MANUAL TESTING

1. Why did you choose testing?

> ➢ Scope of getting job is very high

> ➢ No need to depend upon any technology

> ➢ Testing will be there forever one can be constituent through life

2. Who can do testing?

Any graduate, who is creative, can do testing

3. What is project? (Service based)
Project is something and that is developed based on the particular customer requirements and

used for their usage only.

4. What is product?

Product is something that is develop based on the company specification and used by many Customers.

Note: The Company will decide the specification by gathering so many common customers' requirements in the market.

5. Why now a day's only the test engineers are been explicitly recruit into the software company?

> ➢ One person can't efficiently perform two task at a time.
> ➢ Sentimental attachment. (finding the defects others)

6. What is Quality?

QA (Quality Analyst), Software Test Engineer, Software tester

Quality is defined as justification of all the requirements of a customer in a product.

Proof

Note: quality is not defined in a product it is defined in the customer mind.

Quality is defined as not only justification of all the requirements and also the presence

of value (user friendliness)

7. What do you mean by testing?

Testing is a process in which the defects are identified, isolated(separated) subject for rectification and ensure that the product is defect free in order to produce Quality product in the end and hence customer satisfaction.

Bidding the project: It is a defined as request/proposal, estimation and signing off.
Kick of meeting: It is an initial meeting conducted in the software company soon after the project is signed off. In order to discuss the over view of the project and also to select a project manager for that project.

Usually project managers, technical managers, quality managers, high level management and if required development lead, test lead will be involved in this meeting. In some cases the customer representative may be involved in this meeting.

Note: a part from this meeting any other start up meeting is also called as kick off meeting.

 8. What is PIN? (Project Initiation Note)

It is a mail prepared by the project manager and sent to the chief executive officer of the software company as well as to all of the core team members in order to intimate them that they are about to start the actual project activity.

Software Development Life Cycle: It contains 6 phases

1. Initial phase (or) requirement phase (or) Planning phase

2. Analysis phase

3. Design phase

4. Coding phase

5. Testing phase

6. Delivery & maintenance phase

1. **Initial phase:**

Task: interacting with the customer for gathering the requirements.

Roles: i. Business Analyst (BA) - Non-Technical (Functional)

ii. Engagement manager (EM)

 Process: first of all, the business analyst will take **an appointment** from the customer, he will collect the **template** from the company, and he will meet the customer on the appointment date, **gather the requirement with the support of the template**, and come back to the company with requirement documents.

The engagement manager will go through the **requirement document** and will try to find **extra requirements and confused requirements**. It at all any extra requirements are found he deals with **excess cost** of project and if at all any **confused requirements** are found then he will ask the corresponding team to build a **prototype**, he will take that prototype, go to the customer and demonstrate it to the customer and gather clear requirement finally he will hand over the requirement document to the business analyst.

Proof: the proof document off the initial phase is requirements document.

This document is also called with following names in different documents.

FRS (functional requirement specification)

SRS (system requirement specification)

CRS (client/ customer requirement specification)

URS (User requirement specification)

BRS (business requirement specification)

BDD (business design document)

Some companies will maintain two documents in the initial phase

➢ They will keep high level business information

➢ They will keep the detailed functional requirement information

But some companies will maintain both the information in a single document.

Template: It is a predefined format which is used for preparing a document in a easy and perfect manner.

Prototype: It is roughly and rapidly model which is used for demonstrating to the client in order to gather the clear requirements and also to win the confidence of a customer.

2. Analysis phase:

Tasks: i. Feasibility (possibility) study

ii. Tentative (temporary) planning

iii. Technology selection and environment confirmation

iv. Requirement analysis

Roles:

i.   system analysis
ii.   Project manager
iii.   Technical manager (or) Team manager

a) **Feasibility study:** It is a detailed study of requirements in order to conform whether all of those requirements are possible within the **given budget time and available resources** or not.

b) **Tentative planning:** resource planning as well as scheduling will be temporary planned here in this section.

c) **Technology selection and environment conformation:** the list of all the technologies that are required to accomplish the project successfully will be analyzed and the suitable environment for that project also will be analyzed and mention this section.

d) **Requirement analysis:** the list of all requirements (that all required by a software company) to accomplish that project successfully will be analyzed mention hear in this section.

