SESSION 6

Software Testing Terminology

Regression Testing

Regression Testing

Regression testing is the process of testing a software application to ensure that new changes (like updates, bug fixes, or new features) do not negatively impact the existing functionality of the software.

Example:

Imagine you have a mobile app that allows users to log in, view their profile, and send messages. If a developer adds a new feature, such as the ability to upload profile pictures, regression testing ensures that after this change, users can still log in, view their profiles, and send messages without any issues.

Types of Regression Testing

- Unit Regression Testing:
  - We test only the specific changes made by the developer.
- Regional Regression Testing:
  - We test the part that was changed along with the connected parts.
  - Impact Analysis Meeting will be conducted to figure out which parts will be affected by the changes, involving both the testing team and the developers.
- Full Regression:
  - We test the main part that was changed and also check the rest of the software to be thorough.
  - For example, if the developer made changes in many areas, instead of checking each one separately, we test everything together in one full round.

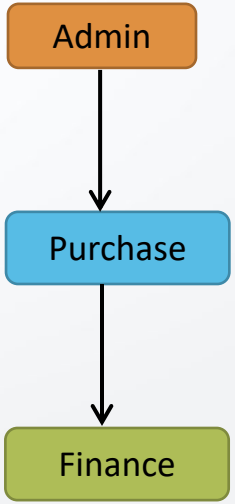
Re-Testing

Whenever the developer fixed a bug, tester will test the bug fix is called Re-testing.

- Tester close the bug if it worked otherwise re-open and send to developer.
- To ensure that the defects which were found and posted in the earlier build were fixed or not in the current build.
- **Example:**
  Build 1.0 was released. Test team found some defects (Defect Id 1.0.1, 1.0.2) and posted.
  Build 1.1 was released, now testing the defects 1.0.1 and 1.0.2 in this build is retesting.

Re-Testing Vs Regression Testing

- An Application Under Test has three modules namely **Admin**, **Purchase** and **Finance**.
- Finance module depends on Purchase module.
- If a tester found a bug on Purchase module and posted. Once the bug is fixed, the tester needs to do **Retesting** to verify whether the bug related to the Purchase is fixed or not and also tester needs to do **Regression Testing** to test the Finance module which depends on the Purchase module.
- In summary, **Re-testing** is specifically testing the fixed bug to confirm it's resolved, while **Regression Testing** is making sure that the fix hasn't caused new issues or disruptions, especially in modules that are interconnected.



Smoke Testing

Smoke Testing

- Also called BVT(Build Verification Test).
- Focus on **stability of the build** and we check whether it is ready for further testing or not.
- During smoke test we check build installation, appearance of basic screens and navigations etc.
- The term "smoke" comes from the idea that if there's a major issue, it would generate enough smoke to stop further testing.
- Examples:
  - Application is able to install or not
  - Once installed, all the screens are working properly or not
  - Backend database is up and running or not.
  - All the API’s related to application are invoked or not etc.

