NOTE ON USING THIS TEMPLATE:

In this template, text that is coloured green is for guidance only. This text uses the “Normal Green” style. To revert back to the more usual black-coloured text select the “Normal” style from the “Styles and Formatting” option under the “Format” menu. Type over with your own text or delete. A table of contents has not been included since my experience shows unless the document is more than, say, 20 pages, then it adds little value.

This template is free to use and change and has been provided by www.The-Software-Tester.com.

SOFTWARE TEST PLAN:

Project name

Approvals:

<table>
<thead>
<tr>
<th>Approved By:</th>
<th>Signature</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approver name 1</td>
<td>Signature 1</td>
<td>Date 1</td>
</tr>
</tbody>
</table>

Document Control

<table>
<thead>
<tr>
<th>Name</th>
<th>Document full name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doc. Ref. No.</td>
<td>Unique reference number</td>
</tr>
<tr>
<td>Document Status</td>
<td>Draft, For Approval, Issued, Superseded, etc.</td>
</tr>
<tr>
<td>Date of Issue</td>
<td>Date the document was approved for issue</td>
</tr>
</tbody>
</table>

Change History

<table>
<thead>
<tr>
<th>Doc. Version</th>
<th>Author</th>
<th>Date</th>
<th>Description / Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document version identifier</td>
<td>Author name</td>
<td>Date</td>
<td>Summary of content or changes. Include reference to software change request IDs if applicable for traceability purposes.</td>
</tr>
</tbody>
</table>

Distribution List

<table>
<thead>
<tr>
<th>Name</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person or company name</td>
<td>Role description</td>
</tr>
</tbody>
</table>
1 Introduction

This is essentially the executive summary part of the plan. State the purpose of this Software Test Plan. If it links in with other plans (for example, Project Plan or Master Test Plan) then identify the level to which this plan belongs. You may want to include any references to other plans, documents or items that contain information relevant to this project. If desired, use the References section (see paragraph 4 below) to list all your referenced documents.

2 Scope

Identify the scope of this Software Test Plan in relation to the overall project plan that it relates to. Other items to consider in relation to the scope may include resource and budget constraints, other test phases by other teams, constraints on the test team and how testing relates to other evaluation activities (for example, quality audits or process assessments). If special change control processes or communication plans have to be used then cover this here too.

3 Test Plan Identifier and Document Change Control

The Test Plan Identifier is just a type of unique number or reference-id to identify this test plan and the software that it is related to. If you work for a medium to large size company then there will probably be a document numbering system already in use and you will use that. In this template, the Test Plan Identifier is the same as the “Doc. Ref. No.” on the title page and in the page headers. Remember that there can be many draft and published versions of this document, so version history is essential if the changes in this document are to be traceable to the changes in the software under test. The title page makes provision for version and change history. In order to support bi-directional traceability, you should include the software change request ID numbers that have mandated these updates and changes. Note also that the footer of this template includes a Date-Time Stamp. This is the last saved date and time of the document. If you are using a document management system then this is a useful way to tell if a printed copy of the document reflects the latest version or not.

4 References

List all documents that support this test plan. Refer to the actual version/release number of the document as stored in the document or configuration management system. Try and include hyperlinks if possible to aid access by your readers. Avoid repeating material from other documents
since it adds duplication and increases the maintenance overhead. Documents that could be referenced include:

- Project Plan
- Requirements specifications
- Architecture specifications
- High Level design documents
- Detail design documents
- Functional specifications
- Implementation specifications
- Quality system process documents
- Corporate standards and guidelines

<table>
<thead>
<tr>
<th>Document Reference &amp; Version</th>
<th>Document Title / Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ref ID and Version</td>
<td>Document title and brief description</td>
</tr>
</tbody>
</table>

## 5 Test Items

These are the software products (code, 3rd party products, user manuals, etc.) you intend to test within the scope of this test plan. This list of items will be populated from the software products identified in the master project plan as well as other sources of documentation and information. You should include version numbers and configuration requirements where needed, (especially if multiple versions of the products are in scope). Bear in mind that what you are testing is what you intend to deliver to the customer (whether internal or external).

<table>
<thead>
<tr>
<th>Test Item Name</th>
<th>Test Item Version No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>The formal name of the test item</td>
<td>Version of the test item</td>
</tr>
</tbody>
</table>

### 5.1 Features to be Tested

This is a high-level view of what is to be tested from the user’s viewpoint of what the system does and should refrain from being a technical testing breakdown of the system since that is covered in section 7 below. It is also useful to list the features to be tested with respect to the names of the parent components, etc., as they are known by the configuration management system. A bulleted list format can serve well here, or use the table format given below.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Parent Component / System</th>
<th>Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature name</td>
<td>The component or system the feature belongs to</td>
<td>Brief outline of what is being tested from user’s perspective</td>
</tr>
</tbody>
</table>
5.2 Features not to be Tested
What is not to be tested can be sometimes just as important as stating what is to be tested. It removes any ambiguity in order that other project stakeholders are clear on what to expect from the test phases. Make this list the same format as above. Additionally, however, you should state clear reasons why the feature is not being tested. There could be any number of reasons and all should be given alongside the mitigating factors.