Ex: usually s/w Company requires h/w, s/w and resources

Proof: the proof document of the analysis is system requirement specification. (SRS)

Note: in requirement space the output document any also is called as SRS but it is software requirement specification.

3. **Design phase:** design phase is used to SRS

Tasks: i) High level designing   ii) Low level designing

Roles: **chief architecture will do High level designing** and **technical lead will do low level designing.**

Process: the chief architect will divide into whole project into modules by drawing some diagrams using a language is **unified modeling language**.

The technical lead will divide the modules into sub module by drawing some diagrams using the same UML. Apart from this GUI part design and Sudoku development will also be done in this phase.

Proof: the proof document is TDD (Technical Design document)

Note: pseudo code is a set of English instructions which will make the developer more comfortable while developing the source code.

4. Coding phase:

Tasks: programming & developing

Roles: programmers & developers

Process: the developers will develop the **actual source code** with the support of **technical design document** and also by following the coding standards like proper identification, (left margin) color coding proper connecting and etc.,.

Proof: the proof document of the coding phase is **source code document.**

5. Testing phase:

Task: testing

Roles: Test engineers, QA

Process:

- ➢ the Test engineer will receive the requirement document (FRS,BRS,SRS)

- ➢ they will start understanding the requirement

- ➢ While understanding the requirements if at all they get any doubts then they will list out all those doubts in the **requirement clarification note (RCN)** book. They will send it to the author (**BA or EM**) of the requirement document note and will wait for the clarification.

- ➢ Once the clarifications are given and after understanding all the requirements clearly they will take the test case template and will write the test cases.

- ➢ Once the first build is released, they will execute all the test cases.

- ➢ During execution if at all any defects are found then they will list out all those defects in the defect profile document.

- ➢ They will send the defect profile document to the development department.

- ➢ Once the **next build** is released, they will execute the required test cases and try to find the defects

- ➢ If at all any defects are identified they will update the defect profile document and will send to the development department.

- ➢ Once the next build is released, they will repeat the same process again and again till they feel the product is defect free.

 Proof: the proof of testing phase is quality product **test case document.**

6. **Delivery and maintenance:**

 Task: handover the application to the client

 Roles: Deployment engineers or installation engineers

 Process: the deployment engineers will go to the customer place, installs the application into the customers environment and handover the original software to the customer.

 Proof: the final official agreement made with win the company and stack holder is the proof document for delivery.

Maintenance: once the application is delivery to the customer they will start using, while using if at all they face any problems then that problems will become tasks, based on those tasks the corresponding roles will be appointed, they will define the process and will solve the problem.

 Some customers may ask for continues maintenance. In that case the company will send a team of members to the customers place and ask them to take care of software continue sly.

Q. where exactly testing comes into picture? (Or) which sort of testing you are expecting? (Or) how many sorts testing are there?

(Ans) There are 2 sort testing are there

    i.   un-conversional testing
    ii.   Conversional testing

QA / Software Tester

Un conversional testing: It is a sort of testing in which the **quality assurance people** will check each role in the organization in order to conform whether they are doing their work according to the **company process guide lines** or not. Right from the **initial phase** of software development life cycle is end.

Conversional testing: It is a sort of testing in which one **test engineers** will check the developed application or its related parts are working according to the expectation or not from the **coding phase** of the SDLC to the end.

Q. Testing methods (or) testing techniques?

Basically, there are 3 types of testing

    i.   Black box testing
    ii.   White box testing (Glass, clear)
    iii.   Gray box testing

Black box testing: if one performs testing only on the **functional part (GUI- Graphical user interface)** of an application without having of the knowledge of structured part (**coding**) then the method of testing is known as black box testing.

Usually, the black box testing test engineers are performing.

BLACK BOX TESTING, also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is **not** known to the tester. These tests can be functional or non-functional, though usually functional.

- o **black box testing:** Testing, either functional or non-functional, without reference to the internal structure of the component or system.

- o **black box test design technique:** Procedure to derive and/or select test cases based on an analysis of the specification, either functional or non-functional, of a component or system without reference to its internal structure.

