Testing (Quality = Level of Reliability)

* **Testing** : Testing the functionality of an application in order to find the defects according to customer requirements is called software testing.

↳ Manual testing is nothing testing functionality of application manually again & again in order to find defects according to customer's requirements.

* **Advantages** :

① in order to give quality software to customer

② Automation testing can be performed only if software is stable and stability can be checked from manual testing.

③ It is not possible to cover all random/ negative scenarios of application using automation testing.

* **Disadvantages** :

① Time Consuming

② It is a tedious job since we test software again and again.

③ Automation testing doesn't require a person to be present.

* **Software development life Cycle** : standard / step by step procedure to develope a software (RFDCTIM)

① **Requirement Collection** : Product Analyst in product based company and Bussiness Analyst in service based company gathers the requirements and conveys to techical team.

② Feasibility study : whether to take up a project or not
↳ whether they have proper resources
↳ whether they have proper lab setup
↳ whether they have proper techonology to implement

③ Design : It is done by Arthitect
↳ High Level Design : How the application will look like
↳ Low Level Design : It is nothing but detailed view of a module in application

④ Coding : by referring LLD

⑤ Testing

⑥ Installation/Implementation

⑦ Maintainance : Done by developers & QA

* Waterfall Model : Basic/Traditional/Sequential Model

* Stages : Requirement collection → Feasibility study → Design → Coding → Testing → Installation → Maintainance.

* Advantages : Good for small projects

① Chances of finding a bug is less since requirement changes are not allowed.

② Initial investment will be less since testers are hired at later stages.

* Disadvantages:

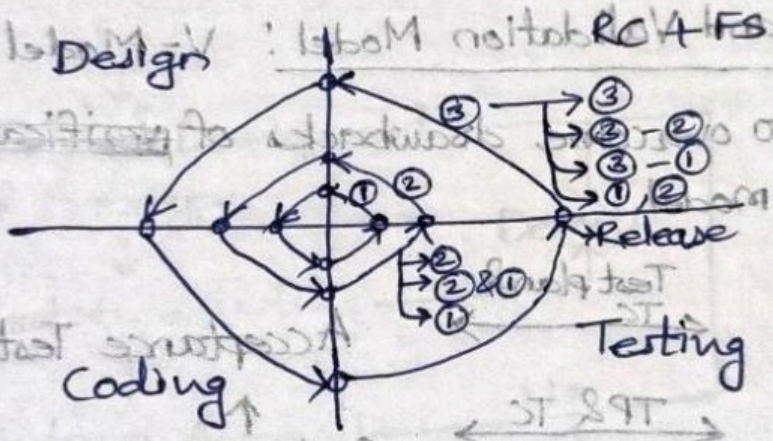
① Requirements can't be changed after feasibility study stage and testing will start only after coding.

② If there is any bug in inception stage itself, it will flow to end.

③ Time Consuming and More investment is required.

* Spiral Model: to overcome drawbacks of waterfall model

↳ used when there is dependency between modules.



↳ It is an incremental and iterative model.

* Advantages:

① Requirement changes are allowed after every cycle.

② It is called 'controlled model' to adopt because we take up another module only if first one is stable.

* Disadvantages:

① Every cycle of spiral module looks like waterfall model

② Requirement changes are not allowed in between cycles

* Stages of Requirements:

① Customer/Requirement Specifications (CRS) → Want to build an application
Bussiness

↓ BA will convert to

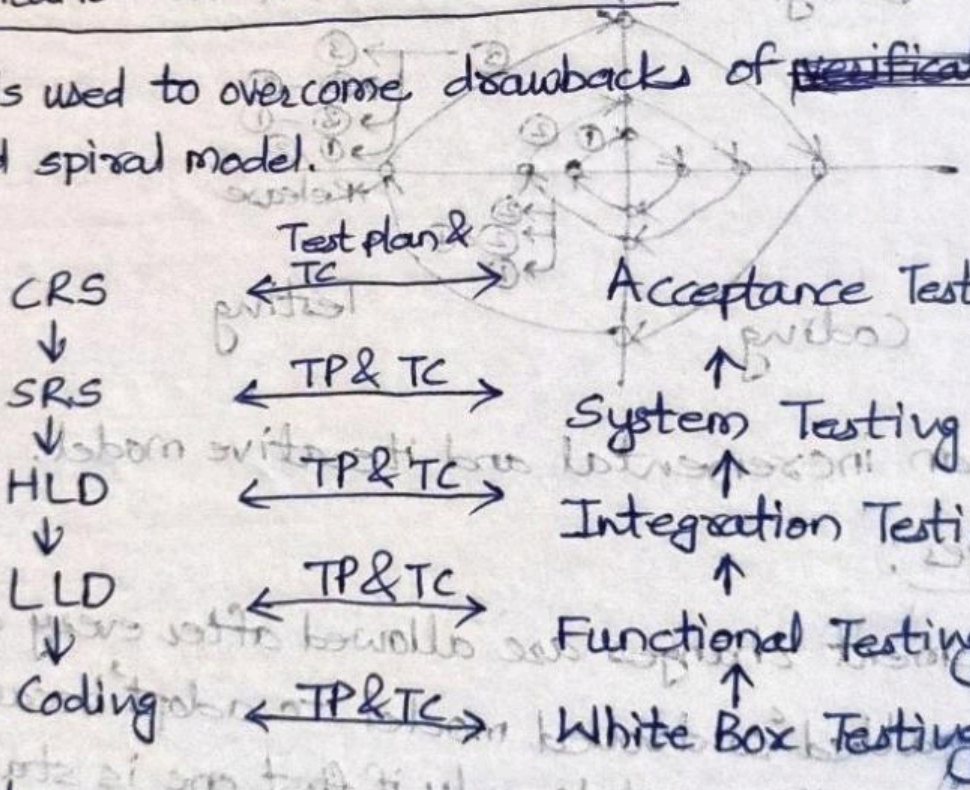
② Software Requirement Specifications (SRS) → What should be present in application
eg: Reg. form should have

