**What is Sanity Testing?**

Sanity Testing is a type of software testing that focuses on verifying that a specific part of the application is working as expected after changes, such as bug fixes, enhancements, or patches. The goal is not to perform exhaustive testing but to quickly check if the changes made to the software are functioning properly without introducing any new issues.

**Analogy for Sanity Testing**

Imagine you’re fixing a car. You take it to the mechanic because the brakes are malfunctioning. After the mechanic fixes the brake system, instead of going for a long test drive to check every part of the car, you simply test the brakes to see if they are working correctly. You press the brake pedal to ensure it’s responding and stopping the car.

Sanity testing in this analogy would be like testing just the brake system after the fix to ensure it works. You're not concerned with the entire car’s performance at this moment; you’re just verifying that the change (brake fix) was successful.

**Key Features of Sanity Testing**

1. **Quick and Narrow in Scope**: Unlike other tests like regression testing, which check the entire application, sanity testing is focused on a small area of the application that was modified or fixed.
2. **Verification of Specific Functionality**: It checks if the functionality of a particular feature works as expected after changes. It does not check the whole system.
3. **Unclear Requirements or Documentation**: It is often performed when the requirements are not clear or are very basic. Sanity testing is more about ensuring that the functionality is not broken than about validating each requirement in depth.
4. **Non-exhaustive**: It doesn’t involve detailed test cases, rather just enough tests to verify that the major components work.

**Real-time Example of Sanity Testing**

**Example 1: Software Update for a Mobile App**

Suppose you're testing a mobile application, and the developers have fixed a bug in the login screen where the "Login" button was unresponsive.

* **Sanity Test**: After the fix, a sanity test would simply check if the "Login" button now works as expected. You would enter a valid username and password, click the button, and check if it proceeds to the dashboard. You wouldn’t test other features of the app like the profile page, search functionality, or notifications at this stage, unless they are directly affected by the login issue.
* **Outcome**: If the "Login" button works, you can be sure that the specific bug fix is successful, and you can then decide whether to move on to more detailed tests like regression or functional testing.

**Example 2: E-commerce Website Bug Fix**

Imagine you're testing an e-commerce platform, and a bug was identified in the shopping cart functionality where the "Remove Item" button wasn't working.

* **Sanity Test**: After the fix, you perform a quick sanity test to verify that the "Remove Item" button is now working correctly. You add an item to the cart, click "Remove," and check if the item is removed from the cart. You don’t check the whole checkout process or any other shopping cart features at this point.
* **Outcome**: If the issue is resolved, you can confirm that the change has worked, and the system is stable for now.

**How to Perform Sanity Testing**

1. **Understand the Changes**: Review what part of the application has been changed. This could be bug fixes, new features, or enhancements.
2. **Identify Key Areas Affected by the Change**: Focus on the areas directly impacted by the changes. For instance, if a bug was fixed in the payment gateway, ensure that payments are being processed correctly.
3. **Test Core Functionalities**: Quickly test the critical functionalities that would break the system if not working properly. For example, in a booking system, testing the ability to book tickets might be the key functionality to test.
4. **Determine the Stability**: Perform the test and determine if the change has introduced new issues or if the modified feature works as expected.
5. **Pass/Fail Decision**: If the feature works without issues, it’s safe to say that the build is stable enough for further testing. If it fails, report the failure for further investigation before proceeding with other tests.

**When to Use Sanity Testing**

* **After a Bug Fix**: When a developer fixes a bug, you perform sanity testing to check if the fix works.
* **After a Small Change**: When a minor change is made (e.g., a UI change or a new button is added), you quickly check that it functions as expected.
* **When You Receive New Builds**: When new software builds are received after a change or update, sanity testing ensures that the key features are working.

**Difference Between Sanity Testing and Smoke Testing**

* **Sanity Testing**: Focused on checking whether the changes in a specific area of the application work as expected. It’s a subset of regression testing. It’s a narrower scope and usually happens after receiving a new build with minor changes.
* **Smoke Testing**: Often called "build verification testing," it is a basic level of testing performed on the entire system to ensure that the application is stable enough to proceed with more rigorous testing. It's more like testing the whole system at a high level.

**Example of Smoke Testing:**

If the mobile app had a major update, a smoke test might involve quickly checking whether the app opens, whether key functionalities (like logging in, registration, or navigation) work, and whether it crashes. If the app passes these basic tests, it can then undergo detailed sanity and functional tests.

**Why is Sanity Testing Important?**

1. **Quick Validation**: It provides a fast way to check if the system is ready for further testing.
2. **Minimize Risk**: It ensures that critical parts of the application are not broken, minimizing the risk of catastrophic failures later on.
3. **Save Time**: It helps testers avoid wasting time on tests if the most basic features are not working. If sanity tests fail, the system is not ready for further testing.

**Conclusion**

Sanity testing is like a quick health checkup for the specific areas of an application that have been changed. It helps in ensuring that small modifications or bug fixes don’t break core functionality and that the software is in a stable enough state for further, more detailed testing. By teaching new engineers to focus on the critical elements of the application and keep tests simple yet effective, sanity testing ensures that they catch any major issues early in the process.