<table>
<thead>
<tr>
<th>Document Reference &amp; Version</th>
<th>Document Title / Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ref ID and Version</td>
<td>Document title and brief description</td>
</tr>
</tbody>
</table>

6 Testing Risk Register
Software development is full of risk, and the testing phases are no exception. It is always wise to take an active lead in managing the risks facing you. Look to the master project plan as a starting point to begin your list of risks. As a further aide to covering all the potential risks, consider the following:

- Experience of previous similar projects and the risks that did happen and how they were handled
- Third party products and services.
- New versions of interfacing or component software
- The test team’s ability to use any new tools or technologies necessary for the test effort.
- Working across multiple sites, off-shore team members, remote-working
- Any complex functionality
- Adoption of any new technologies, especially ‘bleeding edge’ technologies
- Schedule slippage and its impact on the test schedule
- Modifications to components with a past history of failure
- Poorly documented modules
- Vague requirements
- Ever-changing (volatile) requirements
- Safety aspects
- Cross platform support
- Multiple interfaces or poorly defined interfaces
- Government regulations and rules

Once you have your list of risks, offer them up to other team members and brainstorm them. Furthermore, assign a score to each risk in terms of its probability of occurrence and its impact on the customer. Also document each risk’s triggers and any mitigation action you can complete in order to prevent the risk. If the risk does occur then you can help alleviate by planning in any contingency actions. You may wish to compile your risk register in tabular format, such as the one given below.

| Risk ID No.          | Unique reference ID of the risk item |
I would recommend reviewing your testing risk register once a week. You’ll find that some entries are so insignificant that they can effectively be ignored. Other new ones may be added as they are identified. Other risks may be revised in terms of significance as more becomes known about them.

In both creating and reviewing risks it is well worthwhile teaming with other testers and developers since they will often be vocal in expressing their own personal key concerns about the project.

7 Test Approach (Strategy)

The test approach is the overall test strategy that underpins the whole test plan. A test approach asks, “how are you going to test the software?” If this Test Plan is part of a larger parent project and there are other Test Plans for other parts of the overall system then the test approach should dovetail with the other test approaches. Also consider if you are going to use established, documented test processes or procedures or if you will need to tailor your own specifically for this project. If so, then name these documents and include them in the reference section above.

Other questions to ask in devising the test approach include:

- What type of testing will be done when?
- Will you start by running tests on the most risky area of the software?
- Are there parts of the planned functionality that will be delivered after other parts and therefore require you to ‘stage’ your testing?
- Is there a ‘must have, should have, could have’ approach to the priority of new functionality and if so does your test approach take this into account?
- Will you use predominantly requirements-based manual test scripts or will you make use of rapid-test techniques such as exploratory testing to get an early assessment of the stability of the software?
- What about the depth and timing of regression tests? Will regression tests be run manually or will you use automated test tools?
• Will you have testers dedicated to regression tests and others
dedicated to running tests on new functionality?
• Will any parts of the test regime be executed by remote members of
the test team?
• What about non-functional testing like install/uninstall, compatibility,
load, volume and performance, etc.?
• When will such tests be run and will this impact on the more regular
functional testing?

You will see there are a lot of questions posed here and indeed there could be
many more. Only by asking such questions will you stimulate the thought
patterns that can help to assure the test coverage and approach you build is
solid, dependable, comprehensive and appropriate. Mindmaps can be useful
as a way of jotting down your ideas (see the Tools page on http://www.The-
Software-Tester.com for a useful freeware mindmap tool). Of course, a mine
of information can be obtained from previous projects and it can often be
useful to read over old test reports in order to gain insights into the planning of
new test projects. If there were any previous 'lessons to be learned' reviews
run on previous projects then obtain the related documentation and read it
over since there could be valuable lessons to be learned for your current
project.

7.1 Test Tools
List all the tools required for testing. Such tools may include obvious ones like
test management software, a defect management system, GUI automation
tools, etc., but there may be less obvious tools required like project
management tools, etc. If you have a remote contingent as part of the overall
test team then consider any tools you will need in order to effectively
communicate with them.

If commercial-off-the-shelf (COTS) tools are not suitable then you may need
to develop own tools, harnesses and stubs. This is all work that requires to be
costed, estimated and planned so include it here.