↓ TA (technical Architect) will convert to

③ Functional Specifications (FS) → Detailed view of SRS
eg: Name text box should accept 10 characters

* Verification and Validation Model: V-Model

↳ It is used to overcome drawbacks of ~~verification~~ waterfall and spiral model.



* Advantages:

① every stage is tested unlike waterfall & spiral model.
② Downward flow of defect will be less. time consumption will be less.
③ Dev & Test team work in Parallel

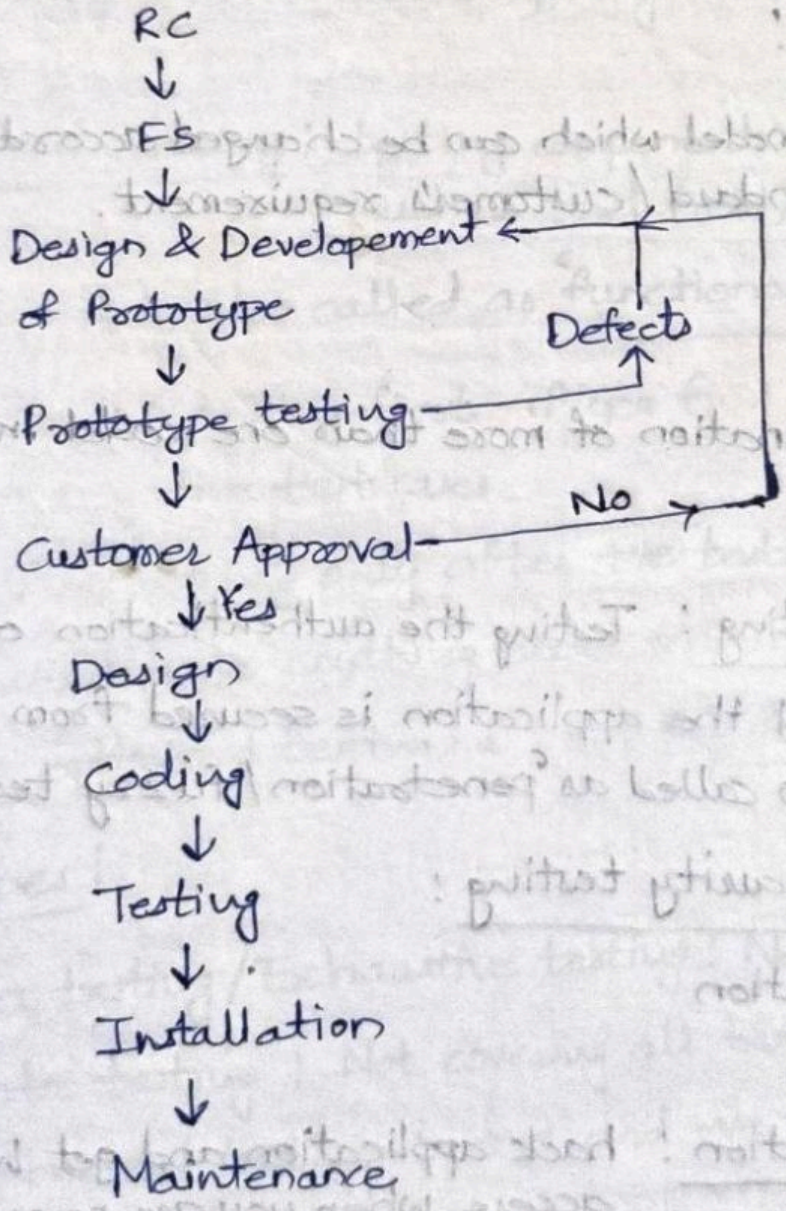
* Disadvantages :

① Documentation will be more because we are writing TP & TC at every stage.

