SOFTWARE TEST PLAN:
Project name

Approvals:

Approved By:	Signature	Date
Approver name 1	*Signature 1*	*Date 1*

Document Control

Name	*Document full name*
Doc. Ref. No.	*Unique reference number*
Document Status	*Draft, For Approval, Issued, Superseded, etc.*
Date of Issue	*Date the document was approved for issue*

Change History

Doc. Version	Author	Date	Description / Change
Document version identifier	*Author name*	*Date*	*Summary of content or changes. Include reference to software change request IDs if applicable for traceability purposes.*

Distribution List

Name	Role
Person or company name	*Role description*

1. Introduction

(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.)

2. 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. 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 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. 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.)

3. Scope

(Identify the scope of this Software Test Plan in relation to the overall project plan that it relates to.)

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. 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)

Document Reference & Version	Document Title / Description
Ref ID and Version	*Document title and brief description*

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).)

Test Item Name	Test Item Version No.
The formal name of the test item	*Version of the test item*

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. 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.)

Feature	Parent Component / System	Overview
Feature name	*The component or system the feature belongs to*	*Brief outline of what is being tested from user's perspective*

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. You can state clear reasons why the feature is not being tested.)

Document Reference & Version	Document Title / Description
Ref ID and Version	*Document title and brief description*

6. Testing Risk Register

(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
- 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

Once you have your list of risks, 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 it 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*
Summary	*Brief description of the risk*
Probability of Occurrence	*How likely is the risk to occur? Assign a score like 1, 2, 3 or tag with Hi, Med or Lo, etc.*
Customer Impact	*How will the customer be inconvenienced should the risk happen? Assign a similar score as the row above*
Trigger	*What would cause the risk to happen*
Mitigation Action	*What can be done to prevent the risk from happening*
Contingency Action	*What can be done to alleviate the situation should the risk happen*

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?"
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?

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.)

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.)

7.2 Test Data

(Plan your test data needs. 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.)

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.)

8. Personnel

(List the members of your test team here. Think about the specialisms each has when assigning them work tasks.
Consider using the example table below to list your personnel requirements:)

Name	**Role**	**Responsibility**
Name of person	*E.g. Test Manager, DBA, Test Automation Engineer, etc.*	*E.g. manage integration test phase, creation of UAT test data, etc.*

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? 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.

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.)

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
-n% 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
-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

9.2.5 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 the 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

(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, or run review weekly meetings, or remote-conferences, then plan all aspects of your communication. Some example tables are given below to help you structure this section.)

Name	**Role**	**Contact Details**
Jim Smith	*Test Team Leader*	*Email: jsmith@testeremail.co.uk* *Office: 01234 567 890* *Mob: 07123 456 789* *Fax: 01671 987 654*

Communication Aspect	**Purpose**
E.g. Daily Test Team Meeting	*E.g. Review immediate issues and plan tasks for the day ahead.*
E.g. UAT Handover Meeting	*E.g. Once-only meeting to review test entry criteria into UAT phase. Project*

12. Glossary

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

Term	**Meaning**
E.g. UAT	*E.g. User Acceptance Testing conducted by selected customers*