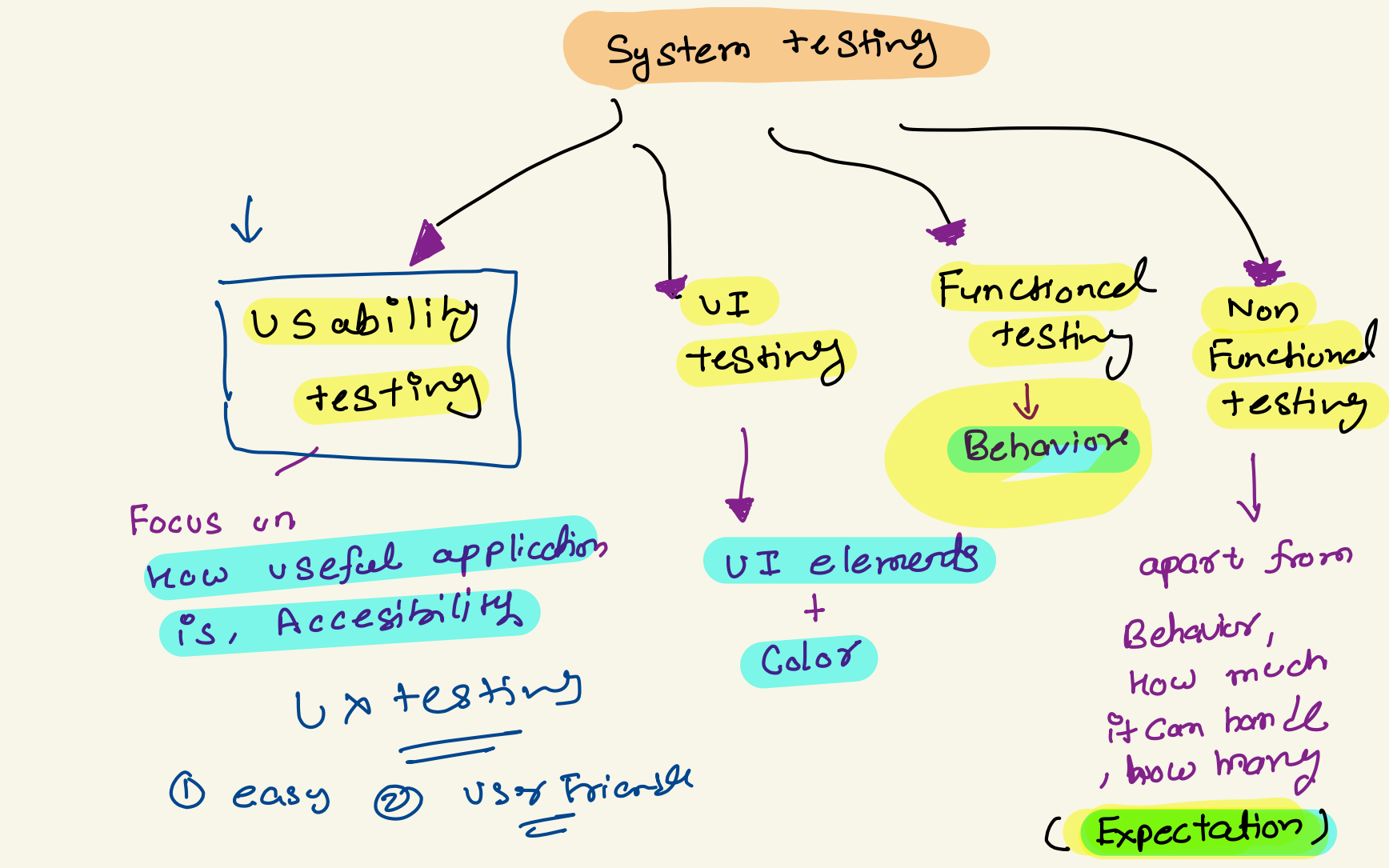

System testing
Usability testing
Focus on
How useful application is, Accesibility
UX testing
① easy ② User Friendly
UI testing
UI elements
+
Color
Functional testing
Behavior
Non Functional testing
apart from
Behavior,
How much it can handle, how many
(Expectation)

* Functional testing → web
→ mobile

→ properties of ^ elements, Focus, Selection, disabled enable, visible. Links

→ web application → Links, cookies, sessions
Author, errors, logs,

→ Database testing → proper data Stored.
→ Transactions (DML)
Data manipulation Langua.
↳ insert, update, delete

→ Functions, procedures,
triggers, indexes, views, Table
& Column, Table Relationship