Example

A tester, without knowledge of the internal structures of a website, tests the web pages by using a browser; providing inputs (clicks, keystrokes) and verifying the outputs against the expected outcome.

Levels

Black Box Testing method is applicable to the following levels of software testing:

- o **Integration Testing**
- o **System Testing**
- o **Acceptance Testing**

The higher the level, and hence the bigger and more complex the box, the more black-box testing method comes into use.

Techniques

Following are some techniques that can be used for designing black box tests.

- o *Equivalence Class Partitioning:* It is a software test design technique that involves dividing input values into valid and invalid partitions and selecting representative values from each partition as test data.

o **Invalid**	o **Valid**	o **Invalid**
o AB001	o XA0634	o XA06341
o A01123	o AC0001	o AB00011

- o *Boundary Value Analysis:* It is a software test design technique that involves the determination of boundaries for input values and selecting values that are at the boundaries and just inside/ outside of the boundaries as test data.

o **Invalid**	o **Valid**	o **Invalid**
o Min-1	o Min, min+1, max,Max-1	o Max+1
o ABCDE, ABCD, QWERT	o	o
o	o	o

- o *Cause-Effect Graphing:* It is a software test design technique that involves identifying the cases (input conditions) and effects (output conditions), producing a Cause-Effect Graph, and generating test cases accordingly.

- o *Username should be 6 chars (XA0634)*

- o *Password should be 6-48 (Qwert) pass only 5 chars*

- o *Result (Effect) should be invalid*

Advantages

- o   Tester need not know programming languages or how the software has been implemented.
- o   Test cases can be designed as soon as the specifications are complete.

Disadvantages

- o   Without clear specifications, which is the situation in many projects, test cases will be difficult to design.
- o   Tests can be redundant if the software designer/developer has already run a test case.

White box testing (or) glass box (or) clear box testing:

      If one performs testing on structural part (**coding part**) of an application then that method of testing is known as **white box testing**. Usually, **developers** are white box testers will perform.

After successful completion of code - Code Review (white box testing)

WHITE BOX TESTING (also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing) is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.

- o   **white-box testing:** Testing based on an analysis of the internal structure of the component or system.
- o   **white-box test design technique:** Procedure to derive and/or select test cases based on an analysis of the internal structure of a component or system.

Example

A tester, usually a developer as well, studies the implementation code of a certain field on a webpage, determines all legal (valid and invalid) AND illegal inputs and verifies the outputs against the expected outcomes, which is also determined by studying the implementation code.

White Box Testing is like the work of a mechanic who examines the engine to see why the car is not moving.

Levels

White Box Testing method is applicable to the following levels of software testing:

- o   Unit Testing: For testing paths within a unit.
- o   Integration Testing: For testing paths between units.
- o   System Testing: For testing paths between subsystems.

However, it is mainly applied to Unit Testing.

Advantages

- o Testing can be commenced at an earlier stage. One need not wait for the GUI to be available.
- o Testing is more thorough, with the possibility of covering most paths.

Disadvantages

- o Test script maintenance can be a burden if the implementation changes too frequently.

Gray box testing: if one performs testing on both the functional part as well as the structural part of an application than that method of testing is known as gray box testing.

Usually, the test engineer who has the structural knowledge will perform gray box testing.

Levels of testing: there are 5 levels of testing

1. Unit level testing

2. Module level testing

3. Integration level testing

4. System level testing

5. User acceptances level testing

1. **Unit level testing: It** is defined as smallest part of an application (**program**) also uses **structural part (coding part)**. In this stage the developers are **white box testers (Developers)** will check every unit as well as the combination of unit in order to conform whether they are working fine or not.
   Error:
   An error is a mistake, misconception, or misunderstanding on the part of a software developer.

2. **Module level testing:** module is defined as a **group of related features to perform a major task** in an application. In this stage the **black box test engineers** will check the **functional part** (**GUI**)of the module. (Only test functional part no relation in programming).

   Bug:
   Actual and expected requirements deviation (bug found during black box testing).