② Initial investment is more.

* Prototype Model : Dummy model by web developers, who converts text format into image format

↳ The process follows as :



↳ New customers in market follows this model.

* Advantages :

① Customers get to see software earlier.
② Requirement changes are allowed.

* Disadvantages :

① It is a time consuming.
② Cost will be more.

* Derived Model :

↳ It is a basic model which can be changed according to company's standard / customer's requirement.

* Hybrid Model :

↳ It is a combination of more than one model into a single model.

* Web Security testing : Testing the authentication of an applicati to test how well the application is secured from unauthoriz users. it is also called as 'penetration / fuzzy testing'

↳ types of web security testing :

① URL Manipulation
② Session Expiry
③ Privilege Elevation : hack application and get higher role access when you are normal user level
④ Sql injection

⑤ Cookies based testing
↳ session cookies : stored in browser itself
↳ Persistent cookies : stored in memory of computer and it won't be deleted automatically

⑥ CSRF : Cross-Site Request Forgery

↳ we will check if url is encrypted or not.

⑦ XSS : Cross-Site Scripting

↳ Getting info of user/anything from fraud link/pop-up.

* Component Testing : testing components of an application such as text field, radio button, tab etc

↳ Rules : it is also called as 'functional testing'.

① Do Positive testing first. if you find any defect, don't test negative test cases
② Do -ve testing only after +ve testing is completed.
③ Don't assume anything about requirement.
④ Do optimized testing i.e. both +ve & -ve testing.

↳ Types :

① Over testing / Exhaustive testing : No need to check some cases
② Under testing : Not covering all test cases
③ Optimized testing : +ve and -ve testing

* Unit testing : it is where developers writing code to check their loops, statements, conditions, paths itself in code.

* Integration testing : testing functionality flow between two or more modules

↳ Types :

① Non-incremental integration testing : where all modules are combined and tested. it is also called as 'big-bang integration testing'. it is suitable for small modules.

② Incremental Integration testing :

↳ Top-down
↳ Bottom-Up
} sandwiched integration testing

* System Testing : End to End testing i.e. testing all modules

↳ It will have similar server, software, hardware, data as of production.

* Types :

① Performance Testing
② Load Testing :
③ Stress Testing :
④ Scalability Testing

* Acceptance Testing: End to End testing by customer to verify that software works properly for all business scenarios.

↳ It is also called UAT/FAT (Final Acceptance Testing)/ RBT (Red Box Testing)

* Reliability Testing: Testing the functionality of an application continuously for a particular period of time (Giving same results on successive totals)

* Recovery Testing: Testing the application to check how well application recovers from crash/ disaster. (Error testing)

eg: Restore tabs in google chrome

* Smoke Testing: Testing the basic and critical features of application before doing thorough and regression testing.

↳ It is also called as Build verification testing/ Confident testing/ Dry Run testing/ Scheme Testing/ Sanity Testing/ Health-check-up of the product/ Positive testing/ Shallow & wide testing.

↳ Types:

① Formal Smoke Testing: set of smoke tcs defined to test in advance

② Informal Smoke Testing: No Documentation/ tcs written in advance

↳ Smoke Testing: whether the build is acceptable or not for 1st time

↳ Sanity Testing: When bug in system testing is fixed and given back to testers.

* Adhoc Testing: Testing the application randomly without looking into application and requirements

↳ It is also called as 'Negative testing / Monkey testing / Gorilla testing', the purpose of adhoc testing is to verify that developed software is able to withstand any kind of business scenarios.

↳ it mostly involves testing of '-ve' test cases.

* Usability testing (Accessability testing): Testing the user friendliness of an application

① Easily accessible eg: present on play store
② Easy Language & easy to understand
③ Proper placeholder name in every field eg: first name
④ Throw pop-up / message when wrong data is entered
⑤ Simple Navigation
⑥ Tooltips should be present wherever necessary.

* Exploratory Testing: It is something which you don't know but you are trying to know how exactly your software works when you don't have requirements / requirement is missing.

↳ Drawbacks:

① You might misunderstand a feature as bug and vice-versa.
② Time Consuming

↳ Get knowledge of product to overcome the drawbacks.

* Regression Testing : It is nothing but testing the unchanged feature of application to make sure that changes like adding a feature, deleting a feature, modifications done in application or fixing a defect is not impacting the unchanged features of application.

↳ Depending on the changes done by developers, regression testing types are:

① Unit regression testing : testing only changed/modified part of application.

② Regional regression testing : testing the changed feature and impacted areas also.

↳ impacted areas are found out in 'impact analysis meeting'.

③ Full regression testing : testing the changed feature and all other features irrespective of impact on it.

↳ Full regression testing is done when changes are done at root level / most of areas or environment is changed.

* Retesting : It is nothing but retesting / testing earlier bug whether it is fixed or not. It is also called as confirmation testing.