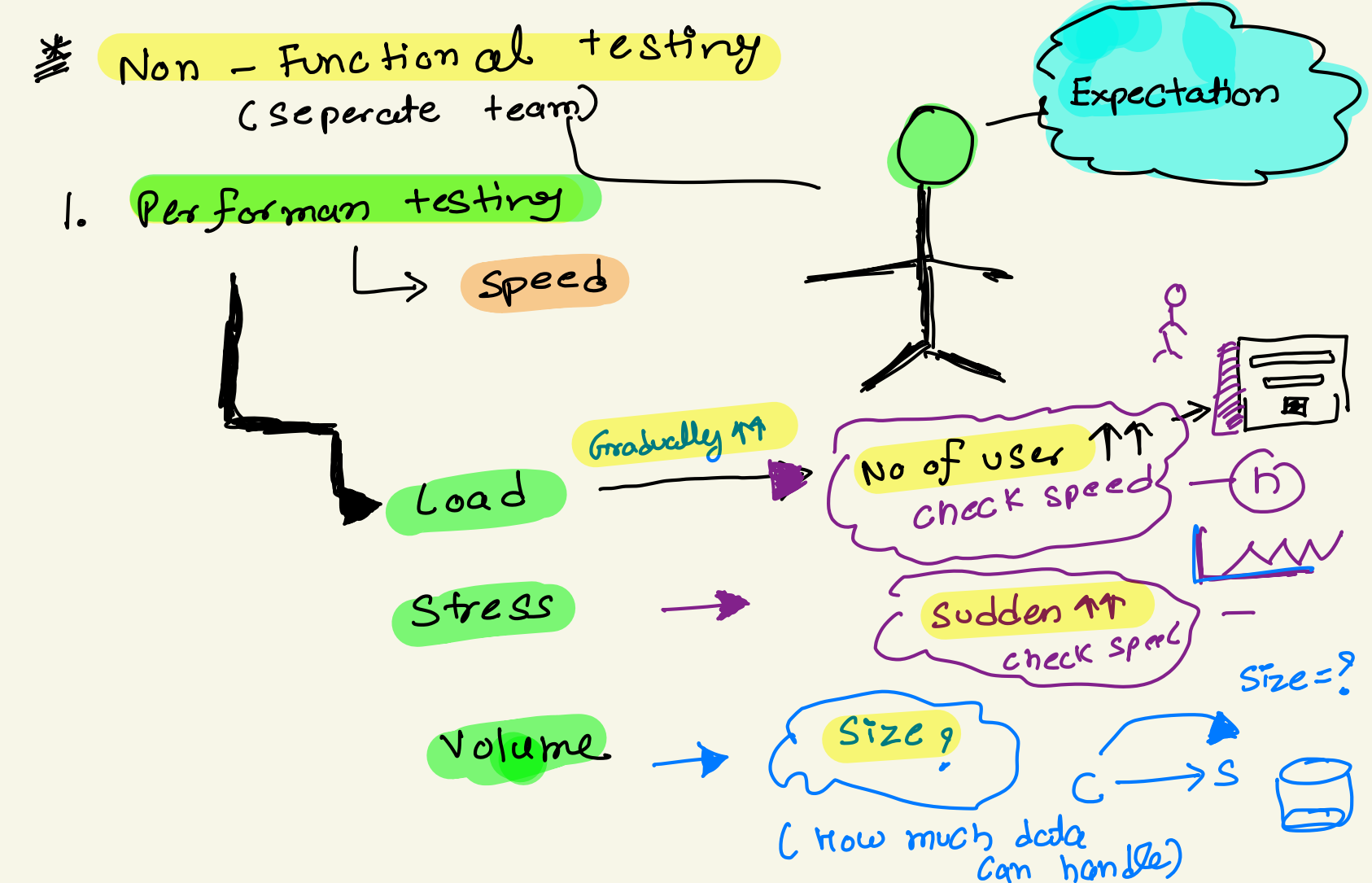
* Proper Errors Handling
* Proper Links (Interal, External, Broken)?
* Cookies & Session →

→ Live Demo of Registeration page.

→ vwo.com

Non - Functional testing

(seperate team)



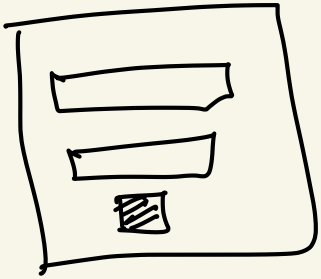
2. Security testing

How secure ?

Authentication

VS

Authorization



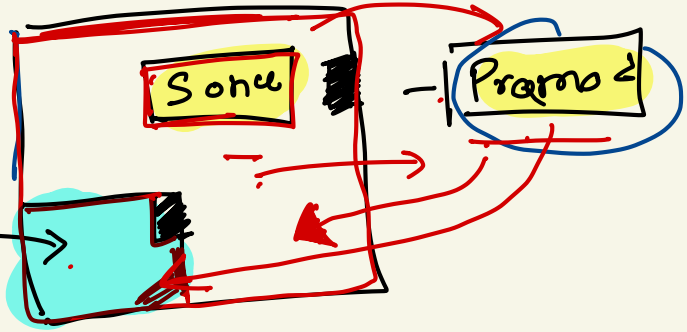
Who are you

(User valid ?)

