* Why Software has bugs ?

→ Mis Communication (or) no Communication [b/w developers & Testers]

→ Software Complexity

→ programming Errors [developers]

→ changing requirements

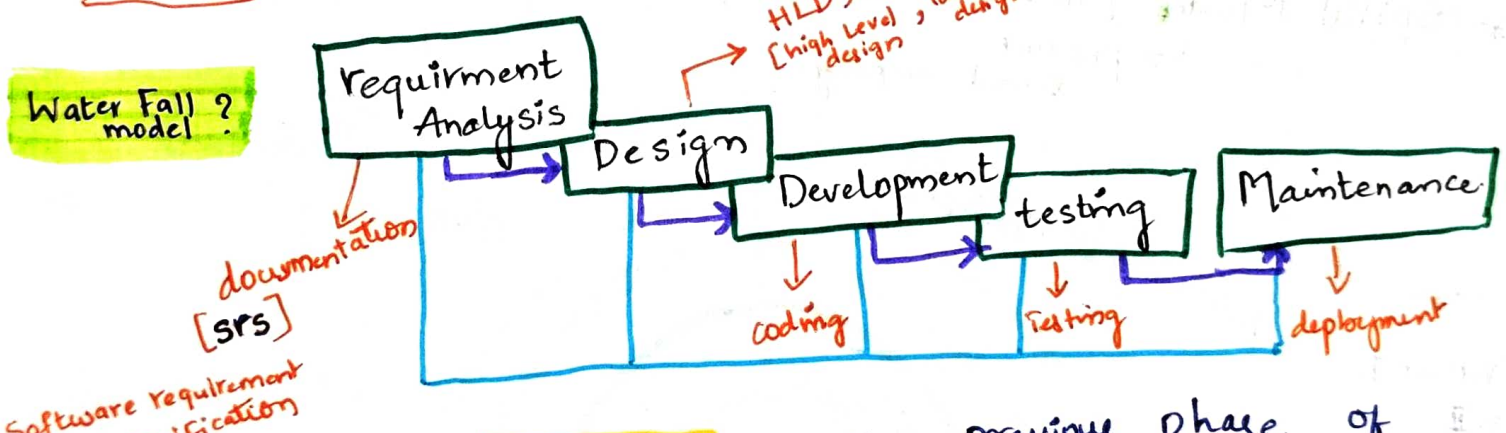
→ Lack of skilled testers

* what is SDLC ?

→ SDLC [Software Development Life cycle]

. it is process used by Software industry to design develop and test Software.

* Water Fall model :



→ Every phase is Dependent on the previous phase of the model.

Advantages :

→ Quality of product will be good : Detailed Documentation.

→ Since requirement changes are not allowed, Chances of findi bugs will be Less.

→ invest is Less, intial investment → Prefered for Small Projects

- disAdvantages of Waterfall Model :

→ requirement changes are not allowed.

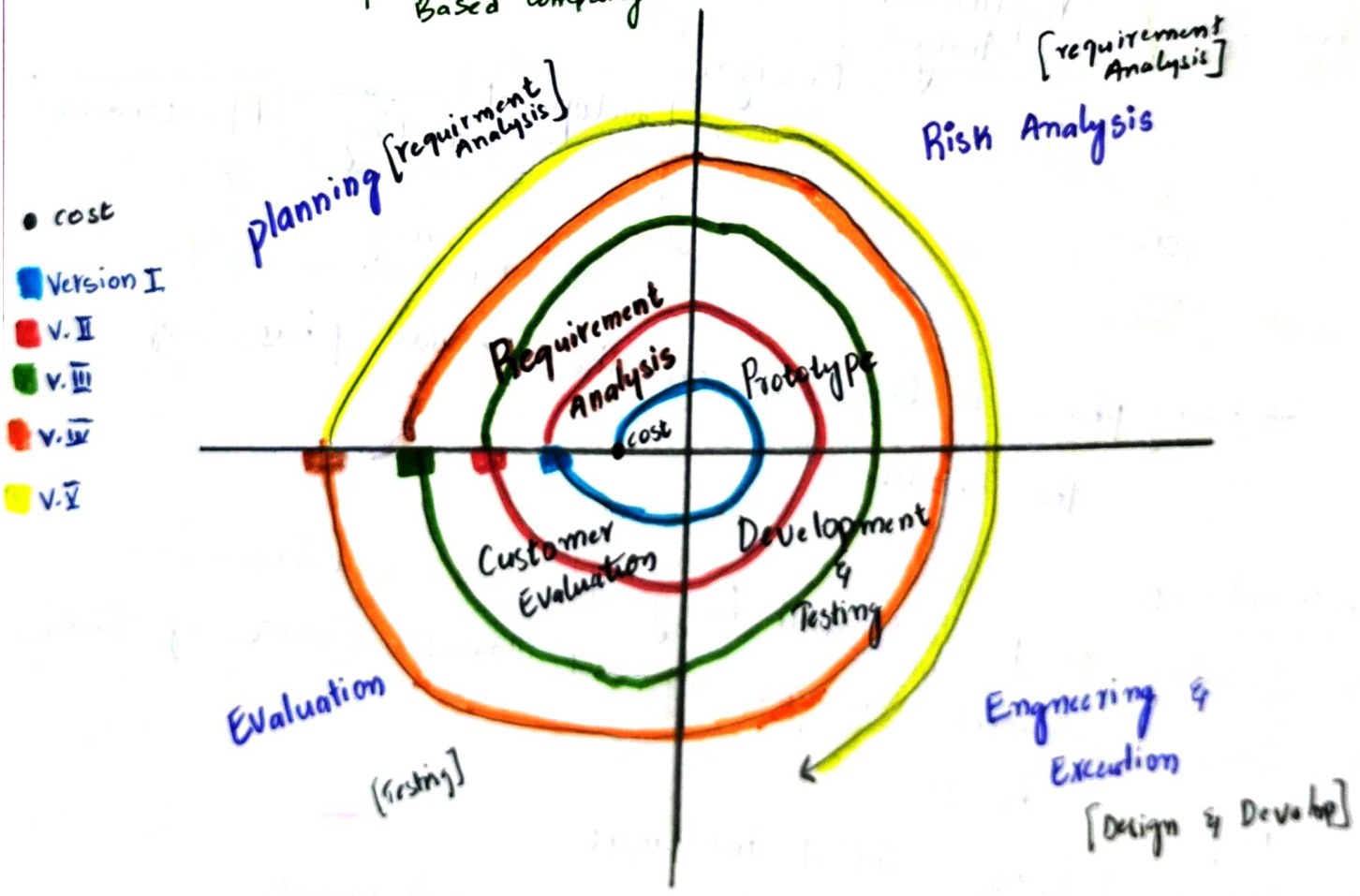
→ If any Defect in requirment that will be Continued in Later phases.

→ total investment is more

Because, Time taking for rework on defect is time consuming which Leads to high investment

→ tester involved in after completion of coding not from first.

* Spiral Model [iterative model] of SDLC
  ↳ product Based Company



→ Spiral Model is Repeatedly iterative model / Version Control model

→ Spiral model Overcome drawbacks of Waterfall model

→ We follow Spiral model whenever there is dependency on the modules. [Version 2 → dependent should be done on Version 1)

→ In Every cycle new Software will be released to Customer.

→ Software will be released in multiple versions. So it is called Version Control model.

Advantages of Spiral Model.

→ Testing is done in Every cycle, before going to Next cycle.

→ Customer will get to use, the Software for Every module.

→ requirement Changes are allowed, after every cycle before going to next cycle. (Not in the middle of cycle.)

disadvantages.

→ requirement changes are NOT allowed in between the cycle

→ Every cycle of Spiral model looks like waterfall model.

→ there is no testing in requirement and Design phase.

- **Proto Type** :

→ **Blue print** of Software

→ initial requirements from Customer

↓ ↑

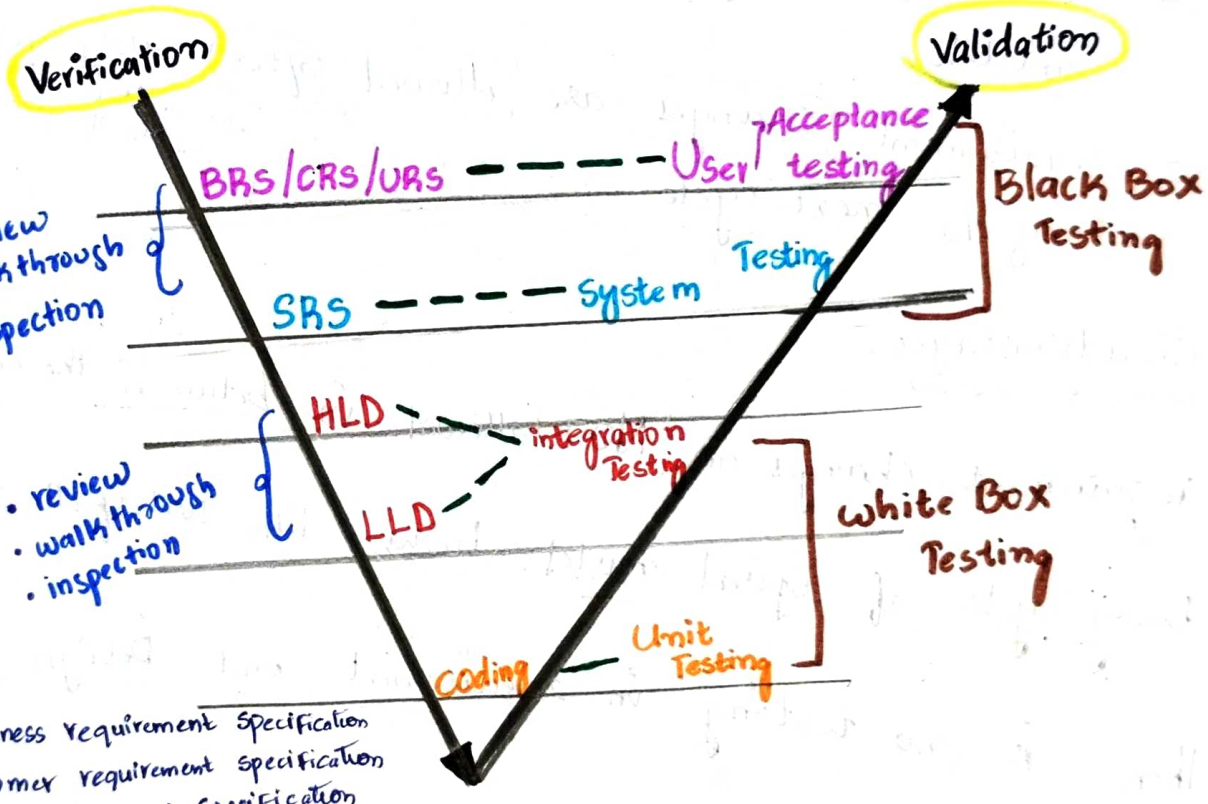
Prototype

→ it comes between the Water-fall model & Spiral model

- → Design
- → coding
- → testing

Water Fall Model ⟹ Prototype Model ⟹ Spiral Model

- **V-Model [or] V.V Model** [Verification & Validation Model.



