**Types of Testing:**

1. GUI testing,
2. Functional testing,
3. Regression testing,
4. Smoke testing,
5. load testing,
6. stress testing,
7. security testing,
8. stress testing,
9. ad-hoc testing etc.,

**Manual: unit testing**, **integration testing**, **system testing, and user acceptance testing**.

user story ,future task

product backlog items (high level description about change we want to do) we will list all the items we want to change and what changes we are going to do (EX compose functionality) After the sprint duration they will move the product backlog items to sprint bcklg items

sprint backlog items

sprint planning meeting (before we start sprint)

They will prioritize what is imp and what is current thing to be done (product owner will do it by taking input from us)

Scrum master manages 3 to 5 scrum teams

vl b part of one scrum meeting

(USER STORY)When we put prodBcklog to sprint Bcklg items we define it as user story

It will decide what needs to be done

Task-Task is details related to the User Story

------------------------------------------------------------------------------------------------------

Gmail Functionality

need to check with possitive test case & negative test case

Send Functionality

Valid email address

Dont enter anything

wrong email

Broken email

-----------------------------------------------------------------------------------------------------------------------------

While writing we have to think in different ways

Whatever change we do we have to check whether its impacting other places or not

How do we know whether page is using webservice or not?

We need to know from the developers.

------------------------------------------------------------------------------------------------------------------------------

IMP

Functional --- Possitive test & Negative testing

Integration ---First we need to find out whether it is interacting with particular database or web services Form is using data base

or API 's or web services

Suppose interacting with database when we click the one button we are going to the page or not

No need to write possitive test cases just check whether we are going to next step

Smoke ---- We write test cases only with possitive data (Ex Inbox page is coming or not) We write smoke cases in the starting

Before we Start we need to make sure its main functionality is working or not...

We need to worry only about possitive test cases

Bank Of America Transfers

Credit card Payment

Regression -----we automate whole functionality..

we take possitive and write code for possitive test cases

-------------------------------------------------------------------------------------------------------------------------------

When we get a build for 2 weeks

We need to first

Integration

Smoke

Functional Testing

Regression (Automation)

we can write automation code for functional,Smoke & Integration

-------------------------------------------------------------------------------------------------------------------------------

**Automation:**

**Selenium can be used for web based application testing and supports**

**Functional and**

**Regression Testing**

**Software Testing:**

Process of testing the application

1. under different conditions
2. over a period of time
3. Evaluating the result (Expected Result Vs Actual Result)

**Product:** We first develop the software and then we advertise and sell it. No client.

Ex: Antivirus, MS Office, (Product based orgaisation)

**Project:** We develop based on the client requirements and then deliver to the end customers.

(Project or service based organization)

**MANUAL AND AUTOMATION DIFF:**

Manual done without using any tool done by tester itself and the reports are created manually. Automation is done using tools like Selenium, QTP, Load runner by creating automation scripts. Same scripts can be used to test the application again and again.

**TYPES OF APPLICATION:**

Window based application: Install on one system and can be used on the same system. Ex: Calculator, games

Web based application: Applications deployed anywhere on the server and we access it by using the URL, Browser and Internet. EX: Google, Gmail, Yahoo, Etc.

Clientt-Server application: These are also deployed on the server but cannot be accessed by using URL. We need to install client software on the system. EX: Skype, gTal**k**