7.2 Test Data
Plan you test data needs. Is this something that you can manage within the
test team or is it something you will need to get a developer or database
administrator to do? What about automatic test data generators? If you are
using data sourced from a current live system then there may be
confidentiality aspects that need to be addressed — or you could anonymise
the data. Will the test data need to be reset for every cycle of testing? If so,
who will do this and how long will it take. If scripts have to be run to do this
they could mean lengthy runs that could impact your progress.

7.3 Test Environment
The test environment encompasses the software being tested, related
platform software, third-party software, communications software, etc. Ensure
that you have the resources required to install, set up and configure your test
environment. The test environment also extends to hardware, test data, technical publications, consumables (like printer paper, ink, etc.), swipe cards, etc. Take a ‘helicopter view’ of your test environment and plan for anything that is required in order to assure the smooth execution of your testing.

8 Personnel

List the members of your test team here. Think about the specialisms each has when assigning them work tasks. If you have senior testers then consider pairing them with more junior members. Is anyone a specialist and is there a risk if this specialism cannot be covered by others in the test team should that person become unavailable? What about holiday and planned absences. This will all need to be factored into your plans and the overall project schedule.

If you are going to require any extra testers then write down your intentions for obtaining the extra personnel. Will you source them internally on secondment or will you have to hire? Do you have the budget for this? How long will you need them for? What if the project over-runs?

Will you require specialist skills that lie outside the test team? For instance, will you require a Human Computer Interface (HCI) specialist to assess usability and accessibility for any new screens/GUIs? What about management of data and the test environment. If test hardware fails will you require lab technicians or hardware engineers to support you? Will you require a dedicated configuration management engineer?

Consider using the example table below to list your direct (and indirect) personnel requirements:

<table>
<thead>
<tr>
<th>Name</th>
<th>Role</th>
<th>Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name of person</td>
<td>E.g. Test Manager, DBA, Test Automation Engineer, etc.</td>
<td>E.g. manage integration test phase, creation of UAT test data, etc.</td>
</tr>
</tbody>
</table>

8.1 Training

Since you have reviewed the roles and responsibilities of test-related personnel it may have become apparent that there are skills gaps. If this is the case then you will need to decide the best way to fill these gaps. It may be that paired testing and mentoring between senior and junior staff is a suitable solution. However, training may sometimes need to be more formal and this becomes a separate task in its own right that will require planning, costing and incorporation into the project.

9 Management and Metrics
Who will be managing the testing? Will different people be managing different phases, for example integration test phase, component test phases? Does the Project Manager require metrics to be collected from you? If so, then list these here and state when and how you will construct the metrics and report them.

Who will be managing the different versions of software released into the testing phase(s)? Will this be the responsibility of the test team or development? Are there other test teams you need to work with? Will a dedicated configuration management team manage this? Cover such aspects here.

Will you set up regular meetings to review test progress? A 15-minute daily meeting of the test team is very useful, although a weekly meeting between the test manager and senior management may be more appropriate for this level. Document your meeting schedules and reporting lines here.

9.1 Test Estimation and Schedule
The duration of the testing schedule should have been estimated as a result of a team consensus. It can be useful to break test estimates down to a level that matches that in the related functional specifications or requirements. In this way the test estimates can be readily accounted for. A team-based approach to estimating can smooth out any ‘bumps’ in the magnitude of the estimates. You may wish to include the original test estimates in this section as a matter of record. This should be the estimates that feed into the overall project schedule. Grouping the estimates together into logical groupings can also be helpful from a management perspective, for instance, grouping into ‘test planning’, ‘test execution’, ‘test reporting’

The testing schedule is a subset of the overall project schedule. Include a hyperlink to the file containing the project schedule rather than repeat any details here. Since project schedules can be volatile and subject to continual revision it makes sense to do this and avoid unnecessary reworking of this Test Plan. The inherent risks associated with schedule slippage mean that this is an area that invariably finds its way into the testing Risk Register in paragraph 6 above.

If this project is part of a larger project, then you may wish to acknowledge any other relevant test/project schedules. Consider how this project and its schedule interface with these other projects and their schedules. For instance, are there any tasks that have to be handed over?

9.2 Test Phase Entry and Exit Criteria
In order that you can manage software stability and quality grade through successive test phases it is useful to plan for test phase entry and exit criteria. The review of such criteria should be the basis of formal management milestone reviews. Each test phase is likely to have its particular set of
criteria. The following points below may help you formulate your own list of criteria.