- BRS = Bussiness requirement specification
  CRS = Customer requirement specification
  URS = User requirement specification
- SRS = Software requirement specification
- HLD = High Level Design
  LLD = Low Level design

* Verification

→ Verification checks whether we are building the right product

→ Focus on "Documentation".

→ * Static testing

→ Testing the "Project releated document" is called as "static testing".

- Review ✓
- walk through ✓
- inspection ✓

* Validation

→ Validation checks whether we are building Product right ✓

→ Takes places after Verification are completed.

→ Focus on "Software"

→ ~~Valid~~

→ * Dynamic testing

→ testing the Actual Software

- Unit testing ✓
- Integration Testing ✓

→ White Box Testing (developers)

[testers] Black Box Testing ←

- system testing ✓
- UAT [User acceptance Testing] ✓

*Black Box Testing

→ it is mostly done by the Software testers

→ No knowledge of coding required x

→ it is a Functional test of Software

* White Box testing

→ it is Mostly done by Software Developers

→ Knowledge of Coding require

→ it is a Structural test of Software, Logic needs to test.

Advantages of V model:

→ Testing is involved in Every phase.

DisAdvantage of "V" model:

→ Documentation is more

→ Initial investment is more,

Because, we need to hire developers and testers at a time before starting project

- **Static Testing Techniques** :- Testing Documentation.
  - * Review
  - * walk through
  - * inspection.

* **Review** done by anyone only single person is Enough
→ Conducts on Documentation to Ensure the Correctness & Completeness.

→ Requirement reviews
→ Design reviews → HLD, LLD
→ Code reviews → Dev1, Dev2 [Documents]
→ Test plan reviews
→ Test Cases reviews ---

- **Walk through** : Minimum 2 (or) 3 should be involved.

→ it is a informal review [No proper plan/schedule]
→ Author [creater of Document] reads the Documents (or) Code and Discuss with Peers (involves Mainly) & Explains
→ its not preplanned and can be done whenever required
→ Also walkthrough doesn't have minutes of meet.

- **Inspection**

→ it is most formal of review type
→ at least 3-8 people will sit in meeting
  1- reader [author of Document]
  2- writes [writes down question/clarifications discuss in meeting]
  3- moderater [Anchor / mediator] organises]
→ Inspection will have proper Schedule which will be intimated Via Email to Concerned developer/tester.

Quality Assurance — Quality Control

Difference blw QA & QC

* P - People [QC - TEsters]
* P - Process [QA - high level Managing people]
* P - Product

} of Company

SDLC

- requirement Analysis QA
- coding } QA
- design } QA
- Testing QC
- Deployement QA
- Maintance QA

QA

- QA is "process" Releated
- QA Focuses on "building in quality"
- QA is "Preventing" Defects
- QA is "Process oriented"
- QA for "Entire life cycle"

QC

- QC is "actual testing" of Software
- QC focuses on "testing for quality"
- QC is "Detecting" defects
- QC is "product oriented"
- QC for "testing" part in the SDLC.

QE — Quality Engineer

- it is a process that applies "rigorous quality checks to Each Stage (or) product development.
- it Does Through Analysis, Development, Management, Maintance
- Through QA Protocols. it often Continued after product has been Delived.

Levels of testing

- Unit testing
- integration testing] developes
- System testing
- User Acceptance Testing [UAT] — Testers

• Unit Testing

→ A Unit is Simple Component or Module of a Software

→ Unit testing Conducts on a Single program (or) Single module.

→ Unit testing is White box testing Techique

→ it is Conducted by developers

● Techinques in Unit Testing :

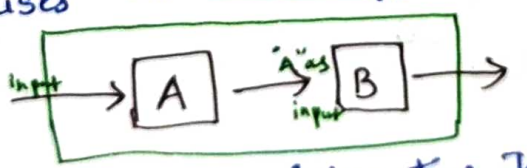
- Basic path testing :- Every line should be Executed Once.
- Control structure testing
  - Conditional coverage :- if & Else conditions are True/False Checks
  - Loops coverage : Verify loops.
- Mutation testing : Repeation

Automation : JUnit

• integration Testing

→ integration testing Performed between 2 (or) More Modules

→ it Focuses on checking data Communication b/w Multipk Modules



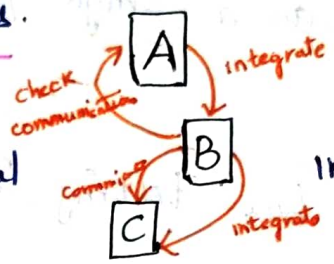
- Tester also does integrating Testing by UI
- Developes does by Coding

- Types of Integration Testing

  * Incremental integration Testing
  * Non - Incremental integration Testing.

* Incremental integration

→ Incrementally adding Modules and testing the data-flow between modules. One After Another is called Incremental integration testing



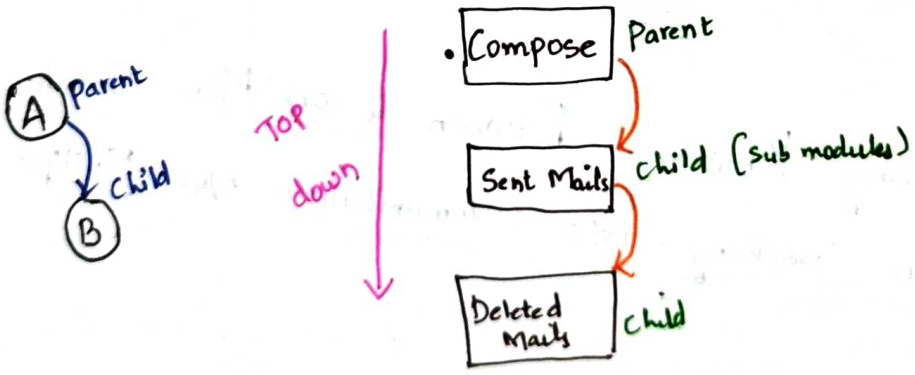
→ 2 Approaches:

- Top Down Approach
- Botton UP Approach

* Top Down - Incremental integration

Incrementally adding Modules & testing the dataflow between the modules, Ensure the Module added is child of previous module.

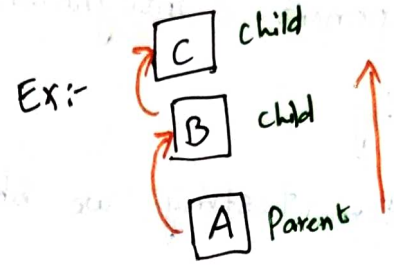
Ex:- gmail Application



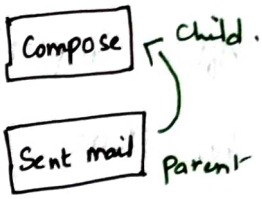
* Bottom up - Incremental testing

Incremental integration testing in Bottom up is that "Incrementally adding modules and testing the data flow between Modules", And Ensure module added is the Parent of previous module.

Ex:-



Gmail



* Sandwich / hybrid Approach

- Combination of Top Down & Bottom up Approach is called Hybrid / Sandwich Approach

* Non - Incremental integration Testing

Adding all modules in a single shot and test the Data flow between modules.

⇒ Drawbacks :

- We might miss Data Flow b/w Some of modules
- If you find any defect we can't Understand the root Cause of defect

21/12/22

• System Testing

→ Testing Overall functionality of the application with respective client requirements.

→ it is Black Box testing

→ This testing is conducted by testing team.

→ After Completion of Component & integration Level testing's We start System testing

→ Before Conducting System testing we should Know the Customer requirements

* System Testing Focuses Mainly

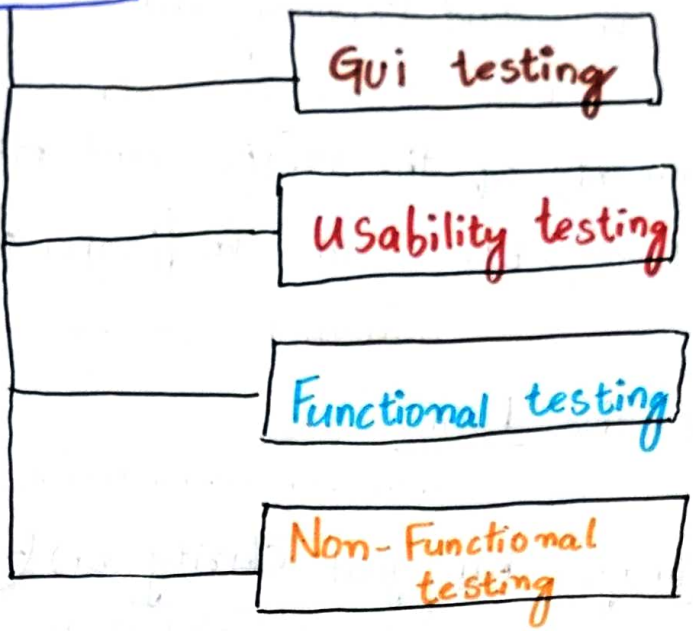
1. User InterFace Testing [GUI]
2. Functional testing
3. Non-Functional testing
4. Usuability testing.