 3. **Integration level testing:** in this stage the **developer** will develop a **interfaces** in order to integrate the modules. The **white box testers** will check whether the interfaces are working fine or not. Developers may follow any one of the following approaches to integrate module.
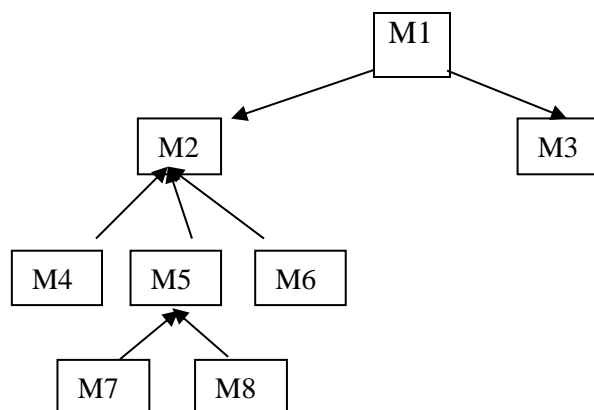
a) **Top-down approach**: in this the parent module will be develop first and then related child module will be developed and integrated.
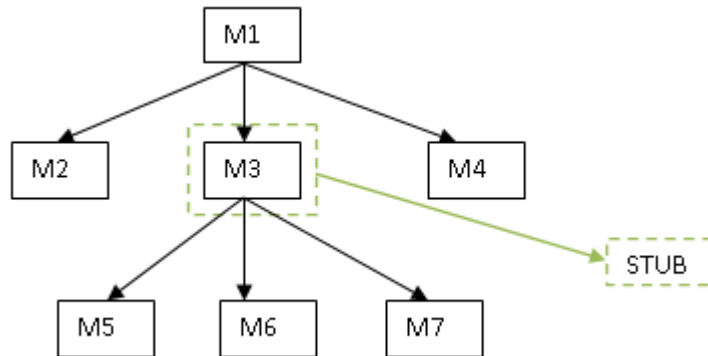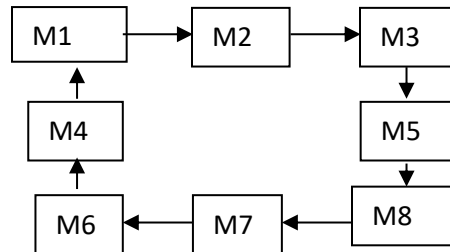

b) **Bottom-up approach**: in this the child module will be develop first and then related parent module will be developed and integrated.
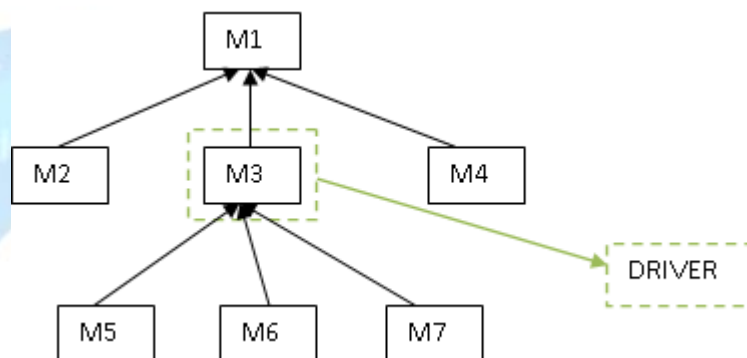
c) **Hybrid approach (or) sand-witch approach**: in this approach is mixed both top down and bottom up of approaches

d) **Big-bang approach**: in this approach one will wait till all the modules are develop and will integrate at a time at the end.

Stub: while integrating the modules in **top-down** approach if at any one mandatory module is missing then that module is replaced with a temporary program known as **stub.**

Driver: while integrating the modules in **bottom-up** approach if at all any one mandatory modules are missing than that module is replaced with a temporary program known as **driver**.

4. **System level testing:** in this stage the **black box test** engineers will conduct so many types

Of testing like **load testing**, **performance testing**, **stress testing**, **compatibility testing**, **system integration testing** and etc.., One of them important is **system integration testing**.

System integration testing: in this type of testing, one will do the actions at one module and

Check the reflections in all the related area of the application. May be in the same module or in the different module reflection.