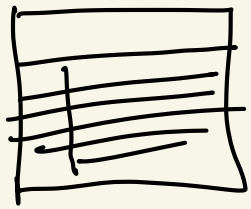
VIP

Permission

Private Room



3. Recovery testing : (Can we recove Something)



Excel

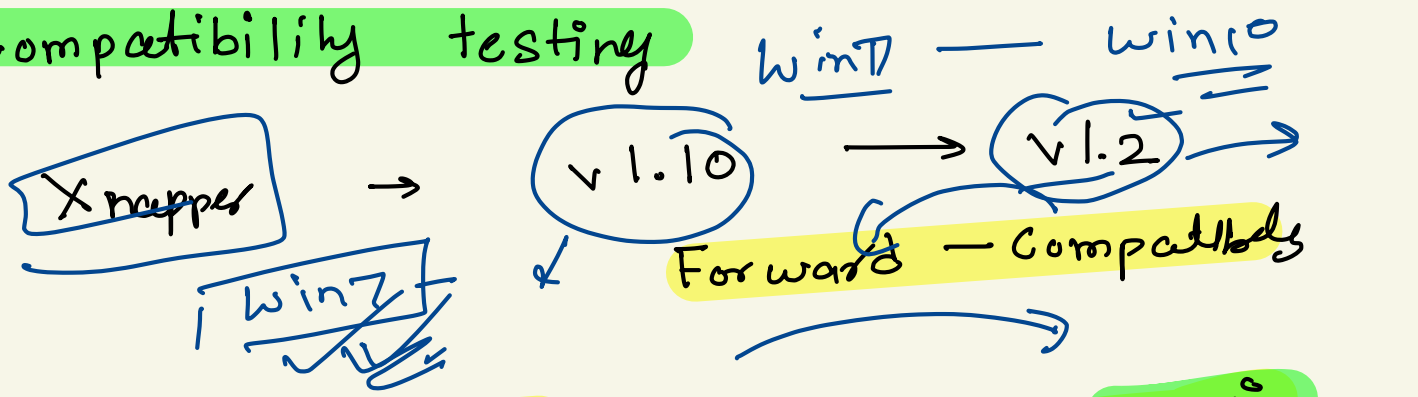
Recovered

Doc

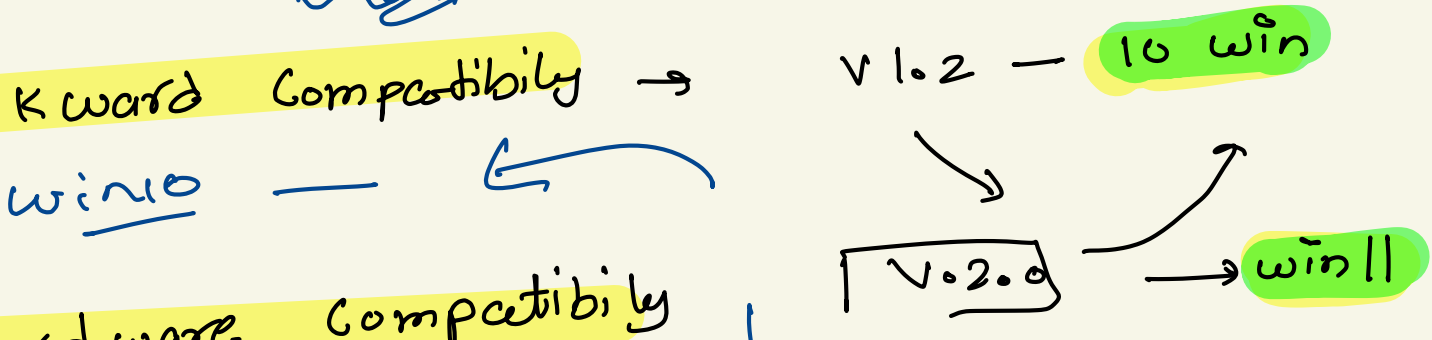
Checking if Net down light off → Normal State

.
.
.

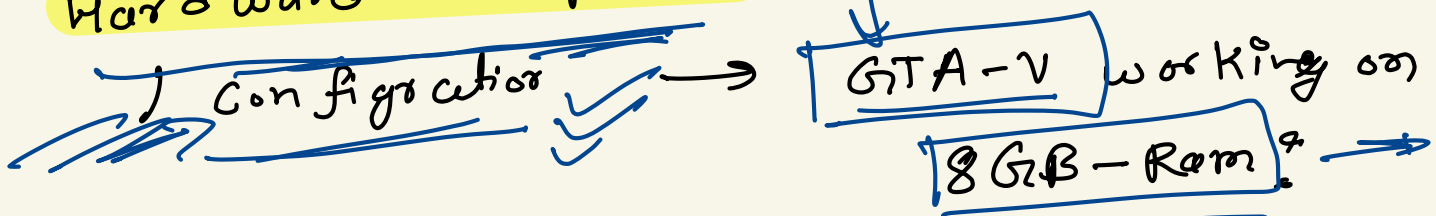
4. Compatibility testing



Backward Compatibility →



Hardware Compatibility



Installation testing → Check if install — easy
Check uninstall

Sanitation / Garbage testing

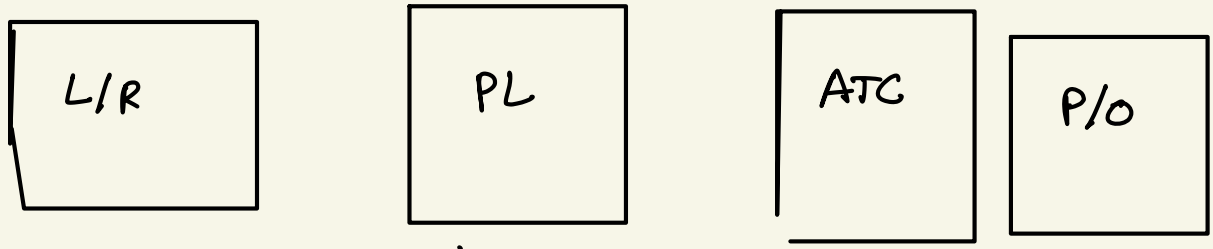
↳ Extract Functionality (Non Req) → X

— Delete them

Functional Testing	VS	Non-Functional Testing
Test the functionality of the software.		Test the non-functional aspects or readiness of the the software including performance, usability, reliability.
It has to be done before Non-Functional Testing.		It will be done after Functional Testing completes.
It is also called as Behavioral Testing and focuses on the underlying application features.		Focuses on the performance of the application.
It can be done manually, though test cases can be automated once application is stable.		It's hard to do it manually. It usually need already existing applications to measure and test application performance.
Types of Functional Testing includes Unit Testing, Smoke Testing, Integration Testing, Regression Testing, System Testing, User Acceptance Testing.		Types of Non-Functional Testing includes Volume testing, Load Testing, Stress Testing, Recovery Testing, Scalability Testing, Security Testing
Test data can be prepared using the business or functional requirements of the application		Test data can be prepared using the performance requirements of the application
Testing tools used for functional testing includes UFT(Previously QTP), Selenium, Ranorex, Telerik Test Studio, Micro Focus, Sahi, TestComplete, IBM Rational		Testing tools used for non-functional testing includes JMeter, LoadRunner, WebLOAD, NeoLoad, LoadComplete
Example Test Case - Test whether the user is able to login to the application.		Example Test Case - Time required to load the home page.