• User Acceptance Testing [UAT]

After Completion of System testing UAT team Conducts acceptance testing in two Levels

1. Alpha testing :- User/test in development Environment.
2. Beta testing : They Test after installation [customer] basic testing

* **System testing** :- Testing Overall Functionality of a System.



* **GUI testing**

- Graphical User-interface [GUI] is a process of testing the user interface of an application.
- it includes all the elements such as Menus, Checkbox, buttons, Colours, fonts, sizes, icons, Content and images

information : design document
FRS
[wire Frames] dummy

- GUI testing Checklist

→ testing the size, position, width, height of the elements

→ testing of the Error messages that are getting displayed

→ testing the different sections of the screen
↳ Logo, Footer, header ...

→ testing of the font whether it is readable or not

→ testing of the screen in different resolutions with the help of Zooming in and Zooming out

∴ Elements should not over lap

→ testing the alignment of the texts and other Elements like icons, buttons, Etc.., are in proper place or not

∴ According to requirement, Exact place should be Available.

→ testing colours of the Fonts.

∴ Avoid Dark (or) Harmful colours to Eye/should be cool.

→ testing whether image has good clarity (or) not

∴ high pixels of images should be used in less size (mb)

→ testing the alignment of the images.

→ testing of the Spelling ∴ without spelling mistakes.

→ User Must not get Frusted by using System interface

∴ should be clean, user Friendly, clean fonts should be used

→ Testing interFace is Attractive or not

→ testing of Scroll bars, according size of the page, if any

∴ if page is small, scroll bars will n't appear.

→ testing of disabled Fields. if any

∴ we can't Enter any Data in that

→ testing Size of images

∴ should be low, because, it effect loading time. size (kbs)

→ testing of headings whether it is properly alligned or not

→ testing of the Color of the hyperlink

∴ color should be changed, after clicking once.

→ testing UI Elements like button, textbox, text area, Check box

Radio buttons, drop down, links Etc

textbox → provide a small text

text area → we can provide huge text, Ex: Address ...

Check box → Select only one / multiple. Ex: Female ○ male ●

Radio buttons → only one

drop down → 

• Usability Testing

↓ Easyness of application.

→ During this testing Validates application provided Context Sensitive help (or) not to the User. ∴ help menu

→ Checks how easily the end users are able to Understand and operate the application is Usability testing.

• Functional Testing

→ Functionality is nothing but, behaviour of application

→ Functional testing talks about how your Features should Work.

* Object properties testing
* Database testing
* Error handling
* Calculations / Manipulation testing
* Links Existence & links Excecution
* Cookies & Sessions

* Object Properties testing

Objet properties testing is that Check the properties of the object present on the application.

Ex:- Enable, disable, visible, Focus....

* **Database testing**

→ Checking database operations with respect to User Applications.

→ Mainly we perform on "DML operations" [Data Manipulation Language]

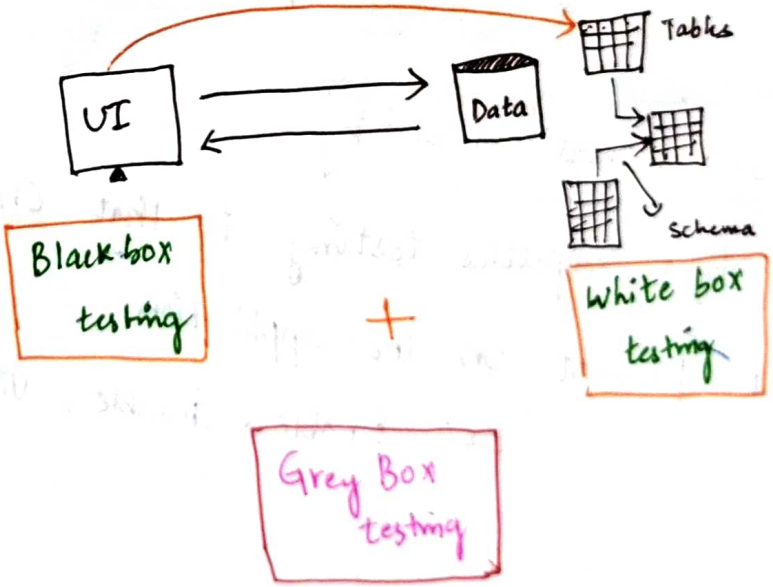
- Insert
- UpDate
- delete
- select

→ these are having both "white box testing" and "black box testing", so it is called as "Grey box testing" it is combination of both, 'white & black' box testing

* Advance Level of Database testing:

- Table & column Level validations [column type, length, no. of columns]
- Relation between tables [Normalization]
- Functions
- procedures
- Triggers
- Indexes
- Views

etc...



* Error Handling

→ tester Verify the Error messages while performing the incorrect actions on the application.

→ Error messages should be readable.

→ User Understandable / simple Language

Ex:- Incorrect Data ✗
Invalid user ✓

* Calculations / Manipulation testing

→ tester should Verify the Calculations

→ tester give "n" no. of tests. while giving different types of combinations of number. +, −, *, ÷

* Links Existance / Execution :- For Webpages

→ Where Exactly the links are placed : Link Existence

→ Links are navigating to proper page or not : Link Execution

- Internal Links :- Opens in Same Page
- External Links :- Opens in another page
- Broken Links :- don't open [Kept For Development Furtur Use.

* **Cookies & Sessions** :- For Web pages / servers

→ Temporary Files created by Browser while browsing the pages through internet :- Cookies

• Sessions : For Servers.

→ Sessions are time slots created by Server. Session will be expired after some time [if no action on page]

• Non-Functionality testing : Expectation of client

→ Once the application Functionality is stable, then we do "Non-Functionality testing"

→ Focus on performance, load it can take & Security etc

* Performance testing
* Security testing
* Recovery testing
* Compatibility testing
* Configuration testing
* Installation testing
* Sanitation / Garbage testing

* **Performance testing** : For Web application

→ **Speed** of The **Application**

- Load
- Stress
- Volume

* **Load** : Increasing the Load on the application slowly, then check the **speed** of the application.

* **Stress** : Suddenly **increase / decrease** the Load on the Application and check speed of application.

* **Volume** : Check, how much **data** is able to **handle** by the application.

* **Security testing**

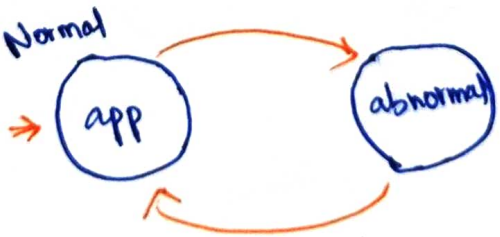
→ How secure our Application

- Aunthentication :- Users are **Valid / not**
- Autherization / Access Control :- **permissions** of Valid user

* **Recovery testing**

→ Check the system change to **abnormal** to **normal**

Ex:- gmail :- Compose → power off → saves in draft



* **Compatability testing**

→ Check The Compability of different Versions of the Software.

- Forward Compatability : $V_1 \rightarrow V_2 \rightarrow V_3$
- Backward Compatability : $V_3 \rightarrow V_2 \rightarrow V_1$
- Hardware Compatability [Configuration testing]

* **Installation testing**

→ Check Screens are clear to Understand.

→ Screen Navigations

→ Simple (or) not

→ Un-installations [removed Every File of application / not]

* **Sanitation / Garbage testing**

→ If any application provides Extra Features / Functionality then we Consider them as a Bug.

→ we should not provide other than the requirements

- Difference between :

Functionality testing	Non-Functionality Testing.
• Validates Functionality of Software	• Verify the Performance, Security, Reability of Software
• Functionality describes what Software does.	• Non-Functionality Describes how Software Works.
• Concentrate on user requirem	• Concrete on User Expectation
• Functional testing takes place before Non-Functional testing	• Non-Functional testing Performed after Finishing Functional testing.

Testing Terminologies

***

• Regression Testing

* Testing Conducts on Modified build. To Make Sure there wil not be impact on Existing Functionality.

Because of changes like adding / deleting / modificatio Features.

- Unit regression testing
- Regional regression testing
- Full regression

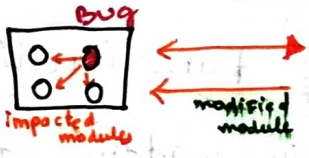
* **Unit Regression Testing**

→ testing **Only** the **changes** / modifications done will be tested by developer

* **Regional Regression**

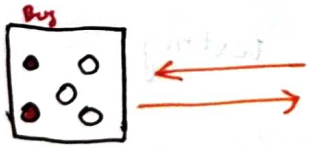
→ testing the modified module **along** with the **impacted modules**

→ Impact Analysis meeting conducts to identify impacted Modules with **Q/A** & **Developers**.



* **Full Regression**

→ testing the main Feature & remaining part of application.



Here we do **One round** of Full regression

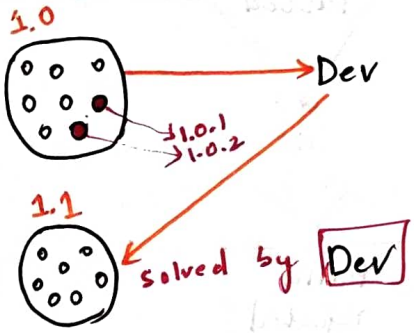
80% changes has done by developer while fixing Bug

• Re-testing :

→ whenever the developer Fixed a bug, tester will test the Fixed bug [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 Earlier build were Fixed or not in the Current build.

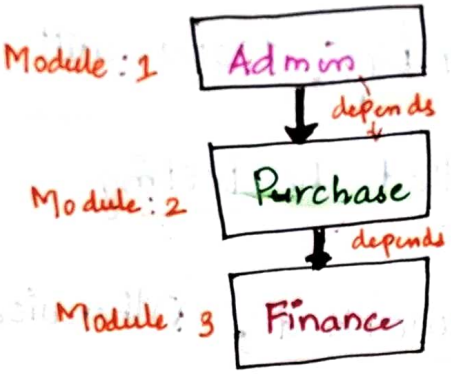


Ex:-

- Build 1.0 was released. Test team Found some defects [Defect Id : 1.0.1, 1.0.2] and posted to Dev.

