SDLC - Software Testing documentation

I used the following process documenting each project in software development life cycle (SDLC). In the following steps:

1) Planning
2) Analysis
3) Design
4) Implementation
5) Testing
6) Maintenance

Requirements Traceability Matrix (RTM)
A document that maps and traces user requirements with test cases. This document captures all requirements proposed by the client and requirement traceability in a single document, delivered at the conclusion of the Software Development Life Cycle. The main purpose of the Requirement Traceability Matrix is to validate that all the requirements are checked via the test cases such that no functionality is unchecked during the software testing.

The main agenda of every tester should be to understand the client's requirements and make sure that the output product should be defect-free (BUG free). To achieve this goal, the QA analyst needs to understand the requirements thoroughly and create positive and negative test cases.

Traceability Matrix
A traceability matrix is a document that details the technical requirements for a given test scenario and its current state.
In a software development, a traceability matrix (TM) is a document, often in the form of a table, (but not always) used to assist in determining the completeness of a relationship by correlating any two baselined documents using relationship comparison. A traceability matrix is often used with high-level requirements (these often consist of marketing or user requirements) and detailed requirements of the product to matching parts of a high-level design, detailed designs and test plans and test cases.

A requirements traceability matrix may be used to check if the current requirements are being met, and to assist in the creation of a request for proposal, software requirements specification, various deliverable documents, and project plan tasks.

Common usage is to take the identifier for each of the items of one document and place them in the left column. Some documents have columns of items verses value to the project. This value indicates the mapping of the two items. Zero values indicate that no relationship exists. Large values imply that the relationship is too complex and should be simplified.

To ease the creation of traceability matrices, it is advisable to add the relationships to the source documents for both backward and forward traceability. That way, when an item is changed in one baselined document, it is easy to see what needs to be changed in the other.

Plan and conduct software testing Quality assurance (QA) and development teams need to know how to write test documentation to have full control of their software testing.

Test Documentation
Developed test documentation per the standard operation procedures (SOPs) are followed by QA engineering and the software developer who performs the software testing. These documents will outline the essential steps in the quality assurance process and ensure more effective communication between QA and the developers.

Proper documentation benefits the company to maintain honest communication to services the FDA submission about the project's status and avoid unpleasant surprises. This also supports QA and the development teams more accountable to the clients.

Creating standard test documentation helps new developers up to speed faster by providing a reference point for them to understand the system they are working on. It outlines the objectives, scope, approach and focus of a software testing effort, allowing all developers to quickly gain an understanding of the system and its requirements.

Types of test documentation

1) Test strategy – A test strategy, or a test policy, is a document that outlines the overall approach to testing a system. It is used to define the objectives of a test effort and the techniques that will be used to achieve those objectives.
2) Test plan – A test plan is a document that outlines the objectives, scope, approach and focus of a software testing effort. It is used to identify the resources, test environment, test schedule and testing activities that will be used to ensure that software is of good quality.
3) Test case – A test case is a set of conditions or variables under which a system should be tested. It is used to ensure that the software meets its specified requirements and behaves as expected.
4) Test scenarios – Test scenarios are descriptions of the steps a tester needs to take to verify the functionality of a system. They are used to identify the different paths a user might take when using a system and to ensure that each path is tested.
5) Requirements traceability matrix (RTM) – An RTM is a document that maps requirements to test cases. It helps to ensure that all requirements are tested and that any changes in requirements are tracked.
6) Test Data – Test data is the data that is used to test a system. It can include both valid and invalid data, and is used to ensure that the system behaves as expected.
7) Bug Report – A bug report is a document that describes a bug found in a system. It is used to track and document the progress of fixing the bug.

Test Execution
A test execution report is a documentation summarizes the results of a software test. The software test is usually presented in a tabular format and contains the information such as the number of test cases run and the number of test cases passed or failed and the time taken to complete the tests and all other relevant information.

Testing strategy process

The testing strategy policy is a comprehensive document that covers almost every aspect of the testing documentation.

1. Introduction

This document outlines the testing policy. This policy establishes the framework for how testing should be conducted in order to ensure that the software meets the framework for how the testing policy should be conducted in order to ensure that the software meets the specified requirements.

2. Scope

This policy applies to all software developed and tested, including both new and existing software.

3. Testing Strategy

The testing strategy should incorporate the following elements:
Testing objectives: Outline the objectives of the test efforts and the techniques that will be used to achieve those objectives.

Testing environment: Identify the resources, the test environment, the test schedule and testing activities that will be used to ensure that the software is of good quality.

Testing cases: Define the conditions and the variables under which the system should be tested.

Testing data: Create valid and invalid data to test the system.

Testing scripts: Develop a set of instructions that a software tester can follow in order to execute the test.

Requirement traceability matrix: Create a document that maps the requirements to test each case.

Bug report: Document all the issues found in the system.

4. Testing Process

The following processes should be followed when conducting the software testing:

Planning: Outline the objectives and scope of the test efforts.

Execution: Execute the tests according to the test plan and the test scripts.

Analysis: Analyze the results of the tests and document any issues found.

Reporting: Create a test report summarizing the results of the tests and any issues found.