↳ if the defect is fixed, status will be set to closed. otherwise, status will be set to 'reopen'.

↳ Retesting takes priority over regression testing.

↳ Only regression test cases are automated.

↳ Retesting is done on ~~passed~~ failed tcs whereas regression is done on passed test cases.

* static and Dynamic testing : DT is also called as validation

↳ static testing is done as a part of verification where dynamic testing is done as a part of validation.

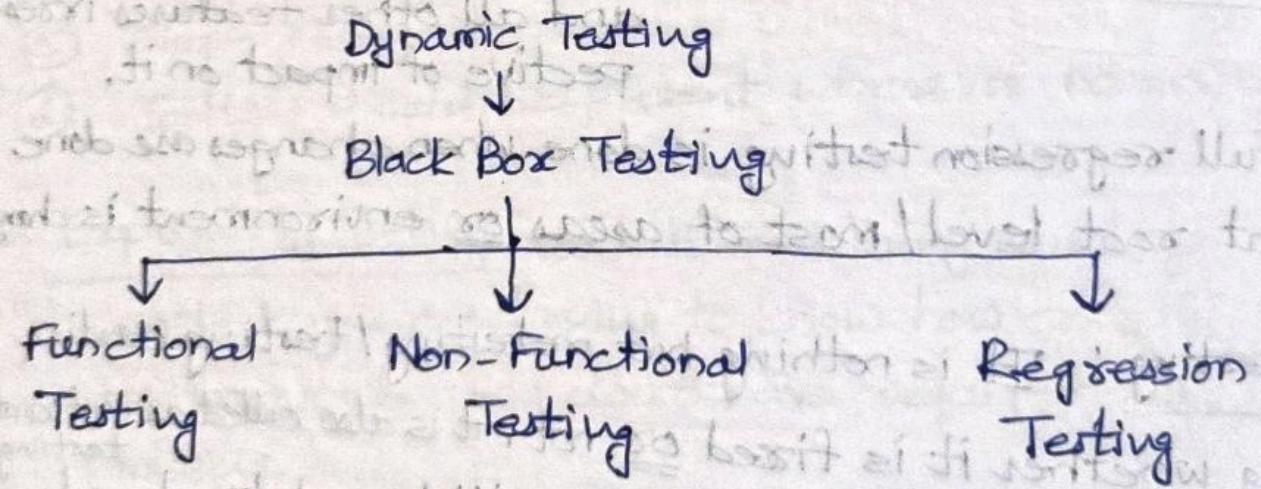
↳ ST is done before code is deployed whereas DT is done after code is deployed.

↳ ST is done to prevent the defects whereas DT is done to catch the defects and fix it.

↳ ST is less costly whereas DT is more costly.

↳ ST is done at early stage of developement whereas DT is done at later stage of developement.

* Black Box Testing : Testing functionality of an application without looking into internal code



* Smoke and Sanity Testing :

① Smoke Testing is also called as 'shallow & wide testing' whereas sanity testing is called as 'Narrow & deep testing'.

② Smoke testing is done to check if critical functionalities are working fine or not whereas sanity testing is done to check if new functionalities / bug fixed is working or not.

③ Smoke testing → Generalized health checkup
Sanity testing → Specialized Health checkup (subset of regression testing)

④ smoke tcs are automated whereas sanity tcs are not automated.

⑤ Documentation is done for smoke testing but it is not necessary for sanity testing.

⑥ Smoke testing is +ve testing whereas sanity testing is both +ve & -ve testing.

⑦ Smoke testing is done by both developers & testers whereas sanity testing is done by testers only.

* Positive and negative testing :

① +ve testing is done for mentioned requirements whereas -ve testing is done for tcs not mentioned in requirement document.

② 100% test coverage is not possible with only '+ve' tcs. it is possible when -ve tcs are added.

③ +ve testing doesn't ensure defect free product but -ve testing helps to ensure defect free product.

④ first +ve testing and then -ve testing is done.

⑤ +ve testing requires less time but -ve testing requires more time.

* Verification and Validation :

↳ Verification is static testing and validation is dynamic testing.

↳ Verification includes checking requirement documents, design, codes and programs whereas validation includes testing and validating actual product.

↳ Methods used in verification are reviews, walkthroughs, inspections and desk-checking whereas methods used in validation are black-box testing, white box testing.

↳ QA team does the verification and validation is done on software code execution with help of testers.

↳ verification comes before validation.

↳ verification confirms whether we are building the product right or not whereas validation confirms whether built product is right or not.

↳ cost of fixing bugs is less in verification since bugs can be found out in early stage whereas cost of fixing bugs is more in validation stage.

* Agile model / process / methodology :

↳ It is an iterative & incremental approach where requirements keep on changing and fast delivery of product to be done in short span time.

↳ the principle of agile model (advantages) are :

① Requirement changes are allowed at any stage of developement.

② Releases will be very fast.

③ Customer satisfaction by delivering piece of code in short span of time.