* Regression testing → Going Back to previous state of Something

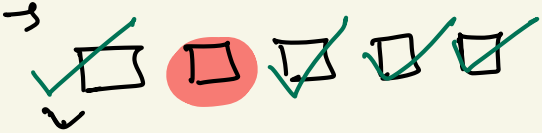
Bug



Build B_0
Build B_1 → Fixed
Build B_2 → -

* verification of dependent module / modified module that should not impact other modules.

1. Unit module → □ - AIC : only

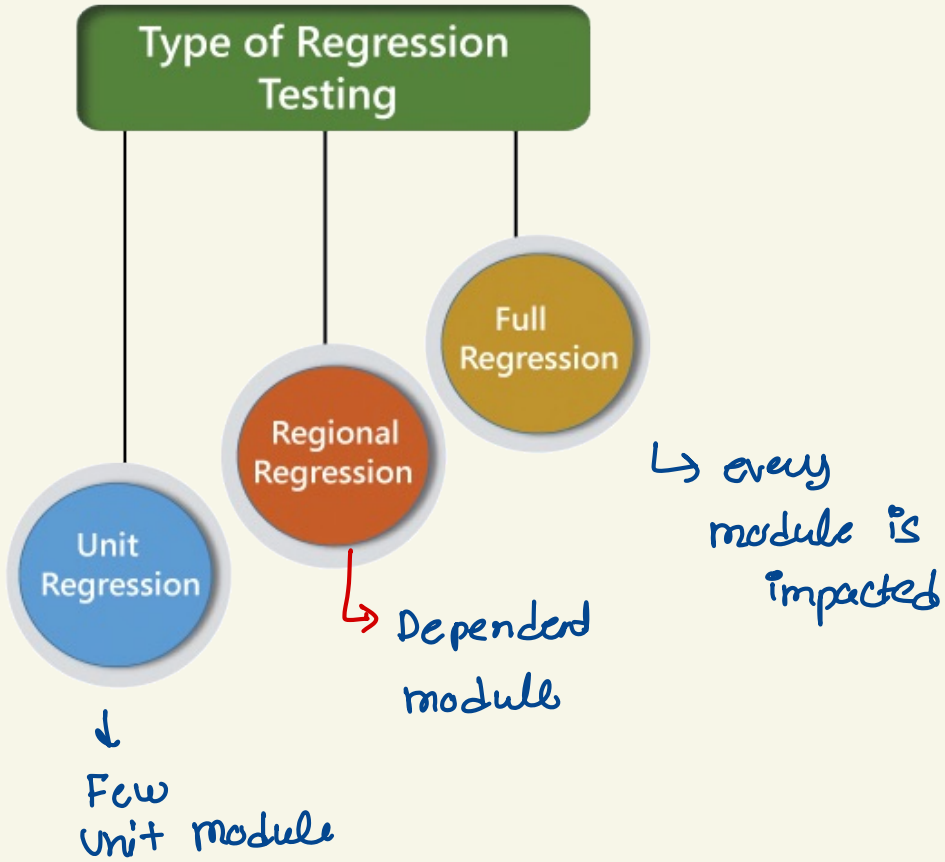
2. Full regression → → 

complete

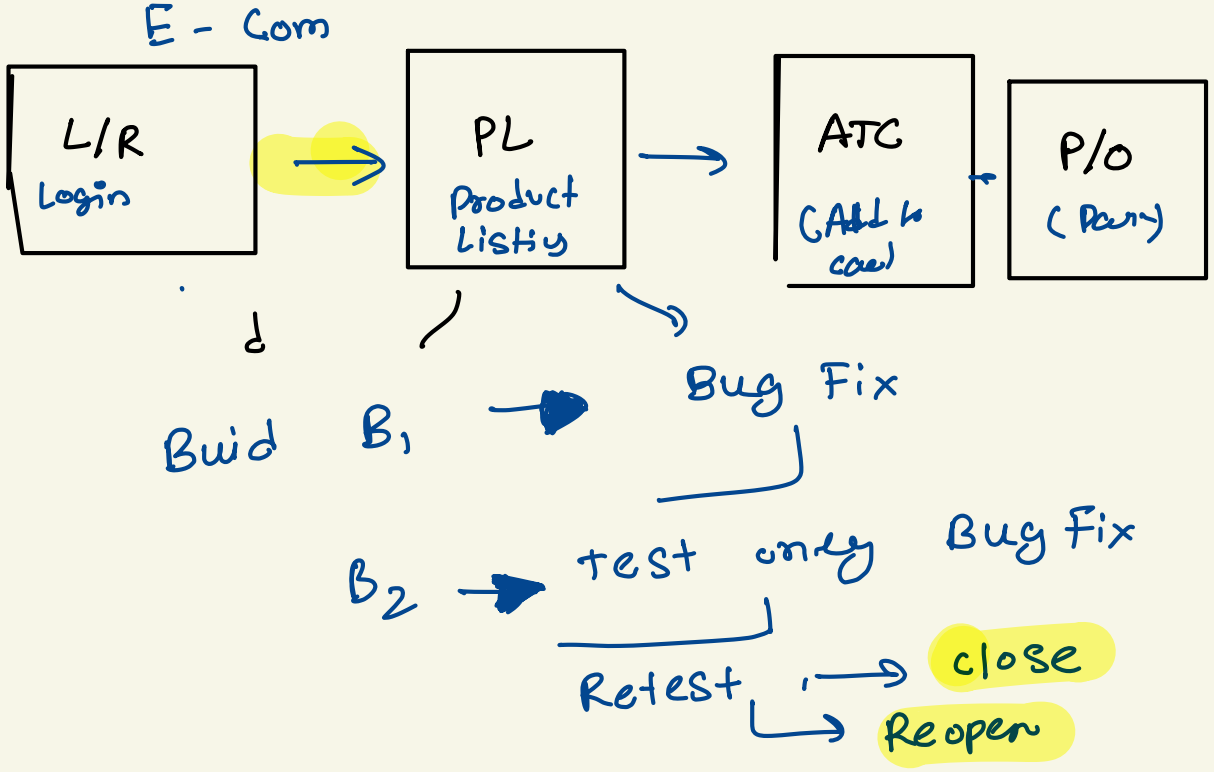
* Make Sure you do a impact Analysis Then Follow the Correct type of Regression testing.