Ex. Net Banking – Balance check – Fund Transfer -- Balance check – Statements

Balance = 10000

Fund Transfer = 5000 transfer to abc person. (action perform)

Balance = 5000

Statement = 5000

Ex: financial → sales → purchase → ware house → integration

5. **User acceptances level testing:** in this stage the **black box test engineers/ end users** will perform testing on the desired areas check the user in order to make him accept the application.

Alpha, beta and gamma testing user acceptance testing

Alpha Testing normally takes place in the **development environment** (inside the organization) and is usually done by **internal staff (black box test engineers)**.

Long before the product is even released to external testers or customers. Also, potential user groups might conduct Alpha Tests, but the important thing here is that it takes place in the development environment.

Based on the feedback – collected from the alpha testers – development teams then fix certain issues and improve the usability of the product.

Beta Testing, also known as "**field testing**", takes place in the **customer's environment** and involves some extensive testing by a **group of customers** who use the system in their environment or **third-party expert testers.** These beta testers then provide feedback, which in turn leads to improvements of the product.

Note: Alpha and Beta Testing are done before the software is released to all customers.

Gamma Testing?

This is the final phase of testing and is performed when the product is ready for release with specific requirements.

Not all the in-house testing activities which are decided to go through this testing phase are performed on the product.

|	Alpha	Beta	Gamma
Why?	validate software in all perspective, ensure readiness for beta testing	get end users' feedback, ensure readiness for release	check software readiness to the specified requirements
When?	at the end of development process	after alpha testing	after beta testing
Who?	in-house development or QA team, customer	a group of real end users	limited number of end users
What get?	bugs, blockers, missing features and others	ideas to improve usability, compatibility, functionality	ideas for updates in upcoming versions
What next?	beta testing	gamma testing	gold release / final release

Environment :(tiers) it is a group of hardware component with some basic software which can hold presentation logic, business logic and database logic.   (or)

Environment is defined as combination of presentation layer, business layer database layer.

Types of environment: there are 4 types of environments

1. Stand-alone environment (or) single tier environment

2. Client server environment (or) two tier environment

3. Web environment (or) three tier environment

4. Distributed environment (or) n tire environment or multiple architecture

Stand-alone environment: this environment contains only one tier. The presentation layer, business layer, database layer will be present in the same tier whenever the application need to be used by a single user at a time then this environment suggested.
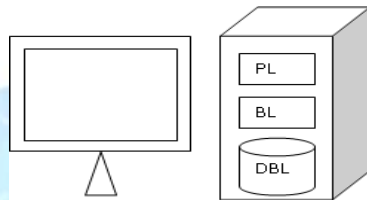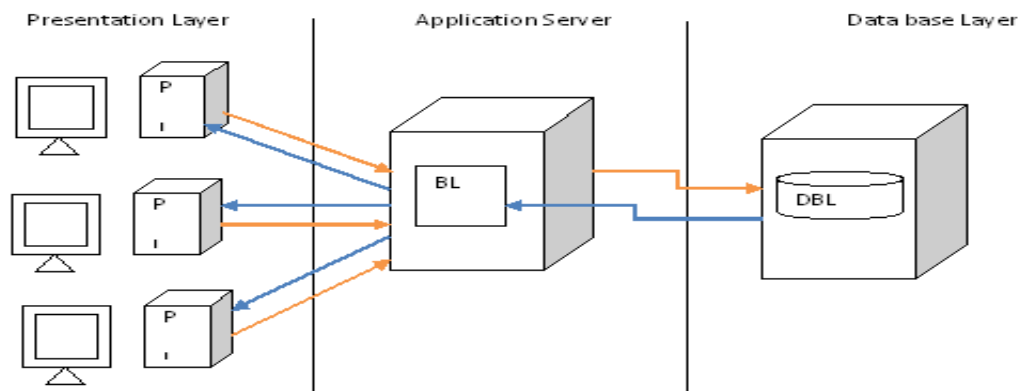

