**Test Cases Documentation**

Test Cases is a specific set of instructions that the tester is expected to follow to achieve a specific output during software testing. Test cases are documented keeping in mind the requirements provided by the client.  
  
**Some best practices for writing test cases include:**

* Clarity and transparency
* Focus on end-user requirements
* Map test cases to reflect every aspect of the user journey
* Break test case steps into the smallest possible segments
* Make test cases easy to understand

**Test Cases can help to:**

* Control test coverage
* Track future project requirements
* Eliminate confusion and anxiety
* Help testing teams understand what needs to be tested

**Best Practices for Writing Test Cases:**

1. Clarity and Transparency:

Why: Clear and transparent test cases reduce the chances of misinterpretation and help testers, developers, and other stakeholders understand the intended functionality.

How: Use simple language, avoid ambiguity, and provide detailed steps and expected outcomes.

2. Focus on End-User Requirements:

Why:Test cases should align with the end-user's perspective to ensure that the software meets their needs and expectations.

How: Base test cases on user stories, requirements, and use cases.

3. Map Test Cases to Reflect Every Aspect of the User Journey:

Why: Covering the entire user journey helps ensure that all critical paths and functionalities are tested.

How: Develop test cases for each user interaction, from login to logout, and all relevant actions in between.

4. Break Test Case Steps into the Smallest Possible Segments:

Why: Smaller steps make it easier to identify specific issues and isolate problems during testing.

How:Divide test cases into small, manageable steps, each focusing on a specific action or verification.

5. Make Test Cases Easy to Understand:

Why: Testers, developers, and other stakeholders should be able to comprehend test cases quickly and accurately.

How:Use a standardized format, provide clear instructions, and use consistent terminology.

**Benefits of Test Cases:**

1. Control Test Coverage:

* Test cases ensure comprehensive coverage of the application's features, helping to identify areas that may need additional testing.

2. Track Future Project Requirements:

* As requirements evolve, test cases act as a reference point for understanding what has been tested, what needs to be tested, and how changes may impact testing efforts.

3. Eliminate Confusion and Anxiety:

* Clearly documented test cases reduce confusion among team members, allowing for smoother collaboration and reducing anxiety related to uncertainties.

4. Help Testing Teams Understand What Needs to be Tested:

* Test cases provide a roadmap for testing teams, guiding them on what functionalities to test, how to test them, and what results to expect.

Test cases are essential for effective and efficient software testing. They not only serve as a guide for testing activities but also contribute to better communication, collaboration, and overall project success.

**What aspects does a test case contain:**

Test Case ID: Test Case ID will be the test case number of story.

Test Case Description: The description will be a short description of the functionality.

Assumptions: Assumptions should be mentioned in the assumptions columns if any.

Test Data: Test Data is the data with which you are performing the testing.

Pre-condition: Pre-condition should be anything that is done before the execution of the test case.

Test Steps: In Test Steps, you must mention steps like Login to application, Enter Username, and password, click the login button, Verify the page redirects to splash page.

Expected Result: The expected Result should indicate the behavior of the application after the execution of the test case. For successful Login expected Result would be a redirection to the splash page.

Actual Result: The actual Result indicates the actual behaviour of the application on the execution of the test case.

Status: Status can be marked as passed or fail depending on the actual Result.

Comments: Last is the optional comments.

**Test cases Design Techniques:**

Test case design techniques are methods used to create test cases that effectively validate the functionality of software applications. These techniques help ensure comprehensive test coverage and efficient testing processes. Here are some commonly used test case design techniques:  
  
1.Error Guessing:

* Rely on the tester's intuition and experience to predict where errors are likely to occur and design test cases to specifically target those areas.
* Example: If a previous version of the software had a specific defect, create test cases to verify that the issue has been resolved.

2.Equivalence Partitioning:

* EP is another technique that helps reduce the required test cases. By partitioning test input data into classes with an equivalent number of data, one can design test cases for each class or partition. This technique ensures that one thoroughly tests the software while minimizing the required test cases.
* For example, if a program requires an input of numbers between 1 and 100, an EP test would include a range of values, such as 1-50 and 51-100, and numbers outside that range, such as -1 or 101. Testing one value from each partition is sufficient to test all values within that partition.

3.Boundary Value Analysis:

* BVA identifies errors at the input domain’s boundary. A simple example of boundary value analysis would be testing a text box that requires the user to enter a number between 1 and 10. In this case, the boundary values would be 1 and 10, and we would test with values that are just above, at, and just below these boundaries.
* For example, we would test with 0, 1, 2, 9, 10, and 11. We can expect that errors or defects are most likely to occur at or near the boundary values. Identifying these issues early can help prevent them from causing problems later in the software development process.

