Sanity Test

Sanity Test

- Introduction

A sanity test is a preliminary, informal, and superficial testing process performed on a software build or a specific module to determine if it is stable enough to proceed with more rigorous testing. The primary goal is to ensure that the most critical functionalities of the software are working as expected and that the build is not fundamentally broken. It is often performed after a new build is deployed or after minor code changes have been made.

- Purpose and Objectives

The core purpose of a sanity test is to quickly assess the basic health and stability of a software system. Key objectives include:

- Early Defect Detection: To identify major defects or show-stopping issues as early as possible in the development cycle.
- Build Verification: To confirm that the latest build is stable and has not introduced critical regressions.
- Resource Optimization: To avoid wasting time and resources on extensive testing of a build that is clearly unstable.
- Decision Support: To provide a quick go/no-go decision on whether to proceed with further, more in-depth testing phases.
- Confidence Building: To build confidence in the stability of the software before commencing more time-consuming test cycles.
- Characteristics of a Sanity Test

Sanity tests possess several distinct characteristics:

- Narrow Scope: They focus on a small, critical subset of the software's features and functionalities.
- Shallow Depth: They do not aim to uncover every possible defect but rather to verify the core behavior.
- Speed and Efficiency: They are designed to be executed quickly.
- Informal: They are often not documented in detail and may be performed by developers or testers.
- Reactive: They are typically performed in response to a new build or a change.
- Unscripted (Often): While some basic steps might be followed, they are frequently exploratory in nature.
- When to Perform a Sanity Test

Sanity tests are typically performed in the following scenarios:

- After a New Build Deployment: When a new version of the software is deployed to a testing environment.
- After Minor Code Changes: When only a few modules or functionalities have been updated.
- Before Regression Testing: To ensure that the build is stable enough to undergo a full regression test suite.
- After Bug Fixes: To verify that a critical bug has been resolved and that the fix has not introduced new issues in related areas.
- As Part of Continuous Integration/Continuous Deployment (CI/CD): To quickly validate each build before it progresses through the pipeline.
- Sanity Test vs. Smoke Test

It is important to distinguish sanity testing from smoke testing, although the terms are often used interchangeably.

- Smoke Test: A broader, more comprehensive set of tests designed to verify that the most critical functionalities of the software work. If a smoke test fails, the build is rejected. It is typically performed on a new build before any further testing.

Sanity Test: A subset of smoke tests, or a quick check of a specific area after a minor change. It is often performed on a changed* portion of the software to ensure that recent modifications haven't broken anything fundamental. A sanity test is usually performed after a smoke test has passed or as a preliminary check before a smoke test.

- Typical Sanity Test Activities

A sanity test typically involves a quick check of the following:

- Installation/Deployment: Verifying that the software can be installed or deployed without errors.
- Login/Authentication: Ensuring users can log in and out successfully.
- Core Functionality: Testing the primary features of the application. For example, in an e-commerce application, this might include adding an item to the cart and proceeding to checkout.
- Basic Navigation: Checking that users can navigate between key screens or modules.
- Major UI Elements: Ensuring that critical user interface components are visible and functional.
- Database Connectivity: Verifying that the application can connect to and interact with its database.
- Benefits of Sanity Testing

Implementing sanity testing offers several advantages:

- Reduces Risk: Identifies critical issues early, preventing them from propagating to later stages.
- Saves Time and Resources: Prevents testers from investing time in unstable builds.
- Improves Quality: Contributes to overall software quality by ensuring a stable foundation.

- Facilitates Faster Releases: Allows for quicker decision-making regarding build readiness.
- Enhances Team Collaboration: Fosters better communication between development and testing teams.
- Conclusion

Sanity testing is a vital, albeit often informal, part of the software testing process. It serves as a crucial gatekeeper, ensuring that software builds are stable and ready for more in-depth verification. By focusing on the most critical functionalities, sanity tests help to identify show-stopping defects early, saving time, resources, and ultimately contributing to the delivery of higher-quality software.