5. **Compliance**

   All software developed and tested by [Company Name] should adhere to this testing policy. Failure to do so may result in disciplinary action.

6. **Revision**

   The policy is subject to change and may be reviewed and revised by the software team at any time.

7. **Date of Implementation**

   The policy will be effective as of the release date.

Source: Testing Scenario

The testing scenario document outlines the specific steps a tester needs to take to verify the functionality of a system. (a test scenario template is used to ensure that the tests are performed in a consistent and repeatable manner).

1. **Introduction**
   This document outlines the testing scenario. This is used to identify the different paths a user might take when using the software and to ensure that each path is tested.
2. **Testing scenario ID**

   Assign unique IDs to each testing scenario.
3. **Testing scenario steps**

   Describe each step of the testing process in detail.
4. **Expected result**

   Describe the expected result of the testing scenario.
5. **Acceptance criteria**

   List the criteria that must be met for the testing scenario to be considered successful.
6. **Date of implementation**

   This testing scenario will be effective as of the release date.

Source: Testing Case Assess

While the test scenario focuses on evaluating end-to-end functionality, the testing case assesses specific features of the application in isolation.

1. **Test case ID**

   Each testing case in a test scenario should be assigned a unique ID. To distinguish between different IDs, create and follow a consistent naming convention.
2. **Test case description**

   Specify what exactly the testing is being done at this stage of the test scenario.

3. **Pre-condition**

   If there are any pre-conditions to be met before running the test, specify them here.

4. **Test steps**

   All of the testing steps must be documented.

5. **Test data**

   Gather and document all the data needed to perform the testing successfully.

6. **Expected result**

   Explain what outcomes can be expected to achieve from the testing.

7. **Post condition**

   Define what should happen if the testing is successful.

Source: Traceability Matrix

The creation of a Traceability Matrix is the essential in the deliverables for the QA team. The traceability matrix document connects the dots between the user requirements proposed by the client to the application being tested. This document maps the test cases to the user requirements to ensure the maximum testing coverage for every required feature.

1. **Project**

   Brief product or feature description.

2. **Date**

   The date of document creation.

3. **Requirement**

   Specific project requirement.

4. **Specification**

   More information on how to meet the requirement.

5. **Requested by**

   The name of a person who requested the feature.

6. **Business need**

   Justification of the requirement.

7. **ID**

   Test case ID.

8. **Test case**

   The name of the relevant test case.

9. **Status**

   Mark if the test is successful or not.

10. **Enhancement Feature**

    A software enhancement user requested.

Source Defect Report

A defect report is created when a problem is identified during the software testing process. The defect report is the responsibility of the testers to document the details of the defects, including the steps taken to reproduce the defect, the expected result and the actual result.

The defect report may also include any screenshots or videos that may be relevant in order to accurately diagnose the issue.

1. **Defect ID**

    The name identifying the defect.

2. **Date**

    The day the defect was detected.

3. **Description**

    Details about the defect.

4. **Steps to reproduce**

    Steps to take to spot the defect.

5. **Expected result**

    How the program should have responded to the test.

6. **Actual result**

    How it actually responds.

7. **Screenshot/Video of the defect (if applicable)**

    It’s good to record a screencast to display the defect in action.

8. **Severity**

    How seriously the defect affects the product’s functionality.

9. **Priority**

    How urgent it is to address the defect.

10. **Status**

    Choose between Open, Assigned, In Progress or Closed.

Source Testing Execution Report

Write a test execution report to document the results of the software testing. The testing execution report is typically filled out after each testing cycle and it is used to track the progress of the tests, identify any issues and provide feedback on the overall process.

1. **Project name**

    The name of the entire test project.

2. **Tester**

   Who is responsible for the test.

3. **Date**

   When the test has been run.

4. **Test cycle description**

   A sequence of specific activities conducted during the testing process.

5. **Test cases executed**

   What test cases have been run during the test.

6. **Test result summary**

   Summary of the overall status of the project.

7. **Test issues/Defects found**

   Issues detected while performing the test.

8. **Test coverage**

   How much of the application has been tested.

9. **Conclusion/Recommendations**

   Confirm completion of all recommendations.

How to write testing documentation for software testing:

1. **Identify the goals and objectives of the test documentation:** Outline the purpose of the test documentation and the objectives it should achieve.
2. **Analyze the current system and the test requirements:** Analyze the system to identify any areas that need to be tested and any requirements that need to be addressed.
3. **Develop a test strategy:** Define the objectives of the test effort and the techniques that will be used to achieve those objectives.
4. **Identify the resources needed to complete the tests:** Identify the personnel, equipment and other resources that are needed to complete the tests.
5. **Develop the test cases:** Define the conditions and variables under which the system should be tested.
6. **Create the testing data:** Create valid and invalid data to test the system.
7. **Create the testing scripts:** Develop a set of instructions that a tester must follow in order to execute a test.
8. **Create the requirement traceability matrix:** Create a document that maps requirements to test cases.
9. **Create the bug report:** Document any issues found in the system.
10. **Summarize the results with the testing execution report:** Document the actual test process and all the findings.

Writing software testing documentation retains tracking of all testing activities and resolves issues easier. Keep the testing documents concise and up-to-date. This will improve the efficiency of the software testing.