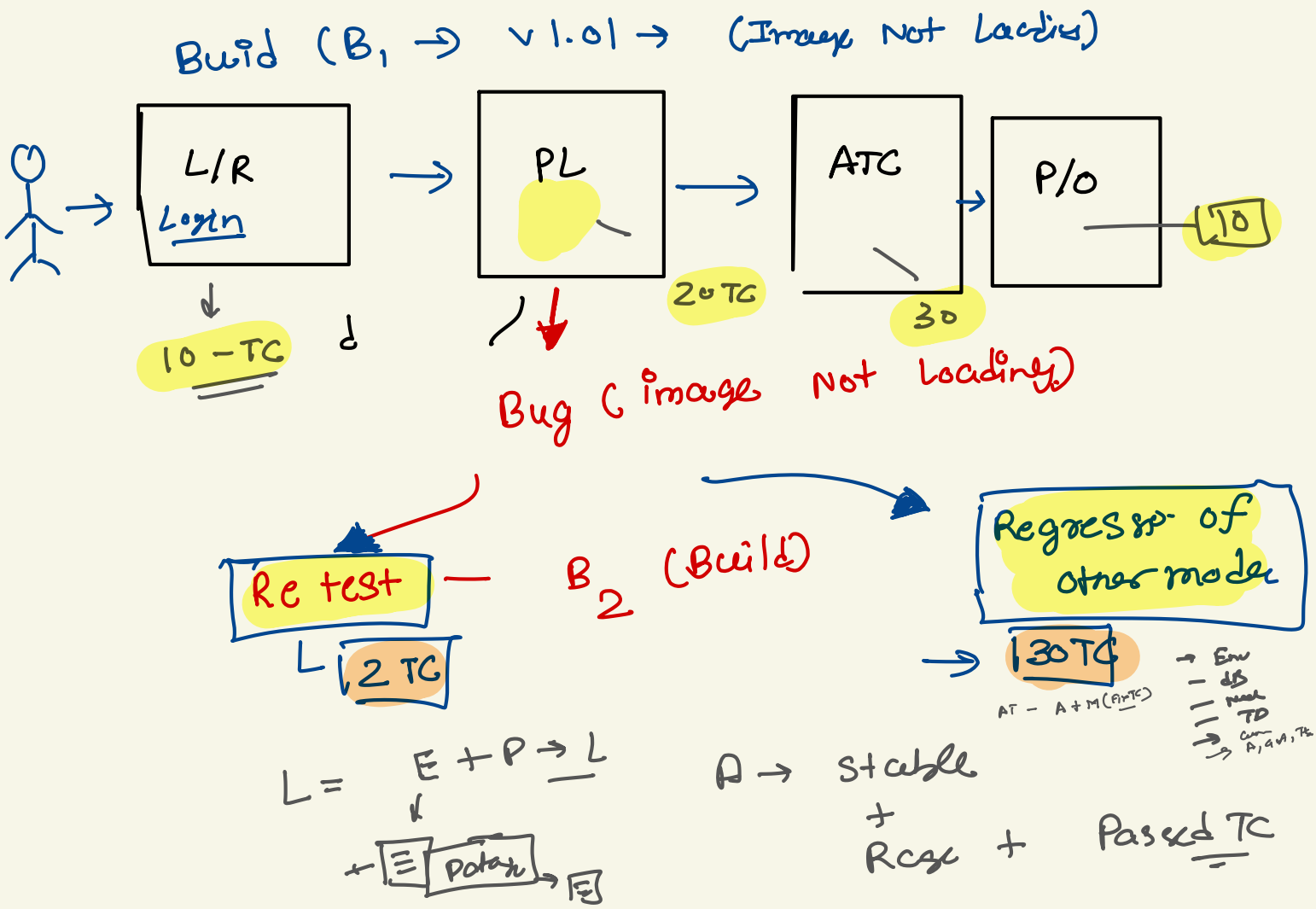
3. Partial regression → (Dependent modules)

