🎯 QA Terminology

Essential terms every QA professional should know

1. Test Case

- **What it means::** A test case is a clear, step-by-step guide to check if a specific part of an application works as expected. It includes what to do, what to input, and what result you should see. Think of it like a recipe for testing one thing.

- **Why it matters:**Test cases help testers stay organized and ensure they check everything properly.

- **Example:** To test a login feature, a test case might say: 1) Go to the login page, 2) Enter username “testuser” and password “Pass123,” 3) Click “Login,” 4) Confirm you see the dashboard. If the dashboard doesn’t appear, the test fails.

2. Bug / Defect

- **What it means:** A bug or defect is a mistake or error in the app that makes it behave incorrectly or not as intended. It’s like a glitch that stops the app from working the way it should.

- **Why it matters:** Bugs can frustrate users or even break the app, so finding and fixing them is crucial.

- **Example:** If you click “Add to Cart” on a shopping app but the item doesn’t show up in the cart, that’s a bug. Another example is a button that crashes the app when clicked.

3. Smoke Testing

- **What it means:** Smoke testing is a quick check to make sure the major features of an app are working after a new update or release. It’s like testing the “heart” of the app to see if it’s alive before diving deeper.

- **Why it matters:** It helps catch big problems early so testers don’t waste time on deeper tests if the basics are broken.

- **Example:** After a new version of an app is released, testers check if the homepage loads, users can log in, and key buttons like “Search” work. If these fail, there’s no point testing smaller features yet.

4. Regression Testing

- **What it means:** Regression testing is when testers re-check old features to make sure they still work after changes, like adding new features or fixing bugs. It ensures new updates didn’t accidentally break something that was working before.

- **Why it matters:** Apps are constantly updated, and new changes can mess up existing features.

- **Example:** After adding a new payment option to an e-commerce app, testers check if older features like adding items to the cart or applying a coupon still work correctly.

5. Unit Testing

- **What it means:** Unit testing focuses on testing the smallest parts of an app, like a single function or button, to make sure they work correctly on their own. It's like checking each Lego piece before building the whole model.

- **Why it matters:** Catching issues in small pieces early prevents bigger problems later.

- **Example:** If an app has a function to calculate a 10% discount, a unit test checks if entering $100 returns $90. Developers usually write these tests while coding.

6. Integration Testing

- **What it means:** Integration testing checks if different parts of the app work well together when combined. It's like making sure all the pieces of a puzzle fit properly to form the full picture.

- **Why it matters:** Even if individual parts work, they might fail when connected, so this ensures smooth teamwork.

- **Example:** After linking the login page to the user profile page, testers check if logging in correctly loads the user's name and details on the profile page.

7.System Testing

- **What it means:** System testing is testing the entire app as a whole to make sure everything works together from start to finish. It's like test-driving a car to check the engine, brakes, and lights all at once.

- **Why it matters:** It confirms the app is ready for real-world use by testing the full system.

- **Example:** For an online store, testers might go through the whole process: sign up, browse products, add to cart, enter payment details, and complete an order to ensure it all works smoothly.

8. User Acceptance Testing (UAT)

- **What it means:** UAT is when real users or clients test the app to confirm it meets their needs and expectations before it goes live. It's like a final "okay" from the people who will actually use it.

- **Why it matters:** It ensures the app is useful and user-friendly for its intended audience.

- **Example:** A company buying a new payroll app has its HR team test it to make sure it calculates salaries correctly and is easy to use before approving it for daily use.

9. Test Suite

- **What it means:** A test suite is a collection of related test cases grouped together to test a specific feature or part of the app. It's like a folder of all the tests needed for one area.

- **Why it matters:** It keeps testing organized and ensures all aspects of a feature are covered.

- **Example:** For a registration feature, the test suite might include test cases for valid registration, invalid email, blank fields, and password strength checks.

10. Sanity Testing

- **What it means:** Sanity testing is a quick check to confirm that a specific fix or small change works as expected without breaking anything nearby. It's like a spot-check to make sure the fix didn't cause new problems.

- **Why it matters:** It saves time by focusing only on the changed or fixed part before deeper testing.

- **Example:** If a bug made the "Reset Password" button crash the app, after fixing it, testers check just that button to ensure it now works without issues.

11. Black Box Testing

- **What it means:** Black box testing is when testers use the app like a regular user, without knowing how the code works. They focus on what the app does, not how it does it.

- **Why it matters:** It mimics real user behavior to catch issues users might face.

- **Example:** A tester tries logging in with a wrong password to see if the app shows an error message, without looking at the code behind the login feature.

12. White Box Testing

- **What it means:** White box testing is when testers or developers test the app while knowing and looking at the code. They check if the code itself is working correctly.

- **Why it matters:** It helps find issues in the code that might not show up during regular use.

- **Example:** A developer checks the code for a search function to ensure it correctly filters results, like showing only items matching the search term.

13. Test Plan

- **What it means:** A test plan is a detailed document that outlines how testing will be done. It covers what to test, who will test, when, and what tools or methods to use. It's like a roadmap for the entire testing process.

- **Why it matters:** It keeps the team organized and ensures nothing is missed during testing.

- **Example:** A test plan for a mobile app might say: Test on iOS and Android, check login and payment features, use 10 testers, and finish in two weeks.

14. Test Script

- **What it means:** A test script is a detailed set of instructions for running a test, often written for automated testing tools. It's like a script for a play, telling the tester or tool exactly what to do.