④ Good Communication betn customers, BA, dev & testing team.

⑤ Daily meeting for tracking process.

⑥ Easy to adopt model

* Disadvantages:

① Less focus on design and documentation

② Difficult to handle long-term project

③ Less scope for junior dev./testers since senior dev/testers are decision makers.

* Scrum: Rules to be followed while following agile model to develope a software i.e. framework for managing agile process

① Product Backlog: High level requirements listed by product owner

② Sprint, sprint planning, sprint retrospective meeting, Bug triage meeting, sprint review.

③ scrum master, scrum meeting (daily standup), epic, stories (feature of a module), story points, swag (it is similar to story points where efforts are measured in hours)

* Principles of software testing :

① Early stage testing
② Test application to find out defects.
③ It is impossible to give 'error-free software'.
④ We should not do 'exhaustive testing' i.e. entering same kind of values multiple times.
⑤ Testing is 'context-based' i.e. planning testing depends on type of application (web based / mobile based)
⑥ Pesticide Paradox: same set of tcs will not be able to find more defects / fix defects in longer run. tcs should be updated accordingly.
⑦ Follow the concept of 'defect clustering' i.e. know the impacted areas in order to avoid operational failure of application.

* Software Test Life Cycle : standard procedure to test any software

↳ STLC is a part of SDLC, whereas defect tracking is a part of STLC.

Requirement Collection
↓
System Study (Analyze requirement with BA, technical team)
↓
Test Plan (Members (testers), tools, strategy
↓

Write test Cases (System study → Identify Possible scenarios → write test case → Review → Fix Review → Approval → upload tcs into repository)

↓

Traceability Matrix (Forward TM (before execution), Backward TM (after execution), Bidirectional TM)
(Cross-Reference Matrix)

↓

Test Environment Setup

↓

Execution

↓

Defect Tracking

↓

Test Case Execution Repost (Pass, fail, defects)

↓

Retrospective meeting / Post Mortum meting (conducted by test manager)

* Defects / Bug / Error / Failure :

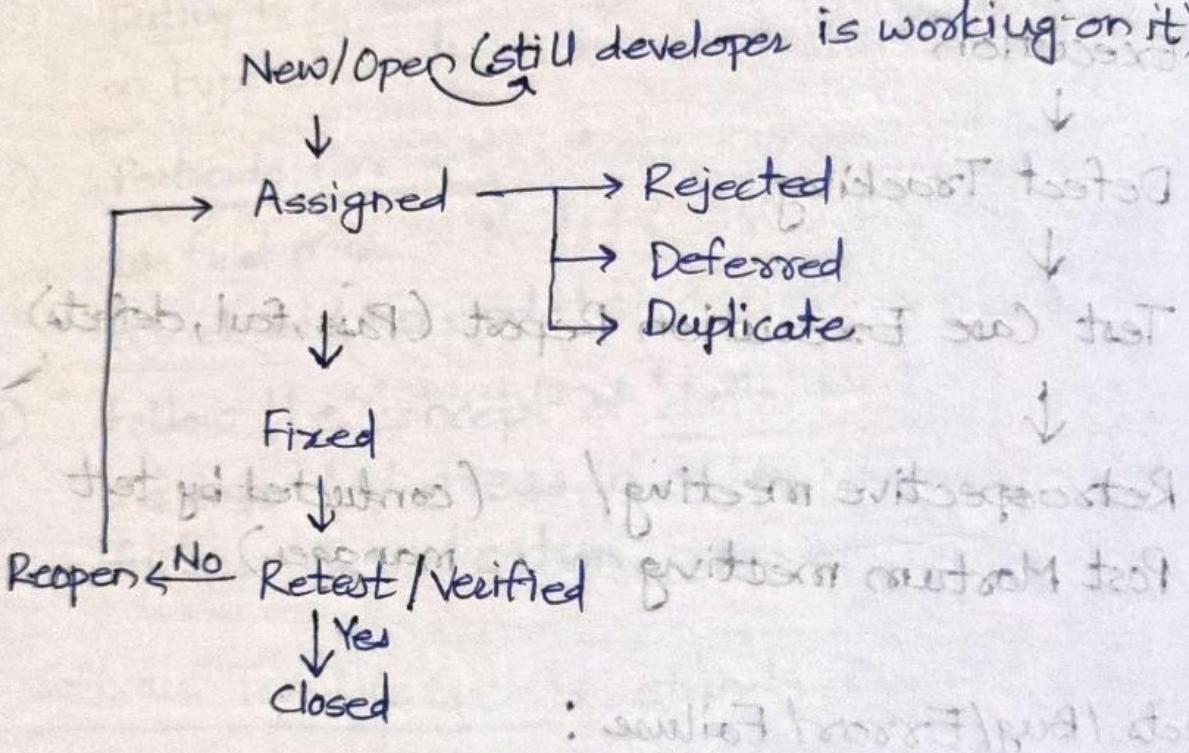
① Defect : aberration from actual & expected result.
② Fault : a state that causes software to fail to acomplish task.
② Bug : Informal name for defect / defect accepted by developer which was reported by tester.
③ Error : Mistakes done in program while writting code due to which code is not compiled
④ Failure : issue faced in the production by end user

* Defect / Bug Life cycle :

↳ Reasons for defect :

1. Wrong Implementation
2. Missing Implementation
3. Extra Implementation

↳ Defect life cycle :



* Severity and Priority :

↳ Severity : impact of defect/bug on customer business workflow.