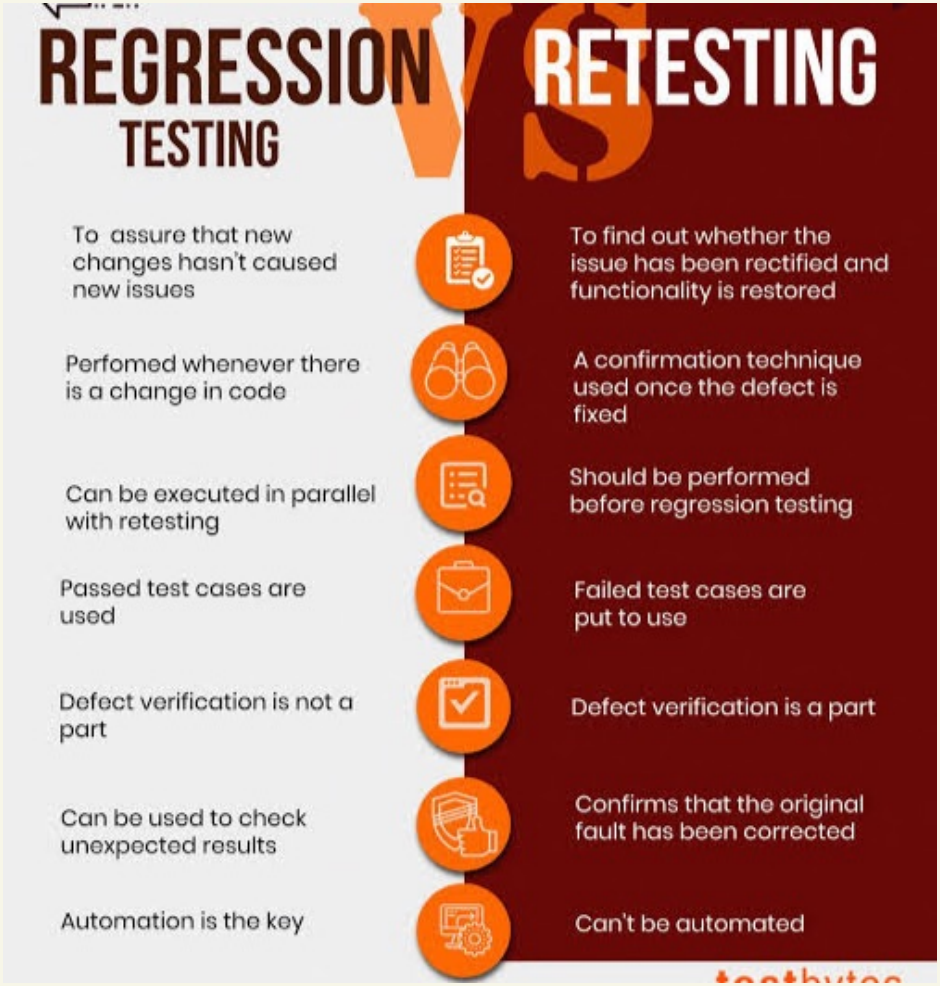
* DO NOT RUN FULL Regression unless absolutely essential


Type of Regression Testing
Unit Regression
Regional Regression
Full Regression
Few unit module
Depended module
every module is impacted

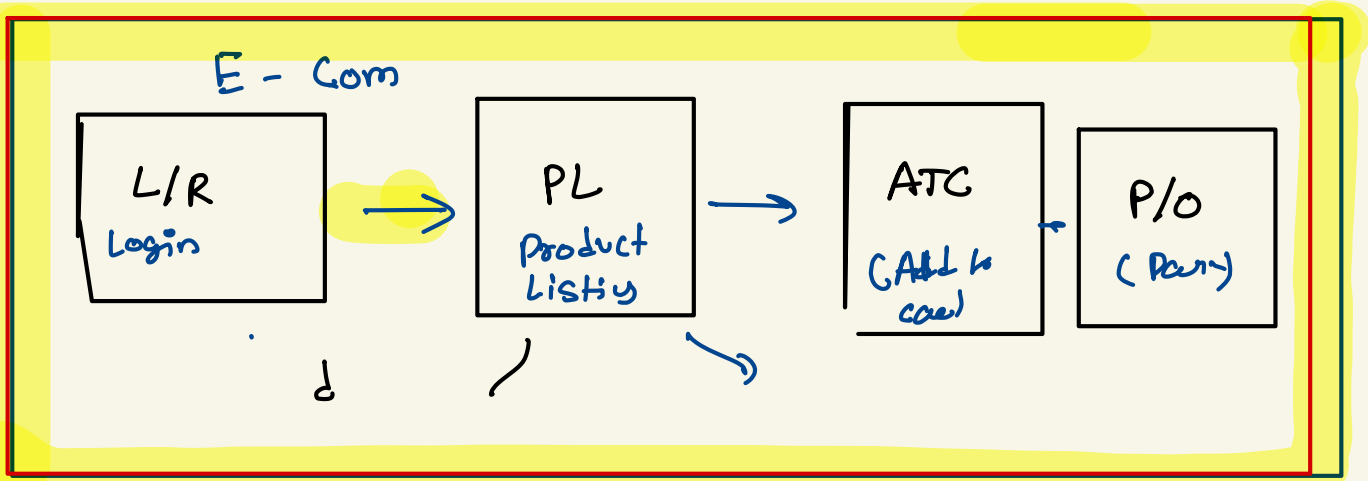
Retest


E - Com
L/R
Login
PL
Product
Listing
ATC
P/O
Build
B_1
Bug Fix
B_2
test only
Bug Fix
Retest
close
Reopen


Build (B1 → V1.01 → (Image Not Loading)
L/R
Login
PL
ATC
P/O
10
10 - TC
20 TC
30
Bug (Image Not Loading)
Re test
B2 (Build)
Regress of other modes
12 TC
130 TC
AT - A + M(A+TC)
→ Env
→ DB
→ Mail
→ TD
→ Conf A, M, TC
L = E + P → L
Data
A → Stable + Regr + Passed TC


REGRESSION TESTING
VS
RETESTING
To assure that new changes hasn't caused new issues
To find out whether the issue has been rectified and functionality is restored
Perfomed whenever there is a change in code
A confirmation technique used once the defect is fixed
Can be executed in parallel with retesting
Should be performed before regression testing
Passed test cases are used
Failed test cases are put to use
Defect verification is not a part
Defect verification is a part
Can be used to check unexpected results
Confirms that the original fault has been corrected
Automation is the key
Can't be automated