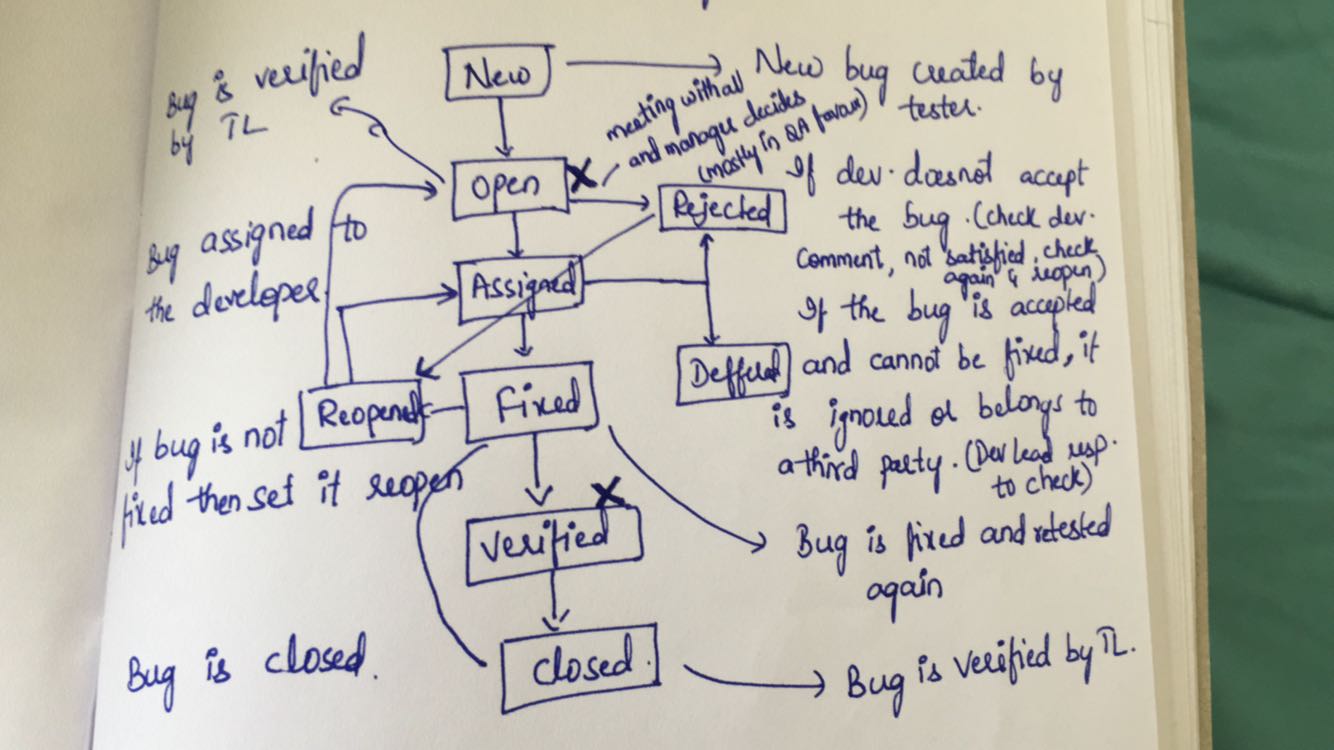
**BUILD:** Software we get internally in a company for testing from the developers is called build. Every time it comes as a different build with a unique number. The bugs in one build can be resolved in the next build by the developers. Every build can have new functionality and/or bug fixes.

**Version:** At a point there will be a build from dev team and everything works fine we release the build to the client and it is called as version with different version numbers.

**BUG:** whenwe get requirements from the client from the requirement we get the expected results. After testing application we get actual result. If both the results are different, we call it a bug (defect, issue).

All the client requirements are mentioned in a document called SRS.(software requirement specification)

**Bug Life cycle:**

****

**Test Case: Di**fferent conditions which we are going to test on an application in a formatted way is called Test case.

1. Requirements
2. Test cases
3. Build
4. Running test cases

Login Functionality:

1. Login with correct username and correct password
2. Login with Correct username and incorrect password
3. Login with Incorrect username and correct password
4. Login with Incorrect username and incorrect password
5. Login with username in caps and password
6. Login with username and password in caps

Test Case Format:

* Test Case ID
* Description
* Pre-Condition
* Steps
* Data
* Expected result
* Actual result
* Status
* Comments

Fields good to be used:

* Priority
* Type
* Bug ID

Test case ID: Name that makes to anyone who refers it in future

Description: Need to provide as much information as possible about what we are going to test. No need to provide expected result.

Pre-Condition:

* Any user or page dependency
* Test environment dependency
* Any special setup done before running the test case
* Dependency on any other test case
* Ignore all generic conditions which are applicable to all test cases.

Steps: Always best to define every step and the step should be in serial manner.

Test data: It is very important for each and every test case. Change in it can change the expected behavior of the application.

We can give exact data or range.

Expected result:

* We get it from client requirements.
* Need for each and every test case
* Clear and understandable
* If needed define any special requirements

Actual Result:

* Applicable only after executing test case
* Not entered while writing the test case
* Parallel from the expected result

Status: applicable but only after executing test case

* Pass
* Fail
* No run
* Blocked

Comment: We enter only after executing the test case. If you want to specify any information after running the test case.

Complete life cycle in case of a valid bug: Accepted

* Reported defect by QA.
* Developer lead will assign this defect to any developer
* Developer will login and verify, it’s a valid defect or not, developer will work on it.
* Developer lead will give new build to tester and set this defect is ready for testing.
* Now tester will login and verify.

