HOW TO CREATE A TEST PLAN?

A test plan is crucial in software development as it outlines the testing team's strategy, goals, and scope to ensure thorough testing before release.

Steps to Create Test Plan

1. **Define Release Scope:** Identify features, constraints, dependencies, and release type.
2. **Schedule Timelines:** Set deadlines, consider past release times, account for external variables, and accommodate development schedules.
3. **Define Test Objectives:** Establish reasons or purposes for testing, guiding testing activities.
4. **Determine Test Deliverables:** Identify products of testing to track progress, meet project and client needs.
5. **Design Test Strategy:** Determine test types, document risks and issues, specify test logistics, and establish test criteria.
6. **Plan Test Environment and Data:** Ensure precise and robust testing by setting up hardware, software, network configurations, and preparing test data.

Test Metrics: Assess quality, progress, and effectiveness of testing using metrics like defect density, test coverage, defect detection efficiency, and time to market.

Test Deliverables: Include test plan document, test suite, test design and environment specifications, test log, defect report, test data, test summary report, test completion report, user acceptance test (UAT) report, and release notes.

Test Strategy: Determine test cost, effort, in-scope and out-of-scope features, identify testing types, document risks and issues, specify test logistics, and establish test criteria.

Test Environment and Data Planning: Determine hardware and software requirements, install necessary software, configure network, create test data, access builds, and verify the test environment.

Use of TestRail: TestRail facilitates adherence to best practices for test plan development by providing flexibility and visibility into the testing process, offering features such as organizing test cases, creating agile test plans, and tracking test execution progress.

Commonly Used Testing Types: Manual testing (e.g., smoke testing, exploratory testing, usability testing), automated testing (e.g., unit testing, regression testing, integration testing), and others (e.g., performance testing, security testing, accessibility testing).

TEST PLANNING IN A TEST MANAGEMENT TOOL

- Utilize test management tools like TestRail for comprehensive and agile test planning.
- Tactical approaches to test planning with TestRail can enhance agility and efficiency.

Milestones:

- Milestones in TestRail serve as containers for aggregating test artifacts and tracking testing activities related to specific outcomes.
- Milestones facilitate organizing test runs and plans, providing a centralized location for all related testing activities.
- The description field within milestones can be utilized to build a one-page test plan, allowing easy reference during planning.

Test Case Priority and Type:

- Test cases in TestRail enable pre-definition of testing scope and approach.
- Organize test cases based on priority and testing type hierarchies to structure the test plan effectively.
- Prioritize critical areas for testing, such as smoke testing for essential functionalities.

Test Reports:

- Test management tools like TestRail offer real-time reporting capabilities for test execution.
- Utilize TestRail's milestone summary report to track test objectives, test runs, assigned priorities, and more.
- Generate and share downloadable reports with team members and stakeholders for enhanced collaboration and visibility.

As projects grow in complexity, relying on spreadsheets for test planning can be cumbersome.

Test management tools like TestRail provide a more sophisticated and flexible approach to test planning, improving efficiency and adaptability.

Utilization of TestRail:

TestRail facilitates flexible and efficient test planning, offering features for organizing test artifacts, tracking testing activities, and generating insightful reports.

Leveraging TestRail can streamline test planning processes and enhance overall project management effectiveness.

Test Case

- Test case is a document under test artifacts.
- Used to verify application features.
- Includes executable steps, pre/post-conditions, test data, expected and actual results.

Test Scenario vs Test Case:

- Test scenario: High-level description of what to test.
- Test case: Detailed steps for testing a specific scenario.

Test Case Writers:

Varies by company:

- Developers write unit tests.
- Testers write integration and acceptance tests.

Steps for Writing Test Cases (Manual Testing):

- Test Case ID
- Description
- Pre-Conditions
- Test Steps
- Test Data
- Expected Result
- Post Condition
- Actual Result
- Status

Test Case Template Fields:

- Project & Module Name
- Reference Document

- Created By/Date, Reviewed By/Date, Executed By/Date
- Comments

Best Practices for Writing Test Cases:

- Easy to understand and execute

- End user's perspective

- Unique identifiers

- Clear descriptions, pre/post-conditions, and expected results

- Reusable & maintainable

- Utilize testing techniques

- Peer review

Popular Test Case Management Tools:

- PractiTest

- Test Rail

- Testpad

- Qase

- Klaros

- Test Collab

- QMetry

- Meliora Testlab

- TestLodge

- TestCaseLab

Verification and Validation in Software Testing

Verification: Evaluates if software artifacts meet specified requirements and standards.

Validation: Ensures software meets user needs and expectations.

What is Verification Testing?

- Static practice ensuring development elements adhere to standards.
- Includes code reviews, walkthroughs, inspections, design analysis.
- Occurs at every stage to confirm accurate system design and architecture.

Advantages of Verification Testing

- Reduces bugs in later stages.
- Provides insights for better development.
- Helps estimate emerging issues.
- Keeps software aligned with requirements.

When to Use Verification Testing?

- Conducted before implementing any feature or element.
- Ensures adherence to specifications and guidelines.

What is Validation Testing?

- Dynamic practice ensuring final product meets stakeholder needs.
- Includes various QA tests like unit, integration, performance, and user acceptance testing.
- Verifies tangible, quantifiable results of system functions.

Advantages of Validation Testing

- Detects bugs missed during verification.
- Reveals inadequacies in specifications.
- Ensures software meets customer demands and works across different conditions.

When to Use Validation Testing?

- Conducted after completing each feature or development step.
- Includes tests like unit tests, integration tests, and cross-browser testing.

Difference between Verification and Validation

- Verification: Static, human verification, targets requirements and design documents.
- Validation: Dynamic, involves both human and machine-based checking, targets final product for customer use.
- Verification precedes validation.
- Verification focuses on adherence to standards, while validation ensures the product meets user requirements.