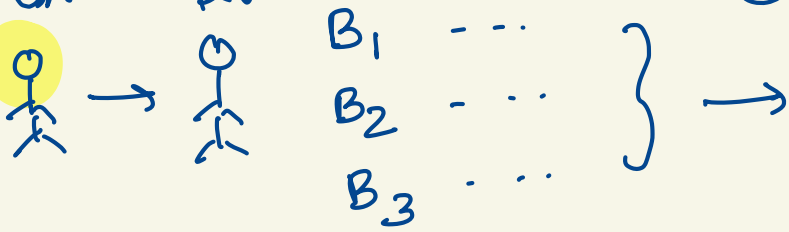
Smoke Testing vs Sanity Testing



Smoke testing → ① Critical Functionality checking

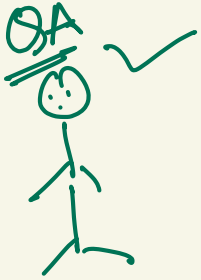
② Build verification testing

QA ✓ DEV ✓



→ Smoke testing (Unstable Build)

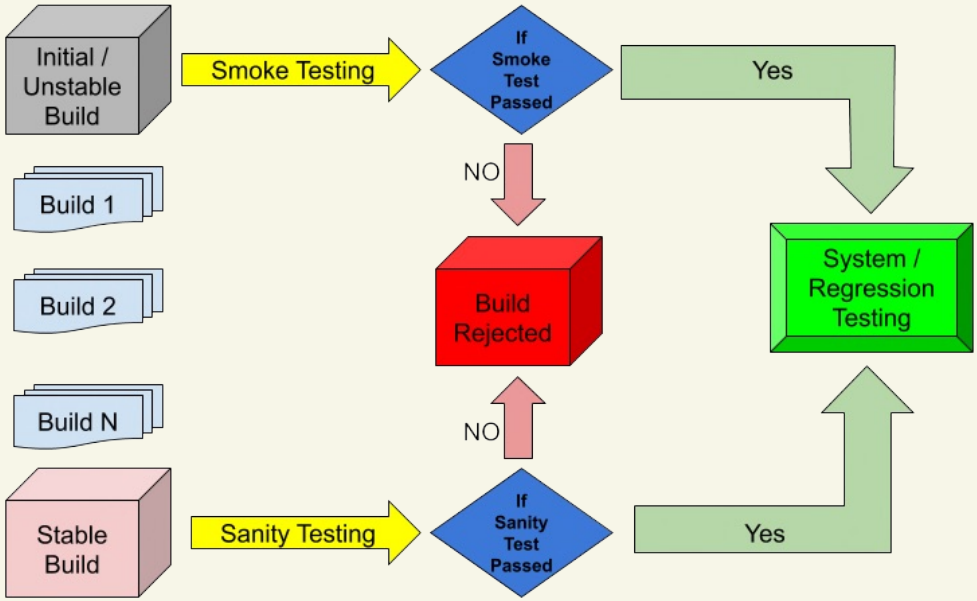
Sanity testing ① Basic Functional Stable Builds (Install work)



② By .... Sanity ③ Tester)

... Reject Build → ?
↳ Bn

(Basic Sanity Fail)


Initial / Unstable Build
Smoke Testing
If Smoke Test Passed
Yes
NO
Build 1
Build 2
Build N
Build Rejected
System / Regression Testing
NO
Stable Build
Sanity Testing
If Sanity Test Passed
Yes

* Smoke test

→ Build is Stable (Initial)

→ Dev / QA

→ Initial Builds

→ Basic testing

New → Build

Sanity testing

→ During Release

→ main Func ✓

→ QA

→ Stable builds

→ regression testing

→ No time to depln.

Exploratory testing :

1. No Req → Build / Application is Ready
2. Explore → Note down → Raise Bug
3. Understand → Rough Req → test → Bug

Problems → * Feature or Bug ?
* Time consuming
* Bug (Find) difficult
* DAA :C (Sad)

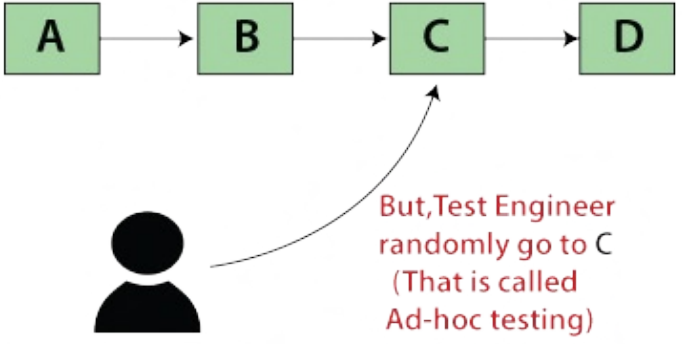
* Adhoc testing

1) → Informal testing → Aim to break the application

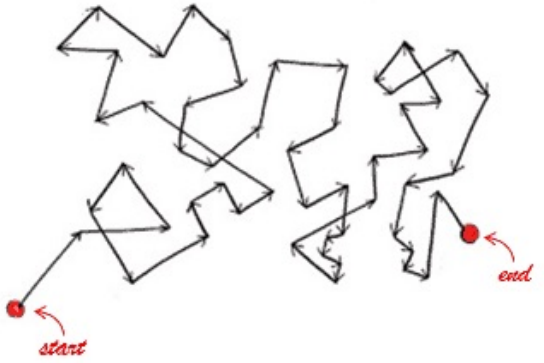
2. No testcase / BRD

3. Un planned activity

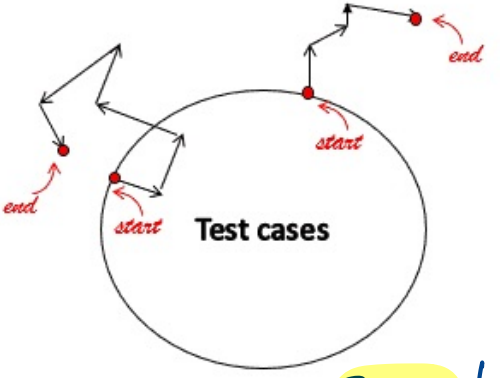
4. Break the System / App (for bloggs)



Ad hoc testing



Exploratory Testing



(Based on previous exp)

(Based on steps to explore)

↳ Break application

↳ some corner case

Adhoc Testing	Monkey Testing
Adhoc testing is random and does not rely on or use Test Cases.	Since, this is also random in nature, therefore test cases are not used in monkey testing.
The purpose of the tester in adhoc testing is to crash the application or find a fault by using the application randomly.	Tests are randomly executed with random or invalid data to check for crashes or not in monkey testing.
In Adhoc testing a group of testers tests whatever they think is required as per their knowledge of the application,	Here tester will not have much knowledge about the application, and they do not test in specific path. They test randomly by clicking on random objects and entering the random and invalid data to check id the application giver an error or nor.