- Build 1.1 was released, Now testing the defects 1.0.1 & 1.0.2 in this build is re testing

Example :-



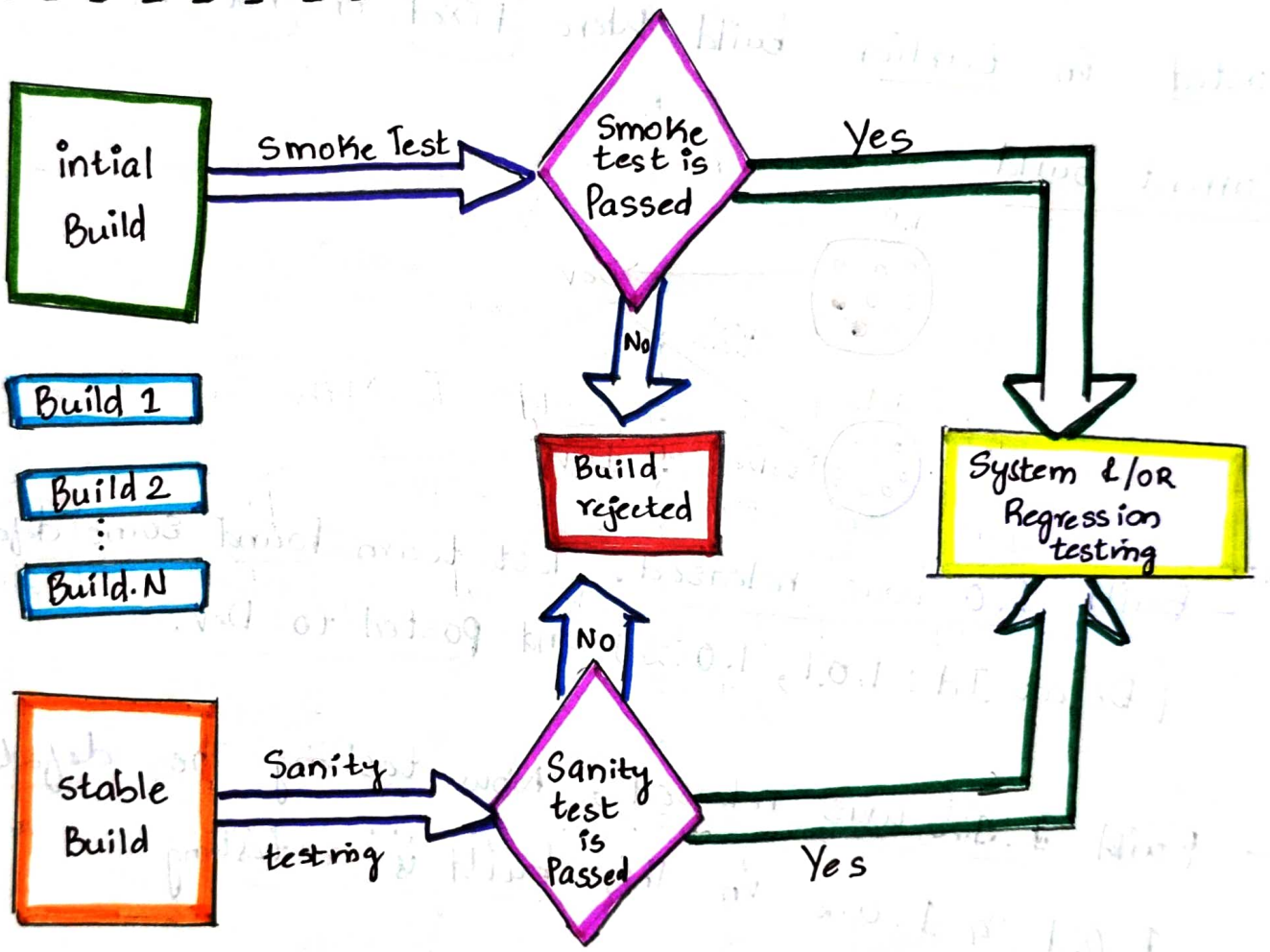
* If tester Found a Bug on Purchase Module and Posted.
* Once Bug is Fixed, the tester needs to do Retesting to Verify whether bug related to Purchase is Fixed or not and this is Retesting
* But, Regression testing to test the Finance module which depends on Purchase Module.

***

Smoke Vs Sanity Testing

→ Smoke & Sanity Testing Come into existance after Build Release

---



* Smoke & Sanity Working Flowchart.

Difference between :

Smoke testing	Sanity testing
• it is done on initial builds	• it is done on Stable builds
• it is part of basic testing	• it is part of regression testing.
• Smoke test is done to Make Sure the build we received From the development team is Stable / testable (or) not.	• Sanity test is done during the release phase to check for the main functionalities of application without going deeper
• Smoke test can be done by both Developers & testers	• Sanity testing is performed by Testers alone / only.
• smoke testing is performed while build may be Stable or Unstable.	• Sanity testing, build is relatively stable
• 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

22/12/22

• Exploratory Testing :

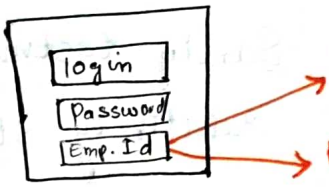
→ We have to explore the application, Understand Completely & test it

→ Understand Application, identify all possible Scenarios, docume it then use it for testing.

→ We do Exploratory testing when the application is ready but there is no requirements like SRS, LLD, HLD...

→ Drawbacks in Exploratory testing

Ex :



→ You might mis-understand any Feature as a Bug

(or)

Any Bug as a Feature since you don't have requirement

→ time Consuming

→ If there is any bug in application, you will never know about it.

• Adhoc Testing

[Randomly checking] Functionality

→ Testing application randomly without any test cases (or) any bussiness requirement document

→ Adhoc testing is an informal testing type with an aim to break the system.

→ tester should have Knowledge (Experience on) of application Even though he doesn't have requirements / test cases

→ this testing is usually an Unplanned activity.

• Monkey / Gorilla testing

[don't have Knowledge on application]

→ Testing application randomly without any test cases (or) any bussiness requirement document.

→ Adhoc testing is an informal testing type with in an Aim to break the system

→ Tester do not have Knowledge of application.

→ Suitable for Gaming application.

Difference between :

Exploratory Testing	Adhoc testing	Monkey testing
• No Documentation	No Documentation	No Documentation
• No plan	No plan	No plan
• informal testing	informal testing	informal testing
• Tester doesn't know application functionality	Tester should know application functionality	Tester doesn't know application functionality
• Random Testing	Random testing	Random Testing
• Intention is to Learn (or) Explore functionality of application	Intension is to breaks the application / find out Corner defects	Intension is to breaks ↵
• Any application which is new to Tester	any application	Gaming application

• Positive testing

→ Testing the application with "Valid inputs" is called Positive Testing

→ it checks whether an application behaves as Expected with positive inputs.

Ex: Enter only Numbers.
9999 Positive Testing

∴ Only numbers should be acceptable, not alphabets then check with different number 0 to 9999.

• Negative testing

→ testing the application with "Invalid inputs" is called Negative testing

→ it checks whether an application behaves as Expected with negative inputs

Ex: Enter only Number
abcdef Negative Testing

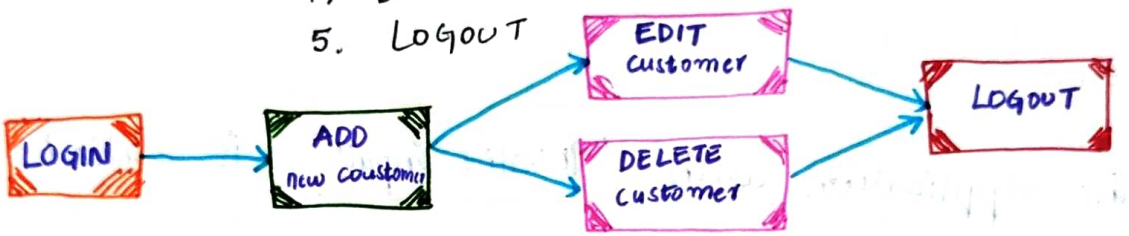
∴ Negative testing can be performed by entering characters "A to Z" (or) from a to z. Software should not accept the values (or) else it show throw an Error message. as invalid data inputs.

• End To End Testing

→ Testing the **Overall Functionalities** of the System including the data integration among all the modules is Called End To End Testing.

Ex:- Banking Test case

1. Login
2. ADD New Customer
3. Edit Customer
4. DELETE Customer
5. LOGOUT



• Globalization Testing (or) i18N testing → internationalization testing

→ Performed to ensure System (or) Software application can run in **any Culture (or) Local Environment**.

→ it should support Every language & different attributes.

→ it test **different** Currency formats, mobile number format and address format are Supported by application

Ex:- Facebook, Google, paypal

• Localization Testing.

→ Performed to check System (or) application for **Specific Geographical and Cultural Environment.**

→ it **only** supports specific kind of Language, Usuable only in specific region.

→ it tests the specific Currency Format, mobile numbe format and address format is working properly or not

Ex: Baidu.com
Supports only chinese language. Only access by Few Countries.

Test Design Technique (or) Test Data Design Techniques.

→ Used To **prepare data** for testing

*** 1. Reduce The Data ✓
**** 2. More Coverage. ✓

* Types of Test design Techniques

- Equivalence Class partitioning [ECP]
- Boundary Value analysis [BVA]
- Decision Table based Testing
- State Transition
- Error Guessing

The value will be Verified in the Textbox (or) Input Fields

↑

Input Domain Testing

* **Equivalence Class partition [ECP]** :- Value check by

→ **Partition / divide** Data into **various class** and we can Select data acording to a class Then **Test**.