- **Why it matters:** It ensures tests are done the same way every time, especially for automation.

- **Example:** A test script might say: 1) Open Chrome, 2) Go to www.example.com, 3) Enter username "test" and password "1234," 4) Click "Login," 5) Verify dashboard loads.

15. Test Scenario

- **What it means:** A test scenario is a high-level idea of what to test, describing a situation or feature. It's broader than a test case and can include multiple test cases to cover it.

- **Why it matters:** It helps testers focus on key areas of the app that need checking.

- **Example:** A test scenario like "Test the checkout process" might include test cases for valid payments, failed payments, and applying discount codes.

16. Exploratory Testing

- **What it means:** Exploratory testing is when testers freely explore the app without a strict plan, trying different things to find unexpected bugs. It's like wandering through a new city to discover hidden spots.

- **Why it matters:** It catches issues that structured tests might miss by mimicking how real users experiment.

- **Example:** A tester randomly clicks buttons, enters weird inputs (like "@#$%" in a name field), or tries unusual actions to see if the app crashes or behaves oddly.

17. Boundary Testing

- **What it means:** Boundary testing checks how the app handles values at the edges or limits of what's allowed. It tests the "boundaries" of input ranges to find errors.

- **Why it matters:** Apps often break at extreme values, so this ensures they handle limits correctly.

- **Example:** If an age field only accepts 18–60, testers try 17, 18, 60, and 61 to see if the app accepts or rejects them correctly.

18. Functional Testing

- **What it means:** Functional testing checks if each feature of the app works as it's supposed to, focusing on what the app does. It's like making sure every button and function does its job.

- **Why it matters:** It ensures the app's core features work for users as designed.

- **Example:** Testing that the search bar finds products, the "Add to Cart" button adds items, and the logout button signs the user out properly.

19. Non-functional Testing

- **What it means:** Non-functional testing checks aspects like speed, usability, security, or how the app looks, rather than specific features. It's about how well the app performs overall.

- **Why it matters:** A slow or hard-to-use app can frustrate users, even if the features work.

- **Example:** Testing if a website loads in under 3 seconds, looks good on mobile, or stays secure when users enter personal details.

20.Test Environment

- **What it means:** The test environment is the setup where testing happens, including the hardware, software, and network used. It's like a sandbox where the app is tested before going live.

- **Why it matters:** A realistic test environment ensures the app will work in the real world.

- **Example:** Testing a banking app on a test server with fake user data, using the same browser versions (like Chrome or Safari) that real users have.

21. Severity

- **What it means:** Severity describes how serious a bug is and how much it impacts the app's functionality or user experience. It measures the "damage" caused by the bug, focusing on its technical impact. Think of it as answering, “How bad is this problem for the app or its users?”

- **Why it matters:** Severity helps the team understand which bugs are critical to fix because they break important parts of the app or ruin the user experience. It’s about the bug’s effect, not how quickly it needs to be fixed.

- **Levels of Severity (common examples):**

  - **Critical:** The app crashes or a major feature is completely broken, making it unusable.

  - **High:** An important feature doesn’t work properly, but users can still use other parts of the app.

  - **Medium**: A smaller feature is affected, or there’s a workaround for the issue.

  - **Low:** A minor issue, like a typo or a small visual glitch, that doesn’t affect functionality.

- **Example:**

  - **Critical:** The “Pay Now” button on an e-commerce app crashes the app every time it’s clicked, preventing any purchases.

  - **High:** The search bar doesn’t show results for certain keywords, but users can still browse manually.

  - **Medium:** A profile picture upload feature doesn’t work, but users can skip it and use the app.

  - **Low:** A button’s text is slightly misaligned, but it still works perfectly.

- **Key point:** Severity is set by testers based on how much the bug breaks the app or affects users, regardless of when it needs to be fixed.

22. Priority

- **What it means:** Priority decides how urgently a bug needs to be fixed. It’s about the importance of fixing the bug compared to other tasks, often based on business needs, user impact, or deadlines. Think of it as answering, “How soon should we fix this bug?”

- **Why it matters:** Priority helps developers and managers decide which bugs to tackle first, especially when time or resources are limited. A bug might be severe but low priority if it’s in a rarely used feature, or less severe but high priority if it affects many users.

- **Levels of Priority (common examples):**

  - **P1 (Urgent):** Must be fixed immediately, often because it stops users from using the app or causes major issues.

- **P2 (High):** Should be fixed soon, as it affects important features or many users.

-**P3 (Medium):** Can be fixed in the next release, as it's not urgent but still needs attention.

- **P4 (Low):** Can wait, as it's a minor issue with little impact on users.

- **Example:**

  - **P1:** A bug stops users from logging into a banking app, so it needs to be fixed right away.

  - **P2:** A checkout bug in an online store stops some payment methods but not all, so it's important but not critical.

  - **P3:** A typo in the app's "About Us" page can wait until the next update.

  - **P4:** A slightly wrong color on a button can be fixed later, as it doesn't affect functionality.

- **Key point:** Priority is often set by the project manager or product owner, considering factors like customer complaints, business goals, or release schedules.

Severity vs. Priority

- What's the difference?: Severity is about the bug's impact on the app (how bad it is), while priority is about how quickly it needs to be fixed (how urgent it is). A bug can have high severity but low priority, or vice versa.