9.2.1 **Unit Test Phase Entry Criteria**
- 100% of unit tests have been peer-reviewed
- Software to be unit tested has been checked into configuration management system
- All planned functionality and bug fixes have been implemented
- Source code for software to be unit tested has been peer-reviewed
- Planned number of issues expected to be found in unit test has been agreed
- etc

9.2.2 **Unit Test Phase Exit Criteria**
- 100% of unit tests are executed
- 100% of unit tests pass
- Unit Test Report has been approved
- Unit tested software has been checked into configuration management system
- Unit tested software is available for next test phase
- Less than $n$ outstanding low severity issues
- Less than $n$ outstanding medium severity issues
- Less than $n$ outstanding high severity issues
- Number of issues found did not exceed planned number by more than 25%
- etc

9.2.3 **Component Test Phase Entry Criteria**
- Component test documentation/scripts have been peer-reviewed
- Software to be component tested has been checked into configuration management system
- Test data completed
- Test environment completed
- Planned number of issues expected to be found in component test has been agreed
- etc

9.2.4 **Component Test Phase Exit Criteria**
- 100% of component tests are executed
- $r\%$ of component tests pass
- Component Test Report has been approved
- Component tested software has been checked into configuration management system
- Component tested software is available for next test phase
- Less than $n$ outstanding low severity issues
9.2.5 **Integration Test Phase Entry Criteria**
- Integration test documentation/scripts have been peer-reviewed
- Software to be integration tested has been checked into configuration management system
- Test data completed
- Test environment completed
- Planned number of issues expected to be found in integration test has been agreed
- etc

9.2.6 **Integration Test Phase Exit Criteria**
- 100% of integration tests are executed
- \( n\% \) of integration tests pass
- Integration Test Report has been approved
- Integration tested software has been checked into configuration management system
- Integration tested software is available for next test phase
- Less than \( n \) outstanding low severity issues
- Less than \( n \) outstanding medium severity issues
- Less than \( n \) outstanding high severity issues
- Number of issues found did not exceed planned number by more than 25%
- etc

9.2.7 **Acceptance Test Phase Entry Criteria**
- Acceptance test documentation/scripts have been peer-reviewed
- Acceptance Testers have been trained (if doing true UAT)
- Software to be acceptance tested has been checked into configuration management system (this could include documentation and user manuals, etc.)
- Test data completed
- Test environment completed
- Planned number of issues expected to be found in component test has been agreed
- etc
9.2.8 Acceptance Test Phase Exit Criteria

- 100% of acceptance tests are executed
- 100% of acceptance tests pass
- User needs 100% validated
- Acceptance Test Report has been approved
- Acceptance tested software has been checked into configuration management system
- Customer has formally approved acceptance of the software into the live environment
- Less than \( n \) outstanding low severity issues
- Less than \( n \) outstanding medium severity issues
- Less than \( n \) outstanding high severity issues
- Number of issues found did not exceed planned number by more than 25%
- etc

9.3 Suspension and Resumption Criteria

During any point in any of the test phases it may become apparent that continuing with the planned test regime is pointless and resources better used on assignment to other tasks. Thinking ahead and anticipating such incidents can help planning and management. Suspension could arise due to:

- A critical issue blocking a significant proportion of the remaining tests and the time to resolve the issue is, say, more than a day.
- The number of issues raised exceeding planned issue levels, particularly if these are mostly high in terms of severity.
- Etc.

Resumption of testing usually commences upon removal of the blockage that caused the suspension in the first place. However, it may be that testing can resume due to other circumstances, for example, if new functionality is released to test that permits new tests to be executed.

10 Test Deliverables

Test deliverables are essentially the work products of the entire test regime. This can cover:
- Test estimates
- Test schedules
- Test Plan
- Test Specification
- Test Scripts
- Test Data
- Test tools, harnesses, stubs, etc
- Test execution results
- Test Reports
• Issue Logs
• Lessons to be learned
• Test Risk Register
• Test Training plans
• Etc.

11 Communication Plan

I have always found it useful to cover how stakeholders will communicate throughout all testing activities. It can be a good idea to include contact details of the key stakeholders in this section, for example the phone and email details of the author, test manager, etc. If you plan to hold morning stand-ups, then state that here. If it is planned to run review weekly meetings, or remote-conferences, then plan all aspects of your communication. Think about why, where, when, who and how. Some example tables are given below to help you structure this section.

<table>
<thead>
<tr>
<th>Name</th>
<th>Role</th>
<th>Contact Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jim Smith</td>
<td>Test Team Leader</td>
<td>Email: <a href="mailto:jsmith@testeremail.co.uk">jsmith@testeremail.co.uk</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Office: 01234 567 890</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mob: 07123 456 789</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fax: 01671 987 654</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Communication Aspect</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.g. Daily Test Team Meeting</td>
<td>E.g. Review immediate issues and plan tasks for day ahead.</td>
</tr>
<tr>
<td>E.g. UAT Handover Meeting</td>
<td>E.g. Once-only meeting to review test entry criteria into UAT phase. Project</td>
</tr>
</tbody>
</table>

12 Glossary

Define terms, jargon and acronyms used in this document to eliminate possible confusion and promote consistent communication.

<table>
<thead>
<tr>
<th>Term</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.g. UAT</td>
<td>E.g. User Acceptance Testing conducted by selected customers</td>
</tr>
</tbody>
</table>