→ it **reduce** the no. of test cases and save more time for testing

Ex: Enter a Number [ ] *Allows Digits from 1 to 500

"we check Only this Data by ECP Technique"

Normal Test Data

1
2
3
4
.
.
.
500

[Normal]

Divide Value into Equivalence classes

-100 to 0 → -50 (* Invalid)
1 to 100 → 30 (valid)
100 to 200 → 120 (valid)
200 to 300 → 280 (valid)
300 to 400 → 310 (Vaild)
400 to 500 → 480 (valid)
500 to 600 → 550 (*Invalid)

Test Data using ECP

-50
30
120
280
310
480
550

Ex:- Name [ ] * Allow any alphabets

Divide values into Ecp classes

A ... Z (valid)
a ... Z (valid)
Special characters (IN Valid)
Spaces (IN Valid)
Numbers (IN Valid)

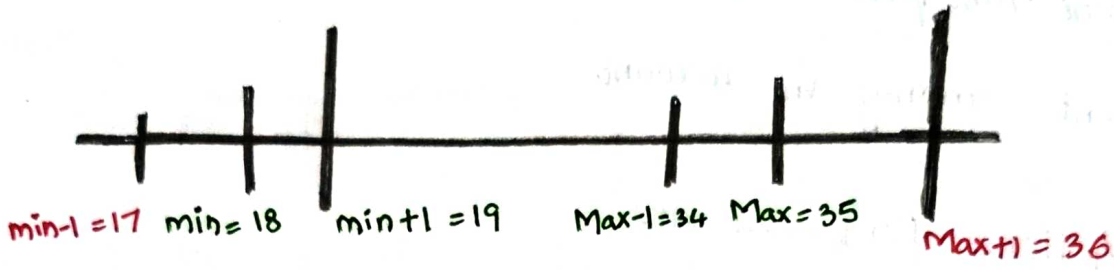
Test Data using ECP

XYZ
zyz
@# $ %
Xy z
1234

* Boundary Value Analysis [BVP]

→ BVA Technique Used to **Check Boundaries** of **input**

EX:- Enter a Age [ ] * Allow Digits From 18 to 35



∴ min = 18 Pass
min-1 = 17 Fail
min+1 = 19 pass

max = 35 Pass
max-1 = 34 Pass
Max+1 = 36 Fail

* Decision Table (or) (Cause-Effect Table) * Combinations

→ This Technique will be used if we have More **Conditions** and Corresponding **Actions**.

→ it deals with Combination of inputs.

→ To identify the test cases with decision Table, we consider **Conditions** and **actions**.

Example : Transferring Money Online To an account which is already added and approved

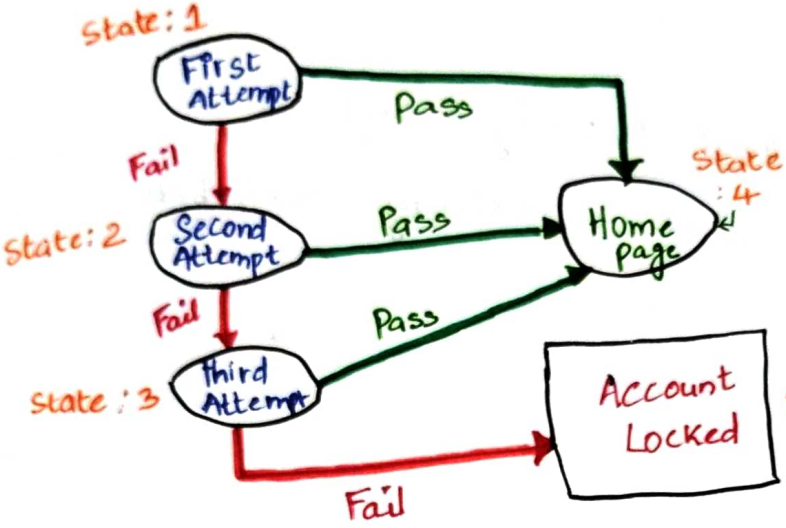
- Here, Conditions to transfer money are
  - Account already approved
  - OTP (one time password) matched
  - Sufficient money in account
- And, Actions performed are
  - Transfer money
  - Show a message as insufficient amount
  - Block the transaction, incase of Suspicious Transaction.

		Test case 1	Test case 2	Test case 3	Test case 4	Test case 5
Condition 1	account already approved	TRUE	TRUE	TRUE	TRUE	False
Condition 2	OTP matched	TRUE	TRUE	False	False	X
Condition 3	Sufficient money in account	TRUE	False	TRUE	False	X
action 1	Transfer money	Executed				
action 2	Show message "Insufficient amount"		Executed			
action 3	Block the transaction incase of Suspious Transaction			Executed	Executed	X

* State transition

→ In state transition Technique changes in input changes the State of Application

Ex:- Take A.T.M Password, Enter Wrong password 3 times and Account Automatically, blocks.



State	Login	Correct Password	Incorrect Password
S1	First Attempt	S4	S2
S2	Second Attempt	S4	S3
S3	Third Attempt	S4	S5
S4	Home page		
S5	"Display a message" Account locked		

→ This testing Technique allows the tester to test the behaviour of an AUT [Application under Test]

→ the tester can perform this Action by Entering Various Input Conditions in a Sequence

→ In State transition Technique, the testing Team provides "Positive" as well as "Negative" input test Values for Evaluating the system behaviour.

* Error Guessing

→ Error Guessing is one of testing techniques used to find bugs in software application based on the Tester Experience.

→ In Error Guessing we don't follow any specific rules

➡ Ex :-

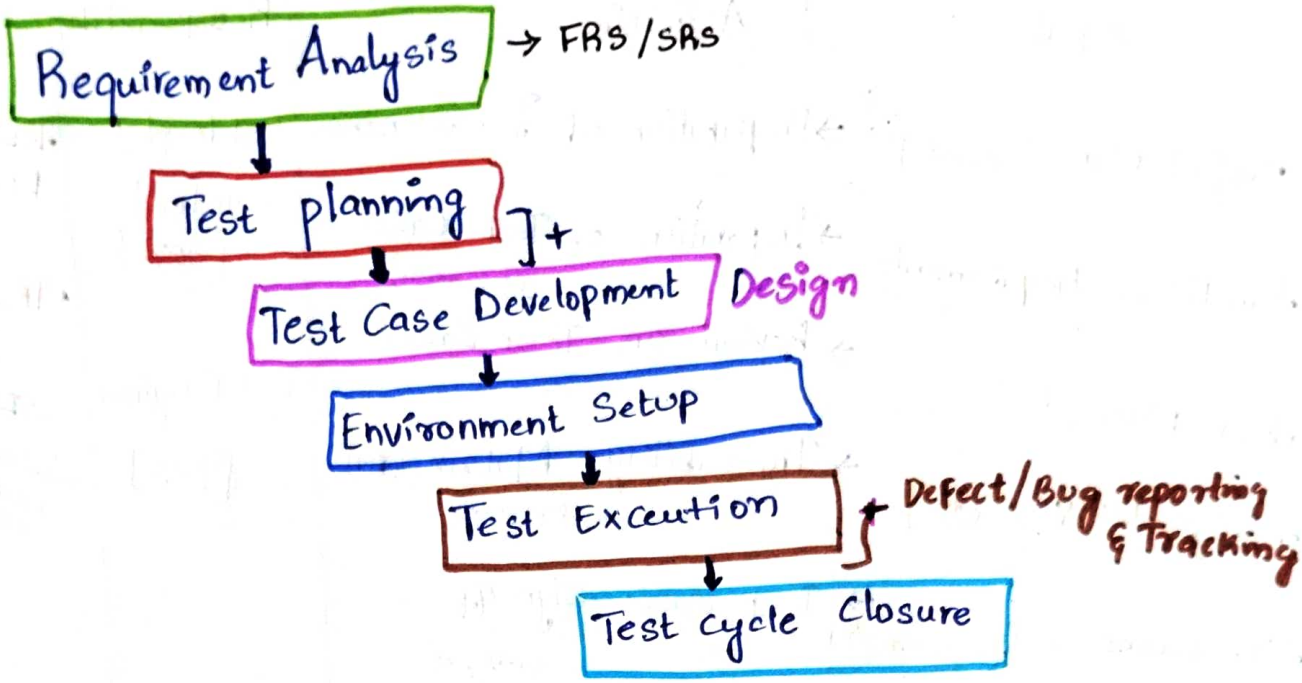
- Submitting a form without Entering Values
- Enter Invalid Values Such as entering alphabets in Numeric field.

23/12/22

- SDLC : Software Development Life cycle
  - → Requirement Analysis
  - → Desining
  - → coding
  - → testing
  - → deployment
  - → Maintance

- STLC : software Testing Life

STLC - Software Testing Life cycle



Test planing

PHASE	Input	Activities	Responsibility	Outcome
1 Test planning • what to test • How to test • when to test	• Project plan • Functional Requirements Documents	→ Identify Resources → Team Formation → Test Estimation → Preparation of Test plan [Document] → Review on Test plan → Test plan Sign-off	• Test Lead / Team Lead [70%] • Test Manager [30%] approved	Test plan Document

* Test Designing / Development [Writing Test Cases]

↳ step by step process / action To Perform To Test Functionality of Feature

input	Activities	Responsibility	Out Come
• Project plan [overall]	→ Preparation of Test Scenarios	Test Lead / Team Lead [30%]	• Test cases Document
• Functional Requirements	→ Preparation of Test Cases [Docs, Excel]		• Traceability Matrix [RTM]
• Test plan	→ Review on Test Cases ← peer	Test Engineer [70%]	Requirement Traceability Matrix
• Design Docs [UI]	→ Traceability Matrix [RTM] [mapping Doc]		
• Use Cases [requirement]	→ Test cases Sign-off (mail) / proof		

* Test Execution

input	Activities	Responsibility	Outcome
• Functional Requirements	→ Executing Test Cases P1 & P2 & P3 ...	Test Lead / Team Lead [10%]	• status / Test reports
• Test plan	→ Preparation of test Report Test log Document	Test Engineer [90%]	
• Test Cases	→ Identifying Defects		
• Build From Development Team → Software	Preparation of Defect Report		• Defect Report
• Test Reports / Test log [Defect reporting & Tracking]	Reporting Defects to Developers		

* Test closure / sign-off

Input	Activities	Responsibility	Outcome.
• Test Reports [meeting] • Defect Reports	→ Analyse Test Reports → Analyzing Bug reporting → Evaluating Exit Creteria P, & P of Ps .... Q: Zero Bug Creteria ? A: No: there will no 100% Free Bug application, minor Bugs will be there	Test lead / Test Manager [10%] Test Engneers [30%]	Test Summary Reports

Go Through Detailing.