↳ Priority : Importance given to defect/bug to fix it.

↳ testers decides the severity & priority but product manager/ dev. can change 'priority' of bug.

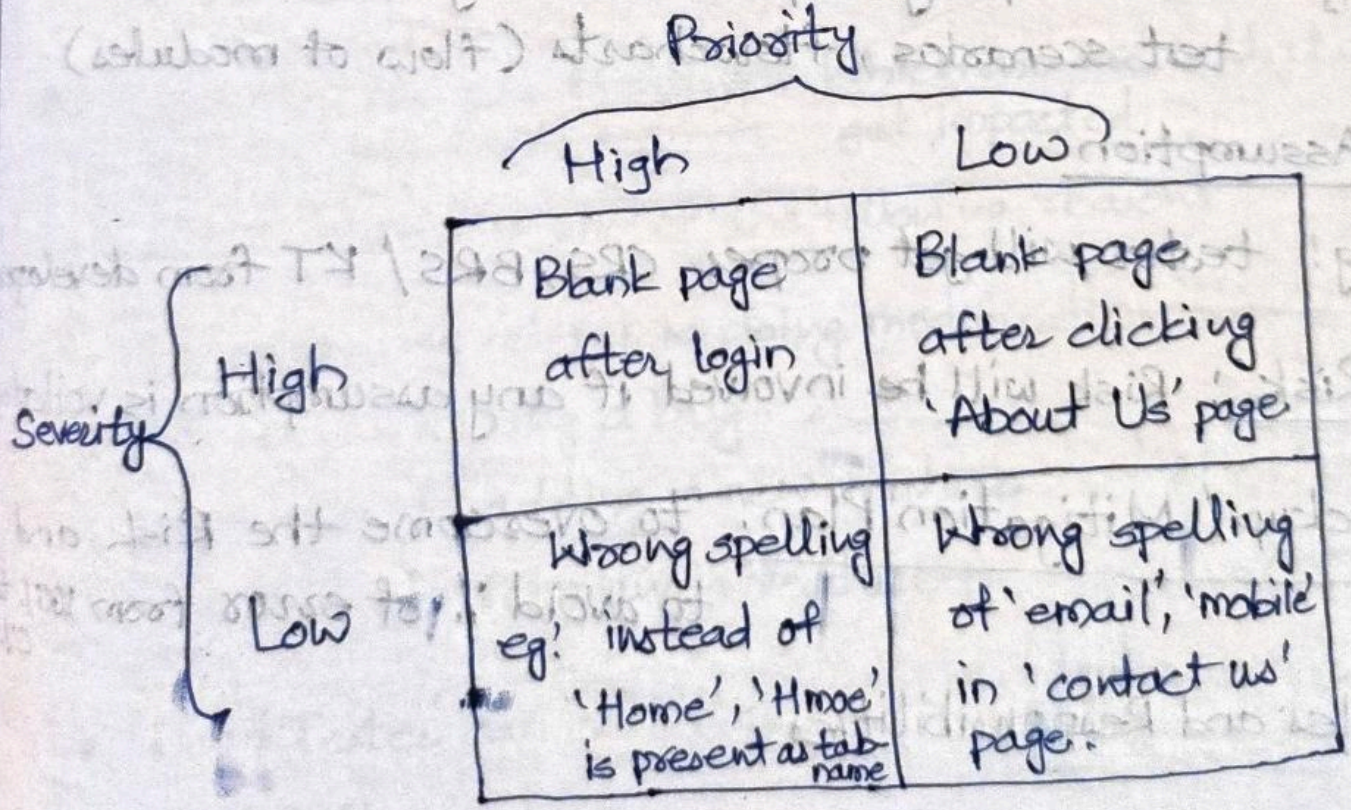
* types of severity :

1. Blocker
2. Critical
3. Major
4. Minor
5. Moderate
6. Cosmetic

* types of Priority :

1. High
2. Medium
3. Low

* Severity & Priority :



* Test Plan : Document consists of all future testing activities prepared by test lead/manager

↳ There are total 15 attributes :

① Objective : Aim of writting test plan

② Scope : What features will be tested and what won't be tested

③ Testing Methodology : Depending on type of application decide testing flow.

eg: Smoke testing, component testing, integration testing, system testing, acceptance testing.