Complete life cycle in case of a valid bug: Rejected

* Login as dev lead
* Developer will login and verify if the defect is valid
* Developer will reject the defect
* Satisfied with developer comments, Close the defect
* If not, reopen the defect, assign to developer

**SQL For testers:**

**Select Basics:**

Select \* from emp; (Select all the data from table emp)

Select ename, Job, Sal from emp; (not case-sensitive)

Select \* from emp where sal >/</= 2000;

Select \* from emp where job = ‘CLERK’;

Select \* from emp where sal = 2000 OR Job = ‘CLERK’;

Select \* from emp where sal = 2000 AND Job = ‘CLERK’;

Select \* from emp where sal in (1500, 3000, 5000);

Select \* from emp where sal not in (1500, 3000, 5000);

Select \* from emp where sal (not) between 3000 and 5000;

Select \* from emp order by sal; (ascending)

Select \* from emp order by sal desc;

Select \* from emp order by deptno, sal; (first orders by deptno and in that sal again)

Select \* from emp where ename like ‘A%’; (% means any number of chars after A)

Select \* from emp where ename like ‘%A%’; (A anywhere in the ename)

Select \* from emp where ename like ‘\_A%’; (\_ means only one character before A)

Select \* from emp where ename like ‘\_\_\_\_\_’; (5 cahracter names)

**Distinct keyword:** It is used to fetch unique values from the table

Select job from emp;

Select distinct (job) from emp;

Select distinct \* from emp;

**Group functions:** Group functions are those which apply on group of multiple rows/columns but give only one result.

Min, Max, Avg, Sum, Count

Select \* from emp;

Select max (sal) from emp;

Select min (sal) from emp;

Select avg (sal) from emp;

Select sum (sal) from emp;

Select count (sal) from emp;

**Group By:** It is used to display data in the form of groups.

Select dept no, Count (\*) from emp group by (dept no);

Select dept no, Count (\*) from emp group by (dept no) having count (\*) > 4;

Select job, Count (\*) from emp group by (job) having count (\*) > 1;

Select job, Count (\*) from emp where sal > 1000 group by job having count (\*) > 2;

Where is before grouping (table data) and having is after grouping (group data)

**Sub Query:** Query inside a query

Select \* from emp where sal > (select avg (sal) from emp);

**DELETE:** Conditionally delete data from the table but table still exists.

Delete from employee where location = ‘noida’;

Select \* from employee;

Delete from employee; (complete table deleted)

**TRUNCATE:** Delete all the data but table still exists

Select \* from testEmployee;

Truncate table testEmployee;

**DROP:** Delete the data from the table and also the table structure.

Drop table Myemployee;

**DIFFERENCES BETWEEN DELETE/DROP/TRUNCATE:**

* Delete is a DML(Data Manipulation Language) command, Truncate and Drop are DDL (Data Definition Language) commands.
* Delate removes data conditionally, Truncate and drop remove data completely
* Delete can be rolled back, whereas truncate cannot be rolled back.
* Delete and Truncate remove data only, Drop will remove the table structure as well.

**JOINS:**

Joins are used to fetch data from two or more tables.

1. Equi/Inner/Simple Join
2. Outer Join
3. Cartesian/cross join
4. Self join

**Cartesian Join:** Each record from one table is concatenated with the each record from another table.

Final number of records = Number of records in table 1 \* number of records in table 2

Select \* from emp;

Select \* from dept;

Select \* from emp, dept; (each row from first table is concatenated with each row in second table)

**Inner Join:** We can fetch data from one or more tables by equating values of 1 or more columns

Select empno, ename, loc, emp.deptno from emp, dept where emp.deptno = dept.deptno;

When we have both the values in the tables we have to mention from which table to pick it up.

Select empno, ename, loc, A.deptno from emp A, dept B where A.deptno = B.deptno; (alias)

**Outer Join:** It is used to fetch data by equating some condition but is alos fetch data which doesnot have corresponding records in other table.

Left Outer join

Select empno, ename, loc, from emp A, dept B A.deptno = B.deptno(+);

Right Outer join

Select empno, ename, loc, from emp A, dept B A.deptno(+) = B.deptno;

**Self Join:** Join a table to itself. Make 2 copies of the table and making a join between them.

Select A.ename, B.ename from emp A,emp B where A.mgr=B.empno;

**CREATE TABLES:**

Tables are used to hold the data in the form of rows and columns.

Create table TWTable (id number(5), name varchar2 (20), salary number(10));

Desc TWTable; (shows the structure of the table)

Create table Emp\_new as select \* from emp; (Create table by copying from the old table)

**INSERT DATA:**

Select \* from TWEMPLOYEE;

Insert into TWEMPLOYEE values(1, ‘Prassu’, 20,000);

**UPDATE DATA:**

Update TWEMPLOYEE set salary = 12000 where name = ‘John’;

Update TWEMPLOYEE set salary = ‘12000’ where salary>9000;

**VIEWS:** Similar to table where we can use select queries, they donot hold their own data but picks from the table when query is executed and views secure our data.