* **Test plan Coments:**

A test plan is a document that describes the test scope test strategy, objectives, schedule, deliverables and resources required to perform testing for a Software product.

- Content should present in the test plan document.

- Test plan Template Contents :

→ OVerview → of Product/project

→ Scope

  - inclusions
  - Test Environment
  - Exclusions

  ] what to Test / what not test

→ Test strategy → what Kind of Test → Sanity / smoke / Functionality / Automation / Manual

→ Defect Reporting procedure →

→ Roles / Responsibilities → QA / Qc / Manager / Author / Reviewer

→ Test schedule → when / How / Date / Days

→ Test Deliverables → Each and Every phase Delivery Some Documents

→ pricing → Price / manager

→ Entry & Exit Criteria [start , End)

→ Suspension and Resumption Criteria [which case stop testing / start testing]

→ Tools [Manual : Excel / jira / word / automation]

→ Risks and Mitigations [Risks / alternative plan]

→ Approvals. [who will approve proj

Test Design

Use Case, Test Scenario & test Case

- **Use Case** : → requirment

→ Use Case Describes the **requirement** / FRS

→ it Contains Three items.

EX:

- Actor : Perform login [user]
  [single/group of People, interacting with a process]
- Action : login operation
  [which is to Reach the final outcome]
- Goal / outcome : Home page
  [which is successful user outcome]

- **Test Scenario** : [Manager / business owner] writes

what to test → A **possible area** to be tested **[what to test]** FRS Document

- **Test Case** :

→ **Step** by **step** actions to be performed to **Validate** the functionality of AUT [Application under test]

how to Test

→ Test Case Contains Test steps, **Expected results** and **Actual result**.

- **Use Case Vs Test Case**

* **Use Case :-** Describes **Functional requirement** // Prepared by ↓ Bussiness Analyst
* **Test Case :-** Describes **Test steps / procedure** / Test Engineer

- **Test Scenario Vs Test Case**

* **Test Scenario :** "**what to be tested**"
* **Test Case :** "**How to be Tested**"

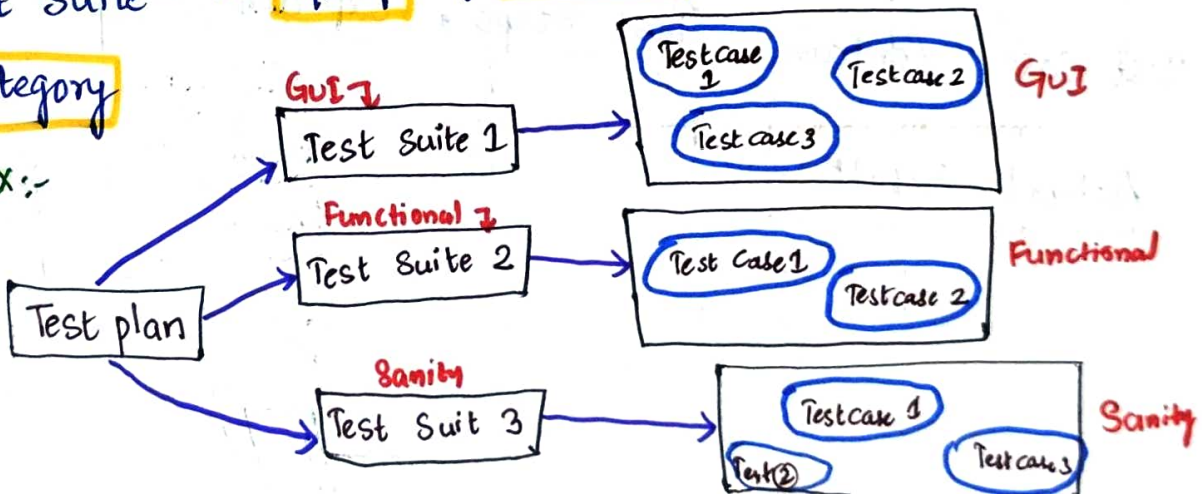
Example :

- **Test Scenario** : checking the functionality of Login button
- **Test Case** :
  - Test case 1 - Click the button **without Entering Username & password**
  - Test case 2 - Click the button **only Entering Username**
  - Test case 3 - Click the button while Entering **Wrong Username and Wrong password.**

- **Test Suite**

Test Suite is **Group** of **Test Cases** which **belongs to Same Category**

Ex :-



.Test Case Contents

Document Excel → Test Managment Tools Jira

→ Test case ID
→ Test case Title
→ Description
→ Pre-Condition
→ Priority [P_0, P_1, P_2, P_3] - Order (Smoke, Sanity, regression Test cases, Functional Test, UI related)
→ requirement ID
→ Steps / Actions
→ Expected Result
→ Actual Result
→ Test Data

- Actual Result

3. Application is maintaing the consistancy
4. Student discount is displayed instead of Sudent

- Expected Result

2. Browser should navigate to the home page.
3. Application should maintain the consistancy
4. Applications should display all the fields on a book flight page.

• Test Case Template:

Excel * Spicejet Book A Flight

1	Module_ name	Req. ID	Test type	Priority	Test case ID	Test Scenario	Pre Condition	Testcases / Test steps	Actual Result	Expected Result	Result
2	Homepage	1	+ve	P1	1	Verify The URL of Hompage	1. Open The browser 2. Enter The URL http://spicejet.com// 3. click on Go or press Enter	1. Open The browser 2. Enter the URL http:// spicejet.com 3. click on Go or press Enter			Pass
3	Book a Flight	2	GUI	P3	1	Verify The GUI of home page	Open The URL http:// spicejet.com	1. Check spell of all Fields 2. check alignment of all the Fields 3. Check the font of all the Fields 4. check the color of all Fields 5. Check the Feel & Look of Page.			Fail
4	Book a Flight	2	+ve	P1	2	Verify the Feilds in Book Flight Page	Open the URL http:// spicejet.com	Check The below Fields in home page Fieldname, Find flight, Round trip, checkbox, One way, Student, Leaving From, Field type, Going to, Radio, Date pickers, checkbox, children, Infants, By Mon			

• Requirement Traceability Matrix [RTM]

↳ "RTM" Describes the mapping of Requirement's with Test Cases

↳ The main purpose of RTM is to see that all test cases are Covered so that No Functionality should miss while doing Software testing

Sample RTM :

Requirment ID	Requirement Description	TestCase ID	Status
123	Login to The application	TC01, TC02, TC03	TC01 – Pass TC02 – pass
455	Ticket Creation	TC04, TC05, TC06 TC07, TC08	TC04 – Pass TC05 – Fail TC06 – Fail
600	Search ticket	TC09, TC10, TC11	TC011 – Pass TC09 – No Run

• Test Environment / Test Bed.

↳ Test Environment is platform specially build for Test case Execution on Software product.

↳ it is Created by integrating the required Software and Hardware / same as client along with proper network Configuration

↳ Test Environment Simulates production / real time Environment

∴ we have To replicate the Customer Environment to conduct Testing

Test Execution

→ During this phase test team will carry out the testing based on the test plans & test cases prepared

Entry Criteria : Test cases, Test Data & Test plan

- **Activities** :
  - Test cases are executed based on Test planning
  - status of Test cases are marked Like passed, Failed, Blocked Run
  - Documentation of test results and log defects for failed cases is done
  - All the blocked and Failed test cases are assigned bug ids
  - Restarting Once the defects are fixed.
  - defects are tracked till closure

- **Deliverables** :

  provide defect and test case Execution report with completed results

Guidelines

- Build being deployed to the QA environment is most important part of Test execution cycle
- Test Execution is done in Quality assurance [QA] Environment
- Test Execution happens in multiple cycles
- Test Execution phase consists Executing the test cases + test Scripts (if automation)

• Defects / Bugs

↳ Any mismatched functionality found in application is called Defect / Bug / issue.

↳ During Test Execution test Engineers are reporting mismatches are defects to developers through templates or using tools.

↳ Defect reporting tools:

→ clear Quest
→ Dev track
→ jira ✓
→ Bug jilla

• Defect Report Contents

* Defect_ID - Unique identification number for the defect
* Defect Description - Detailed description of the defect including information about the module in which defect was found.
* Version - Version of the application in which defect was found.
* Steps - Detailed steps along with Screenshots with which the developer can reproduce the defects
* Date Raised - Date when the defect is raised

* reference :- where you provide reference to the documents like, requirement design, architecture (or) may be even screenshots of the error to help understand the defect.