④ Approach : it tells the way we go about and test application in future.

eg: some company expects test engineers to write test scenarios, flowcharts (flow of modules)

⑤ Assumption

eg: tester will get proper CRS, BRS / KT from developer

⑥ Risk : Risk will be involved if any assumption is violated

⑦ Backup / Mitigation Plan : to overcome the Risk and to avoid % of error from

⑧ Roles and Responsibilities

⑨ Scheduling : start & end date of every testing activity

⑩ Defect tracking

⑪ Test Environment / Test Bed

⑫ Entry and Exit Criteria : it is applicable to every testing activity
(Suspension and Resumption criteria)

⑬ Test Automation : Which tool, framework to be used, which features to be automated.

⑭ Deliverables : outcome from testing team given to client at end of project.

eg: TCs, RTM, Execution Report, Defect Report, Release Notes

⑮ Templates

* Impact Analysis Meeting : Which are areas that are going get impacted.

↳ IAM is done for one of the following reasons :

① Whenever developer is doing modification in code
② developer fixing a bug
③ developer is adding a new feature in application
④ developer is removing a feature from application

↳ 'Dev / Tester' can call this meeting along with product manager.

↳ It is mostly done for very old project.

* Build : Developer writes code, compile and compress

* Patch : A program that makes changes to software installed on a computer / A quick-repair job for a piece of programming designed to remedy functionality issues, improve security and add new feature.

* Bug leakage : A defect which exists during testing but not found by tester, which is eventually found by end user.

* Bug Release : A particular version of software is released with a set of known bugs / defects which are of low priority & severity in nature

* Hotfix : It is nothing bugfix for issue faced by end user in production on immediate basis.

↳ It is done for high priority & severity issue/bug

* Test Cases :

↳ Why do we write tcs :

① To avoid inconsistency in testing and remove dependency on person
② To have best test case coverage i.e. cover all possible scenarios
③ Increase capability to catch any defect
④ To avoid duplication and save testing time
⑤ To make it easy for new people for understanding & testing

* Quality of good tcs:

1. Coverage should be good and No duplicacy of tcs.
2. No. of steps to reproduce that test scenario should be minimal.
3. +ve tcs first and then -ve tcs.
4. anyone should understand it.
5. tcs should be written in tc template.
6. It should have capability to catch defects
7. conversion of manual tcs to automation scripts should be easy.
8. tcs should be designed using tc design technique.

* Test Scenario vs Test Cases:

1. Test scenario is high level documentation whereas test cases is low level documentation
2. Test scenario is just a short sentence whereas tcs will have various components and each component have multiple steps.
3. Execution of test scenarios will be uneasy for those who don't have any knowledge about system whereas anyone can execute tcs even if you have good project knowledge or not.
4. Test scenarios doesn't have data that you need to enter into application to test whereas tcs have it.

⑤ Test scenario tells us 'what to test' whereas tcs tells us 'how to test' an application.

⑥ Test scenarios are derived from requirement whereas tcs are derived from test scenarios.

⑦ ts helps us to validate the functionality of application whereas tcs helps us to validate ts is valid/correct.

⑧ ts requires less time to prepare whereas tcs takes more time.

* Test Case Review process / Peer Review Process:

Test case prepared by test engineer

↓

Review by TL/ senior or junior.. test engineer

will look for:
- → missing requirement
- → Wrong requirement
- → Extra/ Duplicate requirement

↓

Incorporate any change after review if any

↓

Approval by Test Lead (TL)

* Test C

① Bou

eg: inp th

② Eq

eg:

③

↳

↳

↳

* Test Case Design Techniques!

① Boundary Value Analysis

eg: input condition is valid betn 1 to 10
then boundary values : 0, 1, 2 and 9, 10, 11

② Equivalence Class Partitioning: data is divided into different equivalence class

eg: input conditions are valid betn 1 to 10 and 20 to 30
then -10 to 0 → invalid
1 to 10 → valid
11 to 19 → invalid
20 to 30 → valid
31 to ∞ → invalid

so, you select values from each class i.e. -2, 3, 15, 25 and 45

③ Decision table based testing:

↳ It is a way to deal with different combination inputs with their associated outputs.
↳ It is also called as 'cause-effect table'.

eg: user is navigated to home page if correct UN & Pass. is provided. otherwise, error message is displayed

UN	Pass	Outcome
F	F	Error
F	T	Error
T	F	Error
T	T	Home

④ State transition:

↳ Testing is carried out to observe the behaviour of application for different input conditions passed in sequence i.e. both +ve & -ve test input values.

eg: if user enters valid password in any of first three attempts, the user will be able to login successfully.

② user will be asked to enter password again if user enters invalid password in 1st or 2nd attempt.

③ Account is blocked if invalid password is entered for 3rd time.

or

user should be redirected to home page if user clicks on cancel button.