Select \* from emp;

Create view EMP\_VU as select empno, ename, mgr, salary from emp; (user can see the part of the data in the view)

**INDEXES:**  Used to improve performance and created when we make a lot of select query on our table which are similar to the indexes in our book.

Create index EMP\_I on emp(job);

Select \* from emp where job = ‘manager’;

**Constraints:**

**Not Null:**  Null is not zero or space. Null means its not available or not assigned.

Create table TW\_cons1(mane varchar(20), id number(3) not null, salary number(5));

Constraint can be defined as a rule which we can set on a particular column.

Insert into TW\_cons1 values(‘John’, 1, 3000);

Insert into TW\_cons1 values(‘Kevin’, null, 4000); (error because of not null constraint)

**Unique key:** holds unique value but can hold null values.

Create TW\_UN (id number(3) unique, name varchar2(20), salary number(5));

Insert into TW\_UN values(1, ‘John’, 3000);

Insert into TW\_UN values(null, ‘Kevin’, 4000);

Insert into TW\_UN values(1, ‘John’, 3000); (throws error due to unique constraint)

**Primary key:** holds unique value but cannot hold null values

Create TW\_P (id number(3) primary, name varchar2(20), salary number(5));

Insert into TW\_P values(1, ‘John’, 3000);

Insert into TW\_P values(2, ‘Kevin’, 4000);

Insert into TW\_P values(1, ‘John’, 3000); (throws error due to primary constraint)

Insert into TW\_P values(null, ‘Kevin’, 4000); (throws error due to primary constraint)

**UNIX Versions:**

1. Solaris
2. HPUX
3. AIX
4. Ubuntu

**Linux Versions:**

1. RedHat
2. Fedora

**Why UNIX/LINUX?**

In most of the companies application server is deployed on Linux/Unix machine. So testers need to access it many times.

**Major tasks of testers:**

* Install/Uninstall software/application
* View log files
* Take backup of log and other files
* Cleanup spaces before any installation
* Verify memory before any installation
* Verify and kill processes before any installation etc

**How to access: (software: Putty) enter the IP of the machine and then login through username and password**

**UNIX basic commands:**

* **Pwd –** present working directory (shows current location)
* **Ls-** displays all the folders in that location
* **Ls-**l : display listing format with complete details
* **Ls-**la: it also displays hidden files
* **Clear-** to clear the screen
* **Mkdir-** to create a folder
* **Rmdir-** to remove a folder
* **Man (command)-** to know manually about the command

**Checking log files:**

* **Cat filename –** It will open the complete file (cat log.txt)
* **Head –**  It will open only a part of the file from the top (head –n 5 log.txt)
* **Tail –** It will display only a part of the file from the bottom (tail –n 5 log.txt)
* **More -** It will display page by page and we can only move in forward direction(more log.txt)
* **Less-** It will display everything and unlike more we can move in forward and backward directions (less log.txt)

**Processes:**

* **Ps-** It shows the processes only started by me as a user
* **Ps – ef –** It shows all the processes running on my system.
* **Top-**  It displays top 20 processes which are occupying highest memory/CPU in the system. Use Ctrl+Z to come out of the top command.

**Memory status of the machine:**

* **Df-** Total used and free memory of the drives we have (df –h) h is human readable from
* **Du-**  we can find the space taken by files & folders in my current location (du –h)
* **Free-** we can find free and used memory and also buffer and cache status

**Find and Kill Process:**

* **Ps-**
* **Kill-**  If you want to kill any process you can do it by using kill (kill -9 PID) -9 is killing forcefully and PID is process ID.
* **Ps –ef | grep java-** If you want to check if a particular process is running on a system, here java is the process.

**File Comparision:**

* **Cmp –** It will show the first byte which is different in the two files.(cmp fileA fileB)
* **Comm -** Common content which exists in both the files (comm fileA fileB)
* **Diff –** To check the difference between two files (diff fileA fileB)
* **Wc -** WC is the word count (wc log.txt)
* **Sort –** we can sort the content of any file (sort log.txt)

**Search Content:**

* **Grep-** we can search the content in a file (grep Error log.txt)
* **Find-** we can check if a particular file exists in the current folder (find log.txt)
* **Locate-** we can make a search in the complete system (locate log)

**Backup of files:**