Sanity Testing

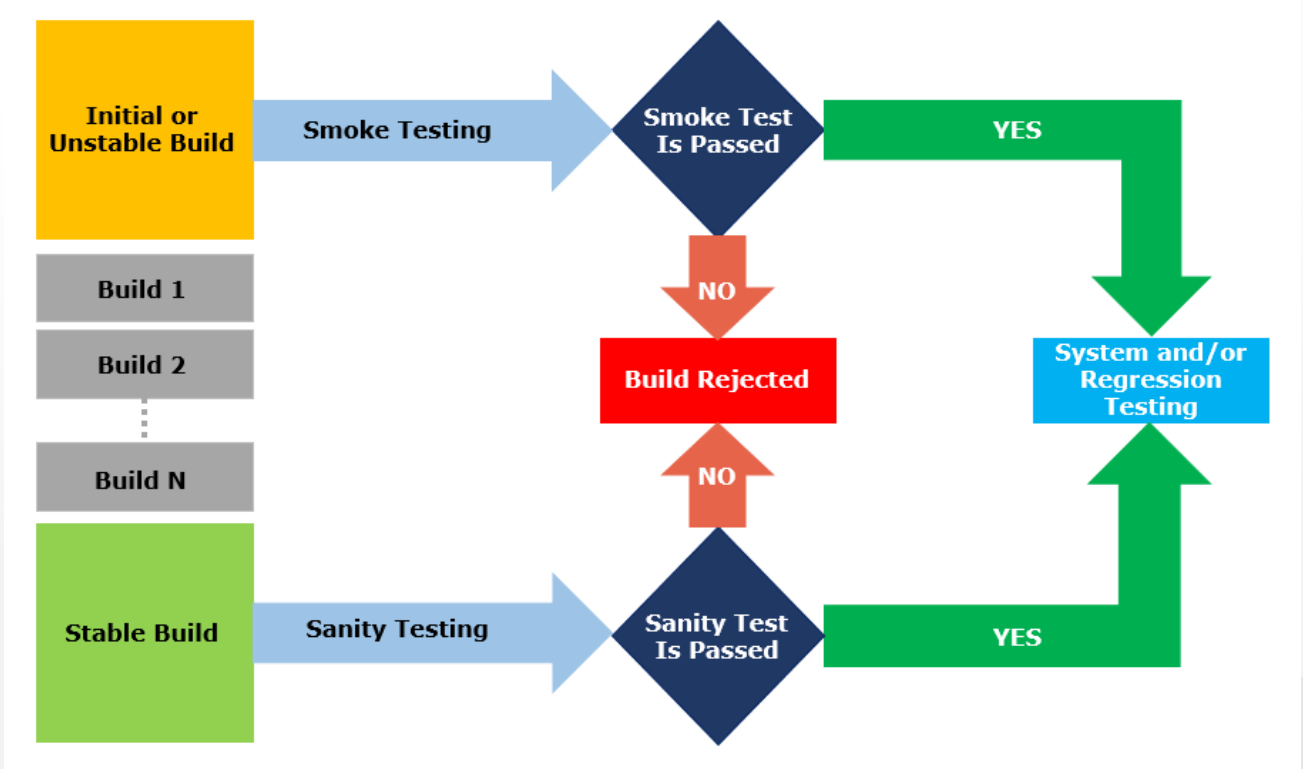
Sanity Testing

- Also called Basic Functional Test (BFT).
- Focus on **basic functionality** and we check whether basic functionality is working or not.
- During sanity test we check basic functionalities like login, user registration and logout etc.
- Examples:
  - Sign up option is available on login page.
  - Clicking "Sign up" redirects to proper , sign up form.
  - Clicking Sign in does not re-direct to "Sign up" form.
  - Submitting "Sign up" form goes successful, with out crash.
  - User signed up, is able to login

Sanity Vs Smoke Testing

Smoke Testing	Sanity Testing
Smoke Test is done to make sure the build we received from the development team is testable/stable or not.	Sanity Test is done to check for the main functionalities of the application without going deeper.
Smoke Testing is performed by both developers and testers.	Sanity Testing is performed by testers alone.
Smoke Testing, build may be either stable or unstable	Sanity Testing, build is relatively stable
It is done on initial builds.	It is done on stable builds.
Usually it is done every time there is a new build release.	It is planned when there is no enough time to do in-depth testing.

Sanity Vs Smoke Testing



Exploratory Testing

- We have to explore the application ,understand completely and test it.
- Understand the application , identify all possible scenarios , document it then use it for testing.
- We do exploratory testing when the Application ready but there is no requirement.
- Test Engineer will do exploratory testing when there is no requirement.

Drawbacks:

- You might misunderstand any feature as a bug (or) any bug as a feature since you do not have requirement.
- Time consuming
- If there is any bug in application , you will never know about it.

Adhoc Testing

- Testing application randomly without any test cases or any business requirement document.
- Adhoc testing is an informal testing type with an aim to break the system.
- Tester should have knowledge of application even thou he doesn't have requirements/test cases.
- This testing is usually an unplanned activity.

Monkey Testing

- Testing application randomly without any test cases or any business requirement document.
- Monkey testing is an informal testing type with an aim to break the system.
- Tester do not have knowledge of application
- Suitable for gaming applications.

