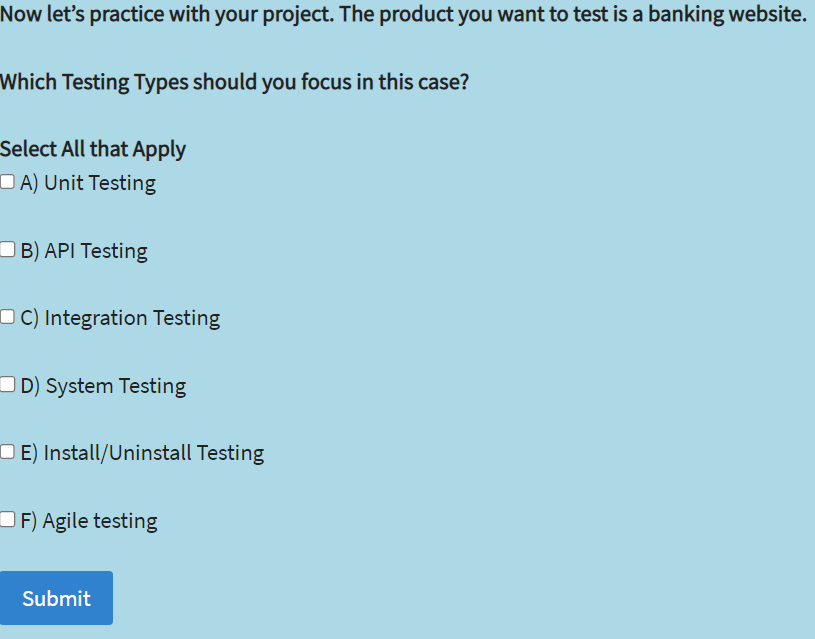
Each testing type is formulated to identify a specific type of product bugs. But, all Testing Types are aimed at achieving one common goal “Early detection of all the defects before releasing the product to the customer”.

The commonly used testing types are described as following figure



There are tons of Testing Types for testing software product. Your team cannot have enough efforts to handle all kind of testing. As Test Manager, you must set priority of the Testing Types

* Which Testing Types should be focused for web application testing?
* Which Testing Types should be ignored for saving cost?



5.4 EXIT CRITERIA

It specifies the criteria that denote a successful completion of a test phase. The exit criteria are the targeted results of the test and are necessary before proceeding to the next phase of development. Example: 95% of all critical test cases must pass.

Some methods of defining exit criteria are by specifying a targeted run rate and pass rate.

* Run rate is ratio between number test cases executed/total test cases of test specification. For example, the test specification has total 120 TCs, but the tester only executed 100 TCs, So the run rate is 100/120 = 0.83 (83%)
* Pass rate is ratio between numbers test cases passed / test cases executed. For example, in above 100 TCs executed, there’re 80 TCs that passed, so the pass rate is 80/100 = 0.8 (80%)

This data can be retrieved in Test Metric documents.

* Run rate is mandatory to be 100% unless a clear reason is given.
* Pass rate is dependent on project scope, but achieving high pass rate is a goal.

Test Plan Example: Your Team has already done the test executions. They report the test result to you, and they want you to confirm the Exit Criteria.



In above case, the Run rate is mandatory is 100%, but the test team only completed 90% of test cases. It means the Run rate is not satisfied, so do NOT confirm the Exit Criteria.

CONCLUSION

A solid test plan is an important part of [dev project briefing](https://www.wearedevelopers.com/magazine/the-importance-of-a-good-briefing-for-every-dev-project). Your testing document should be transparent, concise, and flexible, and adapt to changes in your schedule or environment.

This ensures everyone on your team is working toward the same goal and that nothing gets missed along the way.‍