* Detected by - Name / ID of the tester who raised the defect

* Status - Status of defect, more on this later

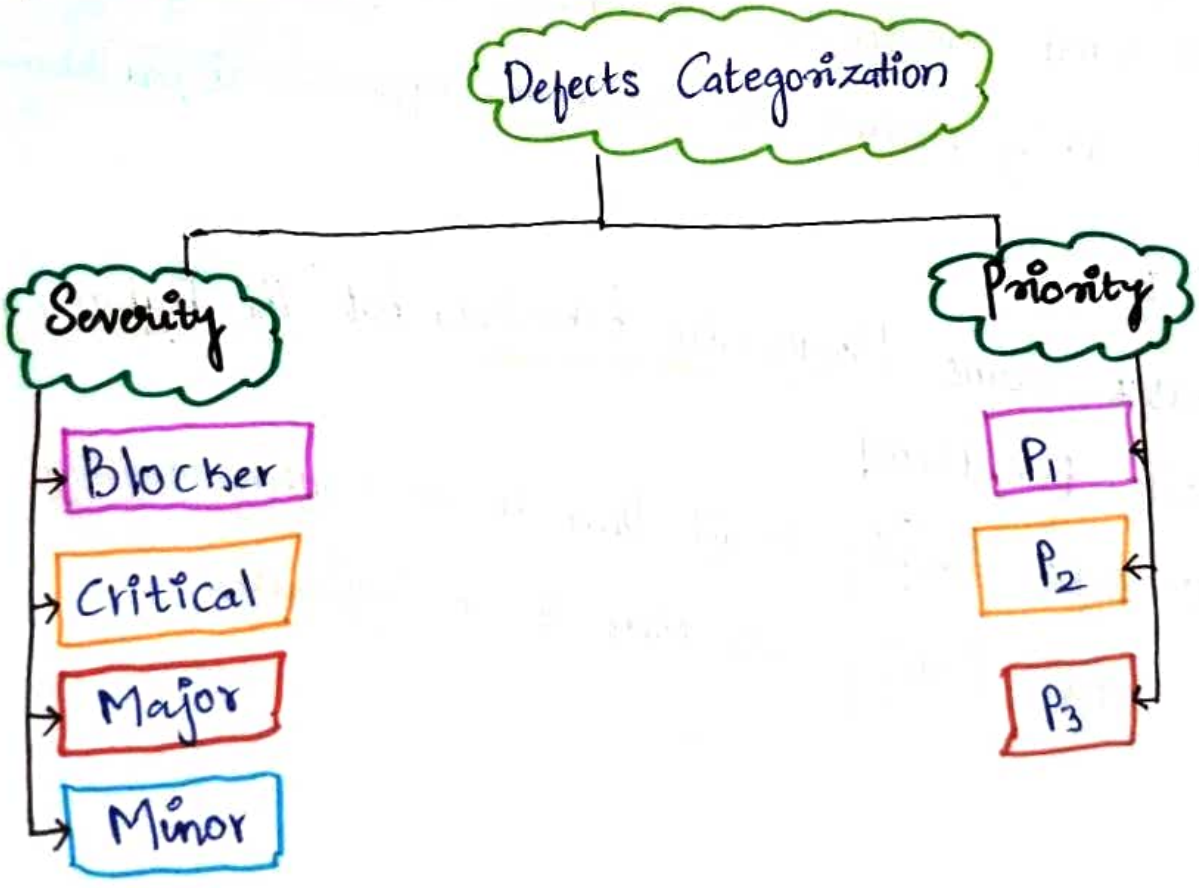
* Fixed by - Name / ID of the developer who fixed it

* Date closed - Date when the defect is closed

* Severity - which describes the impact of the defect on the application

* priority - which is related to defect fixing urgency. Severity priority could be high / Medium / low based on the impact urgency at which the defect should be fixed respectively.

• Defect Classification



→ checked by Tester

• Defect "Severity"

↳ "Severity" Describes Seriousness of defect and how much impact on Bussiness Workflow.

↳ it categorized into Four class

* Blocker [show stopper]: This defect indicates nothing can Proceed further.

Ex:- Application clashed, Login not Worked

* Critical:

The main / basic functionality is not working. Customer bussiness Workflow is broken. They cannot proceed furthur.

Ex: Fund transfer is not working in Net Banking

(ii) Ordering product in Ecommerce application is not working

* Major:

it Causes Some Undesirable behaviour, but the Feature / appli is still Functional

Ex:- After sending Email there is no Confirm message

(ii) after booking Cab there is no Confirmation

* Minor :- it won't cause any major break-down of the System

Ex:- Look & Feel issues, Spellings, alignments

↳ by Tester / manager / developer.

• Defect "Priority"

↳ Priority describes the importance of defect

↳ Defect Priority states the order in which a defect should be Fixed

↳ Defect priority can be categorized into three class

* P0 [High] :

The defect must be resolved immediately as it affects the System Sevely and cannot be used Until it is Fixed.

* P1 [Medium] :

it can wait Until a new versions / build is created

* P2 [low] :

Developer can Fix it in later releases

High Severity, Priority & Low Severity, Priority defects

Severity \ Priority	High	Low
High	login is Taking to Blank Page.	About Us link is going to blank page
Low	After User is logged into The application, he can see Home page. But, There is spelling mistake in Home page.	User Opened Contact page. Email ID has Spelling mistake

Ex:-

* Low priority - Low Severity

  A Spelling Mistake in a page not Frequently navigated by users.

* Low priority - high Severity

  Application crashing in some very Corner Case

* high priority - Low severity

  Slight change in logo Colour or Spelling mistake in Company name.

* High Priority - High Severity

  Issue with login Functionality (User is not able to login to the application)

* High severity - low priority

  web page no Found when User Clicks on link

* Low priority - Low Severity.

Any Cosmetic (or) Spelling issue which within paragraph (or) in the page.

*** **Defect Resolution** → by developer (opinion)

↳ after receiving the defect report from the testing team, development team Conduct a review meeting to fix defects. then they Send a resolution type to the testing team for further Communication.

- Resolution types *

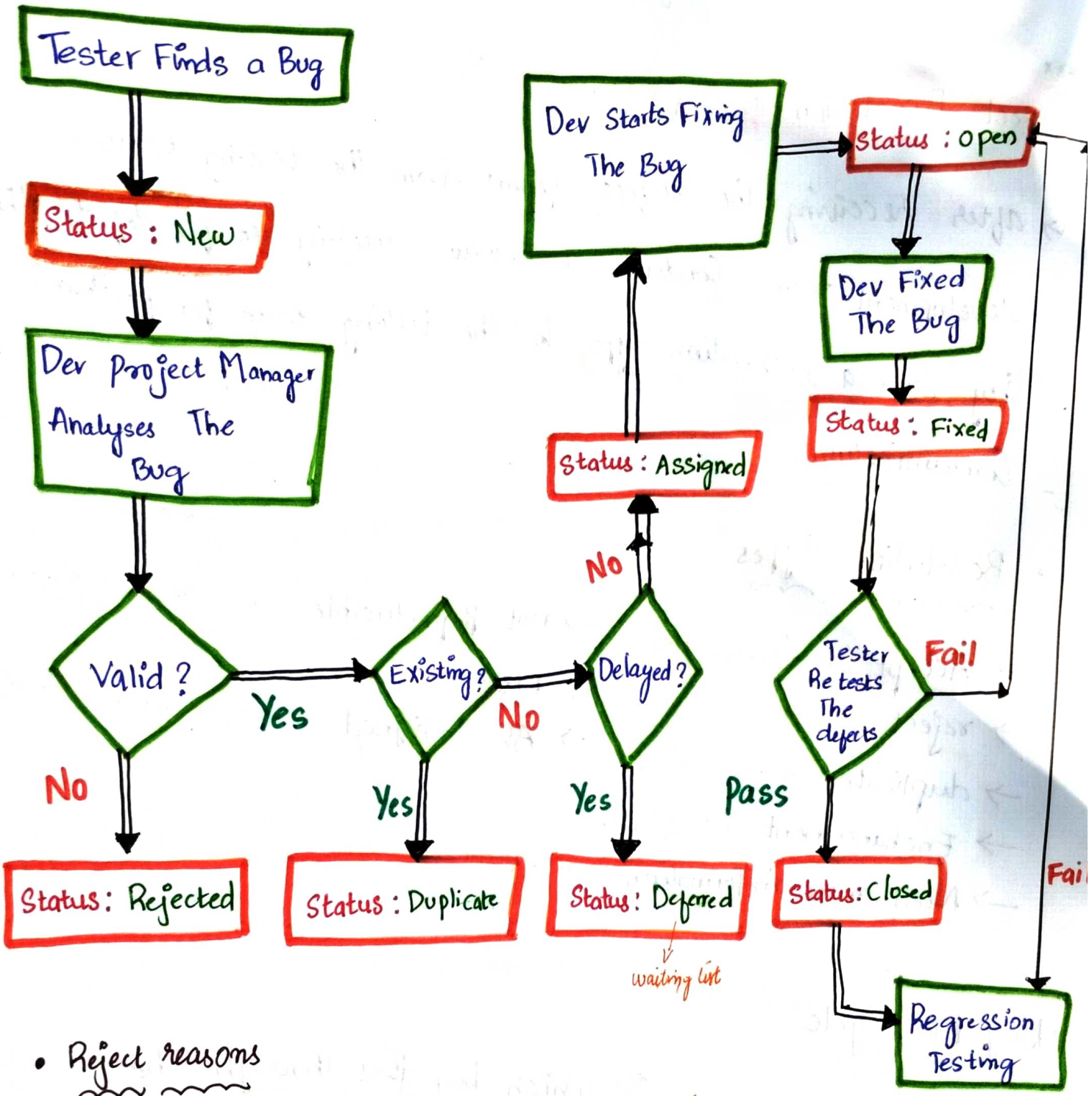
→ Accept [defect / ready to fix]
→ reject [not accepted]
→ duplicate [already raised]
→ Enchancement [new feature will come version]
→ Need more information [Screenshot, log file]
→ not Reproducible [try to fix, but bug not coming as dev] (Environment issue)
→ Fixed [✓]
→ As Designed [Not Bug, But Actual functionality]

• Bug Life Cycle

↳ it is a process in which bug goes through The different Stages in its Entire life

↳ defect Life Cycle is also known as "Bug life cycle".

Bug Life Cycle



- Reject reasons

↳ Enhancements : New Feature will come Next Version

↳ Need more information : Need logfiles, ses, screenshots

↳ Not Reproducible : Bug reports only in your Environment, Not on Developers

↳ As designed : Same Actual Functionality of the Application

• Text Cycle Closure

Activities

↳ Evaluate cycle Completion criteria based on time, Test Coverage, Cost Software, Critical bussiness Objectives, Quality.

↳ Prepare Test metrics based on above parameters

↳ Document the Learning out the project.

↳ prepare Test Summary report

↳ Qualitative & Quantitative reporting of quality due of the Work product to the Customer.

↳ Test Result analysis to Find out the defect distribution by type and Severity

Delivarables

- Test Closure report ✓
- Test Metrics ✓

* Test Metrics :

To test the metrics, we must have required data

To do, that, No. of requirements . . . .

S.no	required Data.
1.	Number of Requirements
2	Avg. no of Test Cases written Per requirement
3	Total. no of Test Cases written for all requirement
4	Total No. of Test Cases Executed
5	No. of test cases Passed
6	No. of test cases Failed
7.	No. of test cases Blocked
8.	No. of Test cases UnExcuted
9.	Total no. of defects Indentified
0.	Critical defects Count
1.	Higher Defects Count
,	Medium Defects Count
3.	Low defects Count
.	Customer defects
.	No. of defects found in UAT.
.	