4.Decision Table Testing:

* DTT is a technique that involves designing test cases based on decision tables formulated using different combinations of inputs and their corresponding outputs based on various conditions and scenarios sticking to other business rules. This technique ensures that we test the software thoroughly and accurately.
* For example, if a program offers discounts based on the type of customer and the amount spent, a decision table would list all possible combinations of customer types and the amount paid to receive a discount. Each cell in the table would specify the value that should be applied. Testers can ensure the program behaves correctly under various scenarios by testing all combinations.

5.State Transition:

* STD to test software with a finite number of states of different types. A set of rules that define the response to various inputs guides the transition from one state to another. This technique is handy for systems with specific workflows within them.
* For example, consider an e-commerce website that has different states such as “logged out,” “logged in,” “cart empty,” “cart not empty,” and “order placed.” The transitions between the states would be triggered by login in and logout, adding the product to the cart, removing the product from the cart, proceeding to checkout, etc. An STD can help visualize and test such complex states and transitions in a system.

**Significance of Test cases:**

Test cases serve several significant purposes in the software development life cycle:

1.Bug Detection:

* Test cases help identify and catch bugs or defects in the software early in the development process. By systematically testing different aspects of the software, you increase the chances of finding and fixing issues before the product is released.

2.Verification of Requirements:

* Test cases are designed based on the software requirements. By executing these test cases, you can verify whether the software meets the specified requirements. This ensures that the final product aligns with the intended functionality.

3. Documentation:

* Test cases serve as documentation of the expected behavior of the software. They provide a clear and structured way to understand how different components of the software are expected to function. This documentation becomes crucial for maintaining and enhancing the software in the future.

4.Regression Testing:

* As software evolves, new features are added, and existing code is modified. Test cases help in conducting regression testing, ensuring that changes to the codebase do not introduce new bugs or adversely affect existing functionality.

6.Quality Assurance:

* Test cases contribute to the overall quality assurance process. They help ensure that the software is reliable, robust, and capable of meeting user expectations. Effective testing improves the overall quality of the software and reduces the likelihood of post-release defects.

7.Cost Savings:

* Detecting and fixing defects early in the development process is generally less expensive than addressing issues after the software has been deployed. Test cases contribute to cost savings by preventing the release of faulty software and minimizing the need for post-release support and maintenance.

8.Customer Satisfaction:

* Reliable and bug-free software leads to higher customer satisfaction. By thoroughly testing the software and addressing any issues before release, you enhance the user experience and build trust with your customers.

9. Functionality Validation:

* Test cases are designed to validate that each aspect of the software's functionality behaves as intended. This includes checking whether features work correctly, user inputs are processed accurately, and the expected outputs are produced. Functionality validation ensures that the software meets user expectations and fulfills the specified requirements.

10. Efficiency Improvement:

* Test cases contribute to the efficiency of the development process by identifying defects early in the software development life cycle. Early defect detection minimizes the need for extensive rework and bug fixing during later stages, saving time and resources. This efficiency improvement is essential for meeting project timelines and delivery goals.

11. Security Assurance:

* Security-focused test cases are designed to identify vulnerabilities and weaknesses in the software's security mechanisms. By validating the software against potential threats, test cases ensure that sensitive data remains protected and that the application is resistant to common security risks. Security assurance is critical for safeguarding user information and maintaining the trust of stakeholders.

18. Sprint Tracking:

* Test cases play a crucial role in tracking progress within sprints in an Agile development environment. By executing and completing test cases, teams can assess the status of tasks, identify any impediments, and provide feedback to developers. Sprint tracking ensures that the team remains on course to meet sprint goals and allows for adjustments as needed.

19. Project Planning:

* Test cases contribute to project planning by providing insights into the testing effort required for each phase of the project. Understanding the testing scope and effort allows project managers to allocate resources effectively, plan testing activities in tandem with development tasks, and create realistic schedules. This contributes to overall project success.

20. Flow of Design:

* Test cases help in validating the flow of the software design by systematically testing the interactions between different components and modules. This ensures that the software's design aligns with user expectations and business requirements, providing a seamless and intuitive user experience.

21.Client Perspective:

* Test cases contribute to delivering a high-quality product, enhancing the client's perspective and satisfaction with the software. By ensuring that the software functions as intended and meets specified requirements, test cases contribute to a positive user experience and build confidence in the product's reliability.

22. Effort Optimization:

* Test cases optimize testing efforts by providing a systematic and repeatable approach to testing. A well-organized set of test cases allows for thorough coverage of different scenarios within a reasonable timeframe. This optimization helps testing teams focus on critical areas, identify defects efficiently, and ensure that the software is of high quality.