Client server environment: this environment contains two tier one is client other one is for server. Whenever the application is need to be used by multiple users by sharing the common data and also we wants to access the application usually a single compound very fast then this environment can be very suggested.
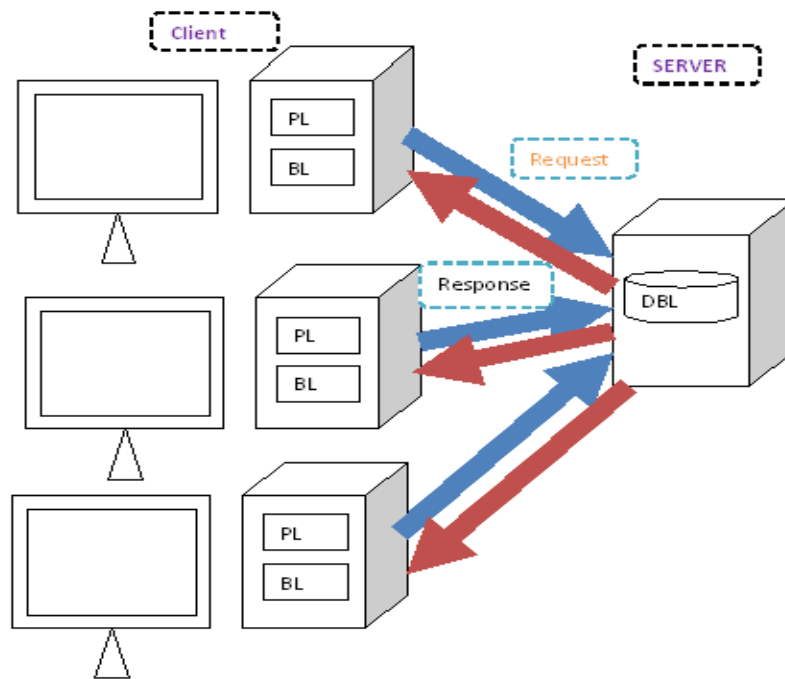
**Web (Internet) environment:** in this environment 3 types will be there

    i. Client   ii. Application server iii. Data base server

    Presentation layer will be present each and every client, business layer will be present in Application server. Whenever the application needs to be used all over the world by multiple users and the feature updating need to be done easily as well as the business logic need to be secure then this environment can be suggested.

**Distributed environment:** this environment is similar to the web environment but the business logic is distributed among number of application server in order to distribute the load and increase the performance.

Whenever the application needs to be used all over the huge number of people then this environment can be suggested.
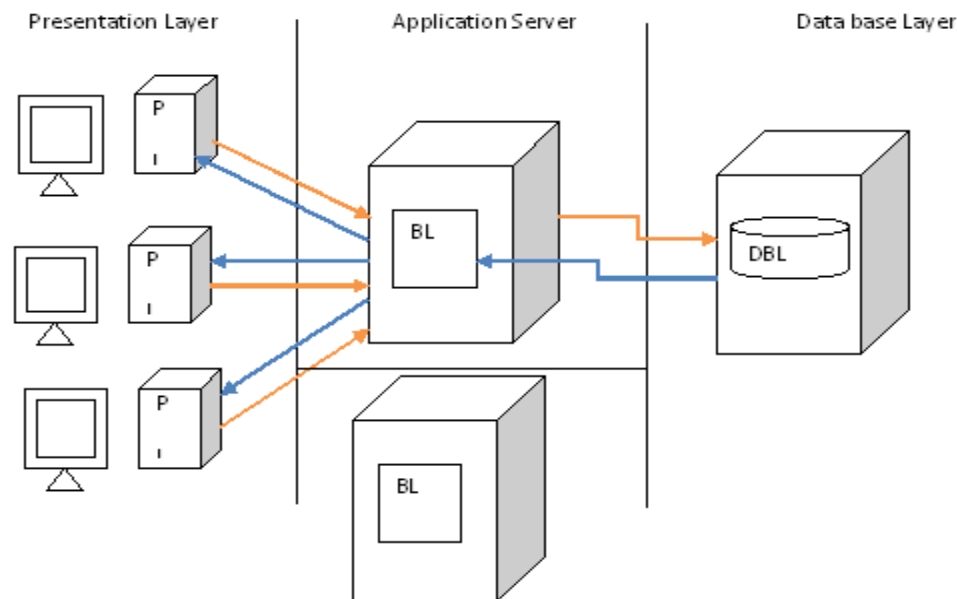
SOFTWARE DEVELOPMENT PROCESS MODELS