Monkey / Gorilla testing

→ Random

→ No Test Case

→ No Knowlege of Application

→ (Good for → apps, Games, Teewee application

Adhoc

No plan

No Documentation

Random testing

Break the App

Knowlegde of App is require

Good for any App.

Exploratory

No plan

No Documentation

Random

Learn / Explore app / Functions

No Knolegde

New App

Monkey

No plan

No Documentation

Random

Break

No Knowlg

Gaming, Utility

Positive testing

1. valid inputs
2. Application should work as Expected
3. Enter pizza order

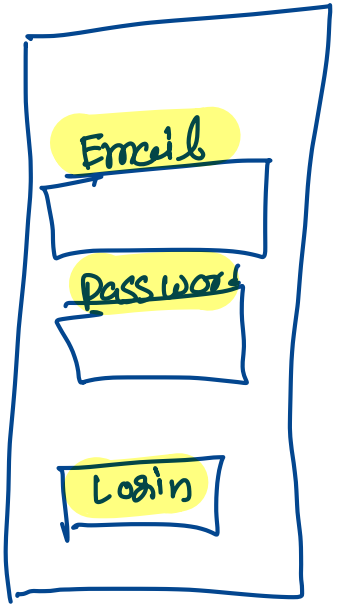
4. Accept No → Entered No.

Negative testing

1. Invalid inputs
2. Application Shoud No breaks
3. order Pizza

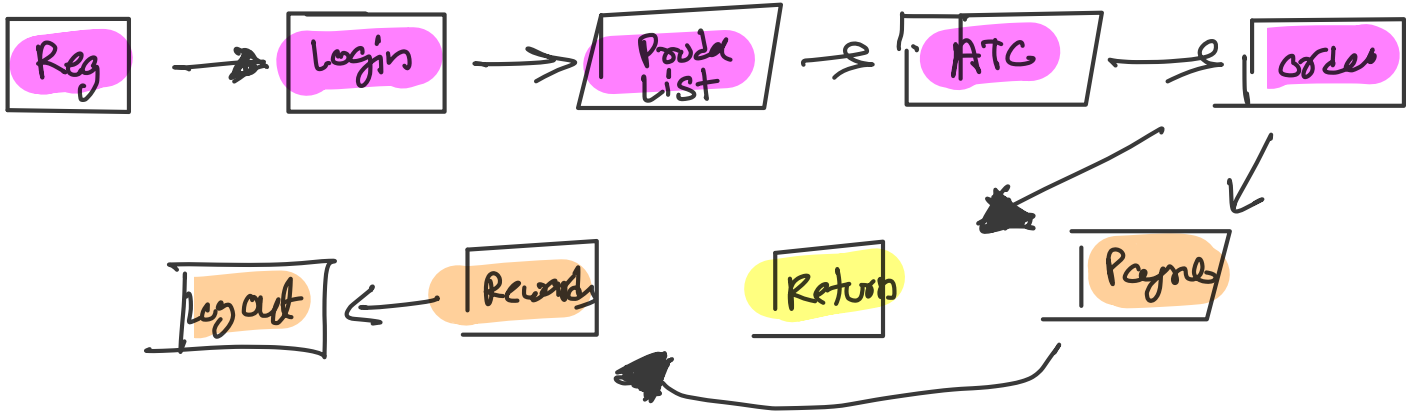
4. Throw error

Exercise for Positive/-ve TC



End to End testing

* Testing overall Functionalities of system including data, modules integrated



Globalization Testing	Localization Testing
Focus is on testing the overall capabilities of the application assuming a generic end user base	Focus is on testing the application capabilities assuming a subset of users specific to a given locale or culture
Application or software is assumed to be compatible with different regional requirements without the headache of rebuilding the application from scratch for each locale	Testing itself is with the assumption of a localized implementation of the software, and is a subset of globalization testing Baidu.com
Hardware and overall platform is subjected to compatibility tests based on target regions where the product is to be launched Google.com FaceBook.com	There is more focus on input confirmation, displays, system adherence & UI specific to a given locale where the product will be used Consistency checks for messages & printed documentation also happens as part of localization testing
Generic process for formal bug reporting at an enterprise level	Bug reporting is localized considering tests are designed for a specific region
Aim is to detect potential problems in the application code, architecture or design which could inhibit global implementation or launch	Validates applicable resources at a localized level, checks for local language accuracy and typographic faults
Messages or information dissemination is separated from the source code of the application Translators are separated from testing team resulting in impartial & just approach i18N-testing	This is not a requirement with local testing The time required for localized testing is much reduced as compared to globalization testing considering assurance activity is only to be performed for a specific locale at a given instance L10

Test Design Techniques

Why TD ?

→ Reduce data & Improve Coverage

* BVA
* ECP
* DT

* ST
* EG

-ve tc for Whatsapp

→ Igb Fill

→ . Inval / Inval no. d

→ 3b + photo

→ Invali d photo →

→ 300+ in Grup

→ OTP - Per

→ admin - By Not d?

→ . text Limit

→ XYZ.Fdf

→ Grup

→ Invalid