⑤ Error Guessing technique: based on experienced testers

* Types of Reviews:

① Informal: to detect potential flaws/defects, No check or documentation

eg: Buddy check, Pair review

② Walkthrough: to find defects, improve product, consider alternative implementation or evaluating conformance to standards & specifications.

↳ It is informal method

③ Peer/Technical review:

↳ It is formal method to check documents to detect and fix errors.

④ Inspection:

↳ It is nothing but thorough verification of documents.

* Root Cause Analysis:

↳ It is a structured & effective process to find the root cause of an issue in software to identify whether the defect was due to 'testing miss', 'developement miss' or 'requirement/design miss'.

↳ we can do RCA on production defects. based on the decision of RCA, we can enhance our test bed and include those production tickets as regression test cases. it will ensure same kinds of defects are not repeated.

↳ there are many factors which provoke the defects to occur:

① Unclear/missing/incorrect requirements
② Incorrect design
③ Incorrect coding
④ Insufficient testing
⑤ Environment issues (Hardware, Software/config.)

* RCA techniques:

① Fishbone / Ishikawa diagram (cause-effect analysis)

② 5 why's technique

* Types of root causes:

① Human Cause eg: under skilled

② Organizational cause eg: No monitoring tool in place

③ Physical cause eg: computer keep on restarting

* UAT:

- Alpha Testing: internal UAT (done by testers)
- Beta Testing: External UAT (done by client team/end user in beta environment)
- Gamma Testing: final testing before release

* Code Freeze: Development & testing is done and environment is stable. So, dev team stops checking in code

* what to do when you find a bug:

① Analyze the issue

② Identify the steps

③ Check the logs

④ Do Root cause analysis

↳ Stubs and drivers are used to test modules. stubs are used to test functionality of modules whereas drivers are used when main module is not ready.

* How to choose test cases for automation:

① Regression test cases: test product is stable or not

② Retesting cases: test cases which are repetitive

③ Automatable test cases: Functional doable test cases

④ Exclusion Criteria: Graphs, charts, images, captcha

* Globalization Testing: Process of validating whether a website or an app delivers a customized user experience to users across the globe. these tests include validating whether the website or app adapts the language, currency and time correctly based on location it is accessed from.

↳ Localization testing: similar to globalization testing but restricted to a small group of users for a specific geographic location or culture.

↳ Teradata: it is one of the RDBMS, suitable for building large scale data warehousing applications for large volume of data.

↳ Data migration testing: it is nothing but migration of application from old system to new system with data integrity, no loss of data and minimal downtime with functional & non-functional aspects are met post migration.

↳ Cause effect graph : it is a black box technique that represents graphically relation between outcome and factors that affect the outcome, which assist in design test cases.

↳ inter-phase testing : verifying communication and data transfer between two different systems where interface can be api or webservices.

eg: booking Airline ticket through Goibibo.com

↳ Concurrent testing : Group of people accessing software simultaneously to detect the defects

eg: one cognizant is used by HR dept, employee from various dept

↳ Behaviour driven testing : what should be system response when an error occurs.

eg: error message should be displayed when wrong value is entered in input box.

↳ installation testing : checking if an end user is able to install / uninstall product manually step by step. it could be full, partial, upgraded installation process.

↳ Comparison testing : testing that compares software weaknesses and strengths to those of competitor's product.

↳ Desk checking : it is conducted by developer of system. it involves reviewing the complete product to ensure that unit testing, requirements, coding standards are met.

↳ Defect Density: it is the no. of defects found in software product per line of code.

↳ Defect density metric: it not only indicates the quality of product being developed but it can also be used as a basis for estimating a no. of defects in next sprint.

↳ Soak/ Endurance Testing: Apply load on application continously in order to check the stability.

↳ Bug Release: It is a specific version of software which is released in market with some known bugs which are intended to get fixed in later versions. these types of issues are of low priority and are mentioned in the release notes while sharing with end-users.

↳ Defect Cascading: it is the triggering of a defect by another defect. it happens when a defect is not caught by the testing team and it gives rise to another defect.

↳ When should we stop testing:

① After test case execution ② once testing deadline is met
③ Based on code coverage value ④ Based on mean time between failure

↳ Components of Test strategy:

Scope and Objectives, Bussiness issues, Roles & Responsible, Communication and status reporting, test deliverables, industry standards to follow, test automation & tools, testing measurement & metrics, Risks & mitigation, defect reporting & tracking, change & configuration management, Training plan

↳ Test strategy is prepared first. it describes how testing is to be done. it is used at organization level & can't be changed it is prepared by project manager and has primary goal 'what techniques to follow and which module to check'.

↳ Test plan describes the scope, objective and weight on each testing task. it is used at project level & can be changed. it is prepared by test manager and has primary goal of 'what to test, when to test and who will verify'.

↳ thread.sleep(-1000) → illegal Argument Exception

↳ Exhaustive testing: Testing all functionalities using valid and invalid inputs and preconditions.

↳