1. **Water fall Model:**
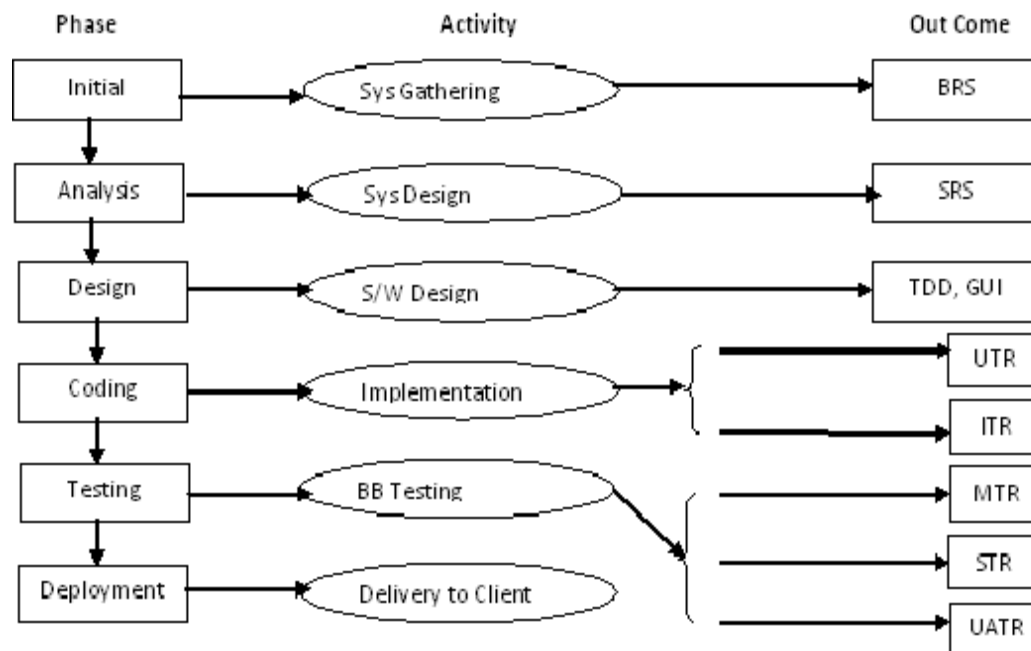
In waterfall model all the phases will move from top to bottom and should not go bottom to up. Once completion of each phase then only we are moving to the next phase.

How the water falls from top to bottom?

in the same manner this model going to work from requirements to delivery and maintenance phase.

Advantages:

It is a simple model project monitoring and maintenance very easy.

For any project if all the requirements are stable, then this is the suitable model to deliver quality product.

Minimal budget is required for the project if requirement are stable.

Draw backs: can't accept the new requirement in the middle of project.
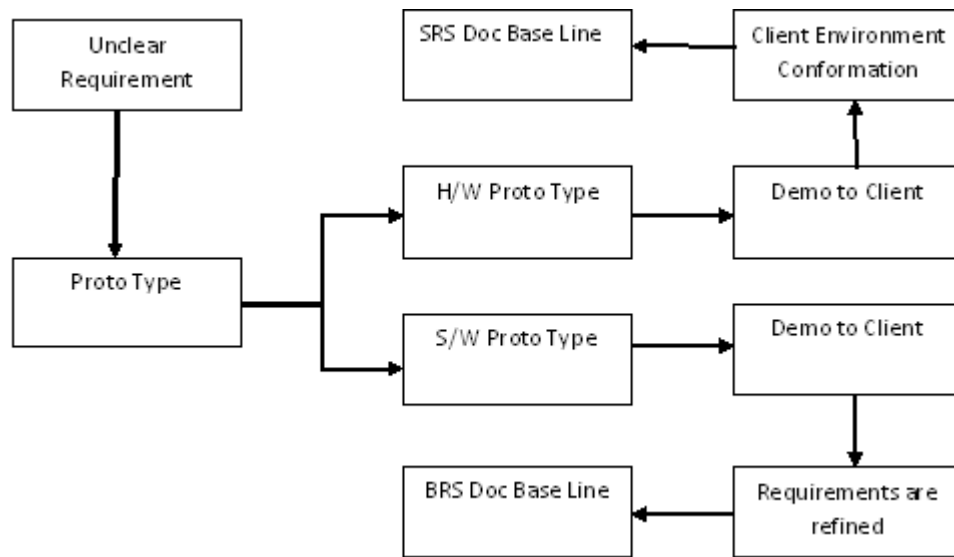
> **UTR** Unit Test Requirement
>
> **ITR** Integration Test Requirement
>
> **MTR** Module Test requirement
>
> **STR** System Test Requirement
>
> **UATR** User Acceptance Test Requirement