These Data Used for Calculating "Test Metrics"

- % of Test Cases Executed :

$$\frac{\text{no. of Test Cases Executed}}{\text{Total no. of Test Cases Written}} * 100$$

- % of Test Cases NOT Executed :

$$\frac{\text{no. of Test Cases NOT Executed}}{\text{Total no. of Test Cases Written}} * 100$$

- % Test Cases passed

$$\frac{\text{no. of Test Cases passed}}{\text{Total test Cases Executed}} * 100$$

- % Test Cases Failed

$$\frac{\text{no. of Test Cases Failed}}{\text{Total test Cases Executed}} * 100$$

- % Test Cases BLOCKED

$$\frac{\text{no. of test Cases Blocked}}{\text{Total Test Cases Executed}} * 100$$

* Defect density : Number of defects identified per requirments

$$\frac{\text{no. of defects found}}{\text{Size (no. of requirment)}}$$

* Defect Removal Efficiency [DRE]

$$[A/A+B] * 100$$

$$\frac{\text{Fixed defects}}{(\text{fixed defects} + \text{Missed defects})} * 100$$

A = defects identified during Testing / Fixed defects

B = Defects identified by Customer / Missed defects

* defect Leakage

$$\frac{\text{no. of defects found in UAT}}{\text{no. of defects found in Testing}} * 100$$

** defect rejection ratio :

$$\frac{\text{no. of defect rejected}}{\text{Total no. of defect raised}} * 100$$

defect age : Fixed date − Reported date

Customer satisfaction : No. of Complaints per Period of time

• Q/A Testing Activities

↳ Understanding the requirements and Functional Specificati of the application

↳ identifying required Test Scenario's

↳ Designing Test Cases to Validate application

↳ Setting up Test Environment [Test Bed]

↳ Excecute Test Cases To Valid application

↳ log Test results [How many test Cases pass/Fail]

↳ defect reporting and tracking

↳ retest fixed defects of previous build

↳ Perform Various types of testing's in application

↳ reports to test lead about the Status of assigned Tasks

↳ Participated in regular Team meetings

↳ Creating automation scripts : selenium

↳ Provides recommendation on whether or not the application/ System is ready for production.

7 Principles of Software Testing

1. Start Software Testing at Early stages. Means from the beginning when you get the requirements.
2. Test The Software in Order to find the defects.
3. Highly immpossible to give the bug Free Software To Customer
4. Should not do Exhaustive testing. Means We should not use Same type of data for testing Every time.
5. Testing is Context based. Means decide what types of testing Should be Conducted based on type of application.
6. We should follow the Concept of Pesticide paradox. Means if you are executing Same Cases for longer run, they wont be find any defects. we have to keep update test cases in Every cycle / release in Order to find more defects
   add more test cases.
7. We should follow defect Clustering. Means Some of module Contains most of the defects. By experience, we can identify Such risk modules. 80 % of problems are found in 20 % of modules.

24/12/22

• Agile MODEL / Agile methodology / Agile process

↳ it is iterative and incremental Approach

↳ repeation ↳ Adding new Features

Agile Principles

1. Customer No need To wait for long time
2. We develop, Test and release piece of Software to the Customer with Few number of Features.
3. We can accept / accodmodate requirement changes.
4. There will be good communication between Customer, BA, Developers & testers.

Advantages of Agile

1. Requirement changes are allowed in any stage of development (or) We can accomodate requirement changes in the middle of development.
2. Releases will be Easy Very Fast [weekly]
3. Customer no need To wait for long time.
4. Good Communication between Team.
5. it is Very Easy model To adopt

Dis Advantages

1. less Focus on design and documentation Since we deliver Software Very Faster
2. Less documentation

SCRUM

↳ Scrum is a Frame Work

↳ which we Build Software product by following "Agile" Principles

→ Scrum includes Group of People called "Scrum Team".

Normally Contains 5-9 members.

↳

1. Product owner
2. Scrum master
3. Dev Team
4. Q/A Team

} Delivery Quality product To The Customer As Soon as possible.

* Product Owner :-

↳ Define the Feature of the product From Customers.

↳ prioritize Features according to market value

↳ P_0, P_1, P_2

↳ Adjust Features and priority every iteration, as needed

↳ Accept or reject work results.

* Scrum master :

↳ The main role is Facilitating & driving the agile process.

He will Take over Total

* Developers and QA :

Develop & Test Software

Scrum Terminology

* User Story : A Feature/module in a Software
* Epic : collection of user stories
* Product Backlog : Contain List of user stories. Prepared by Owner.
* Sprint : Period/Span of time To Complete the User Stories, decided by Product Owner & Team, Usually 2-4 weeks of time.
* Sprint planning meeting : Meeting Conducts with Team to define what can be delivered in Sprint and duration.

* **Sprint backlog**: list of Commited Stories by Dev/QA For Specific Sprint

* **Scrum meeting**: Meeting Conducted by Scrum master Everyday 15 min called "Standup meting"/"scrum call"

Q — What did you yesterday ?
What will you do today ?
Are there any impendiments in your way ?

* **Sprint "retro spective" meeting** :

Conducts meeting after Completion of Sprint. The entire Team, including both the "Scrum Master" and "Product Owner" should participate.

* **Story point**: Rough Estimation of user stories, will be given by Dev & QA in form of Fibonacci Series

0 1 1 2 3

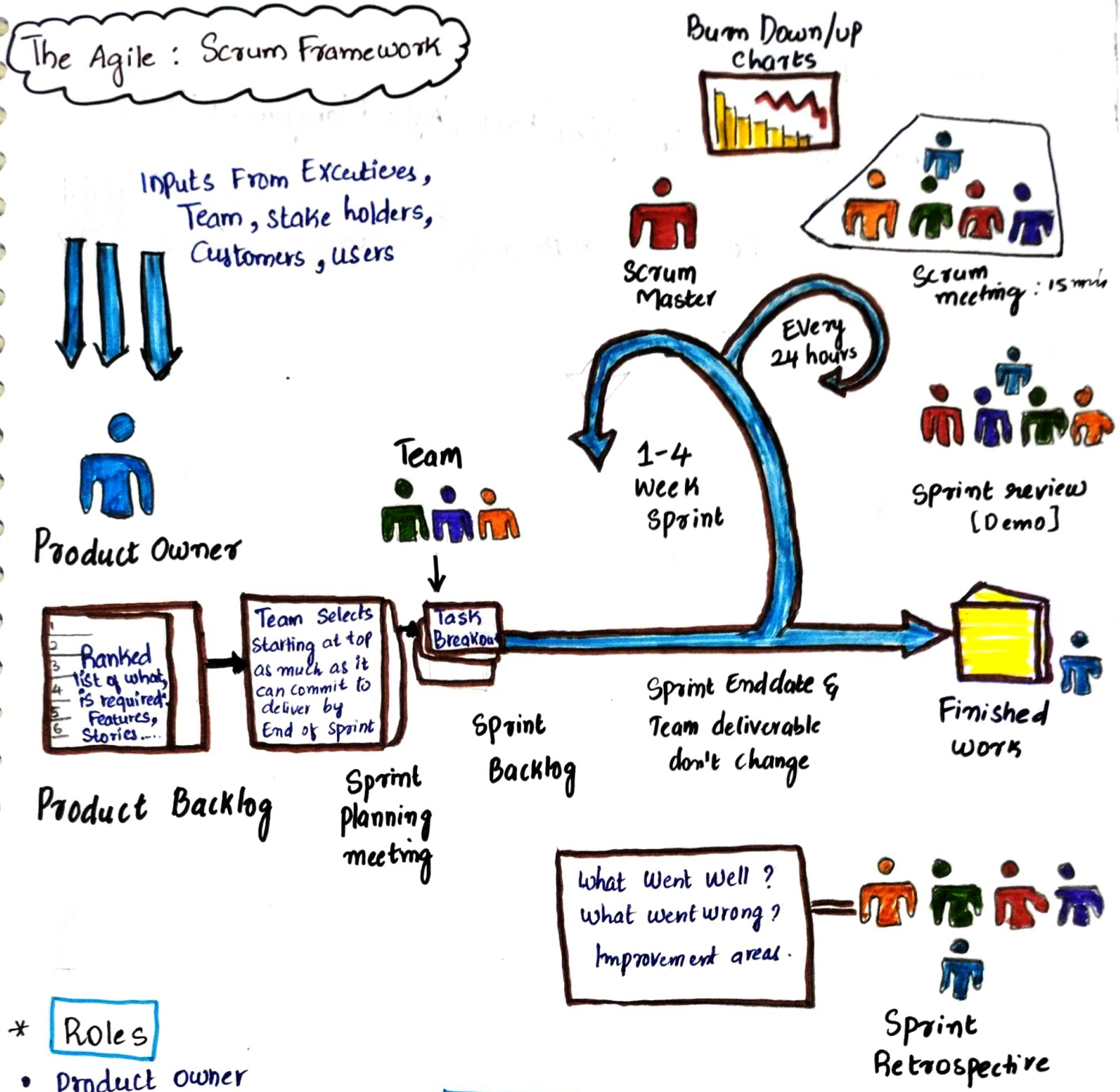
1 Story point - 1 hour / 1 day (6 hours)

Ex: login → Dev 5hrs / QA 3hr = 8hrs / 1 Day

* **Burndown chart**: shows how much work remaining in the Sprint.
Maintained by Scrum master daily.

1 Story --- 3 day (18 hrs)

The Agile : Scrum Framework



* **Roles**
  - Product owner
  - Scrum Master
  - Team

* **Cermonies**
  - Sprint planning
  - Daily Scrum
  - Sprint review

* **Arefacts**
  - Product Backlog : all requirement/stories
  - Sprint Backlog Burdown Chart → completed work : Specific no. of stories / Commited stories by dev/QA :. Ready For dev / Testing

Scrum Board

This is done in "Jira" tool [Agile Management Tool]

Stories	To Do	In Progess	Testing	Done.