Exploratory Vs Adhoc Vs Monkey Testing

Testing Type	Documentation	Plan	Testing Style	Tester's Knowledge	Testing Approach	Purpose	Application Type
Exploratory Testing	None	No formal plan	Informal	Testers don't know much about the application	Random testing	Intention is to learn or explore the functionality of the application	Any application that is new to the tester
Adhoc Testing	None	No formal plan	Informal	Testers should have some knowledge of the application's functionality	Random testing	Intention is to break the application or find out corner defects	Any application
Monkey Testing	None	No formal plan	Informal	Testers don't know much about the application	Random testing	Intention is to break the application or find out corner defects	Typically used for gaming applications

Positive Testing

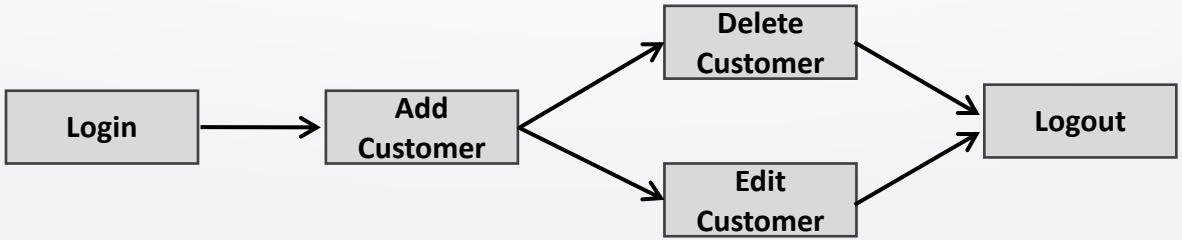
- Positive testing focuses on ensuring that a system behaves as expected when provided with valid inputs. The goal is to confirm that the software functions correctly under normal or expected conditions. In positive testing, the tester checks if the application does what it is supposed to do when everything is right.
- **Examples of Positive Tests:**
- **Login Test:** Entering valid credentials and checking if the user can successfully log in.
- **Calculator Addition:** Adding two positive numbers to ensure the calculator gives the correct sum.
- **Form Submission:** Filling out a form with valid data and verifying that it is successfully submitted.

Negative Testing

- Negative testing is about examining how well a system handles invalid or unexpected inputs. The goal is to identify potential weaknesses or vulnerabilities in the software by deliberately providing it with incorrect or inappropriate data.
- **Examples of Negative Tests:**
- **Login Test (Negative):** Attempting to log in with an incorrect password to check if the system rejects invalid credentials.
- **File Upload (Negative):** Trying to upload a file in a format not supported by the application and verifying if the system handles it gracefully.
- **Credit Card Payment (Negative):** Entering an expired credit card date during a payment process to see if the system detects and handles this scenario correctly.

End-To-End Testing

- End-to-End Testing is a comprehensive testing approach that evaluates the entire application flow from start to finish.
- It involves testing the interactions between various components and systems to ensure they work seamlessly together.



End-To-End Testing

Example of End-to-End Testing for an E-commerce Application:

- Imagine testing the process of a customer making a purchase on an e-commerce platform.
- **Scenario: User Makes a Purchase**

  Steps:

  1) The user logs into the e-commerce website.
  2) The user browses products, adds items to the cart, and proceeds to checkout.
  3) The user provides shipping and payment information.
  4) The system processes the payment and generates an order confirmation.
  5) The user receives an email confirmation.

  End-to-End Testing Checks:

  1) Confirm that users can successfully log in.
  2) Ensure the shopping cart calculates the correct total.
  3) Verify that the checkout process collects and processes shipping and payment information accurately.
  4) Check that the system generates a valid order confirmation.
  5) Confirm that the user receives the expected email confirmation.

Globalization & Localization Testing

Globalization Testing

- Globalization Testing checks if an application can function seamlessly across different regions and cultures, accommodating various languages, date formats, and currencies.

Localization Testing

- Localization Testing ensures that an application is adapted for a specific locale or target audience by verifying language translations, cultural preferences, and regional requirements.
- In simple terms, globalization testing makes sure your software can work anywhere in the world, and localization testing ensures it's a good fit for the specific cultural and linguistic needs of a particular region.