2. Prototype model:

Based on the requirements received from the client, company is going to start preparing prototype by covering all the requirements. Company will share the same to the client for approval.

If at all client reviewed and updated few more changes then we have to update our prototype and share it with the client. This process will continue till either acceptance or rejection of the prototype.

If the client rejected the prototype, then we have to stop our work on it.

If at all he accepted the prototype then we have to kick start with the actual work on it.

Advantages: whenever the customer is not clear with the requirements then this is the best suitable model.

Drawbacks: it is not a complete software development model

- ➢ Slightly time-consuming model
- ➢ Company should bear the cost of prototype
- ➢ Customer may limit this requirement by striking to the prototype

3. Evolutionary model:

Rough requirements specification

↓

Identify the core and other parts to be developed incrementally

↓

Develop the core part using an iterative waterfall model ←

↓

Collect customer feedback and modify requirements

↓

Develop the next identified features using an iterative waterfall model
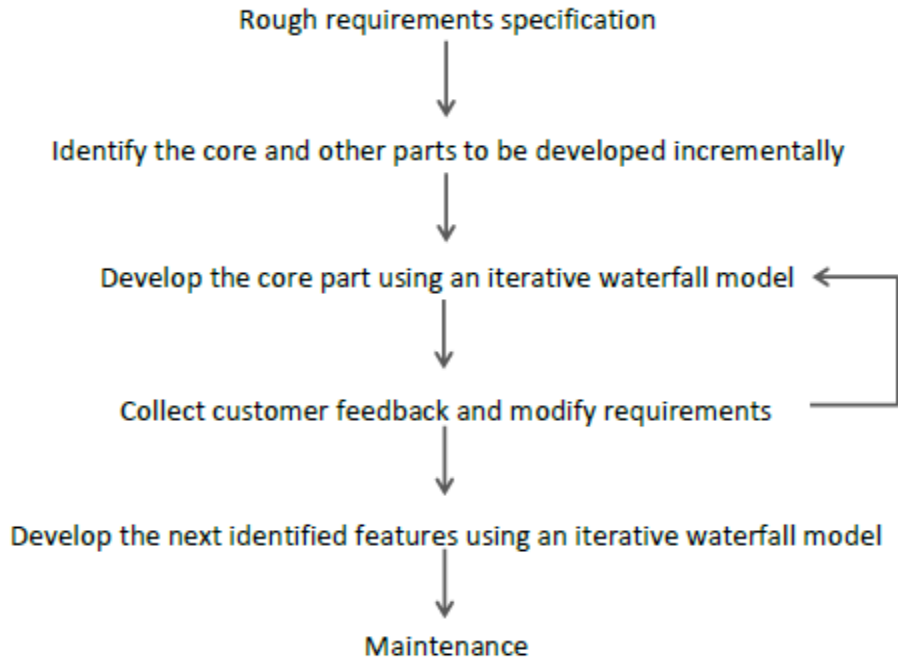
↓

Maintenance

Figure 6: Evolutionary Model of Software Development

Whenever the customers' requirements accept middle of the project then this is the best suitable model.

Drawbacks:

➤ time consuming model

➤ Costly model

➤ Deadlines can't be properly defined

➤ No transferring

➤ Project monitoring & maintenance is difficult.

4. Spiral Model:

Defining the objectives/ work analysis /constraints

Refining and Planning for the next cycle

Risk toots cause analysis Estimati on Conting encies

Implementation (SDLC)

1. Objectives determination and identify alternative solutions

2. Identify and resolve Risks

3. Develop next version of the Product

4. Review and plan for the next Phase
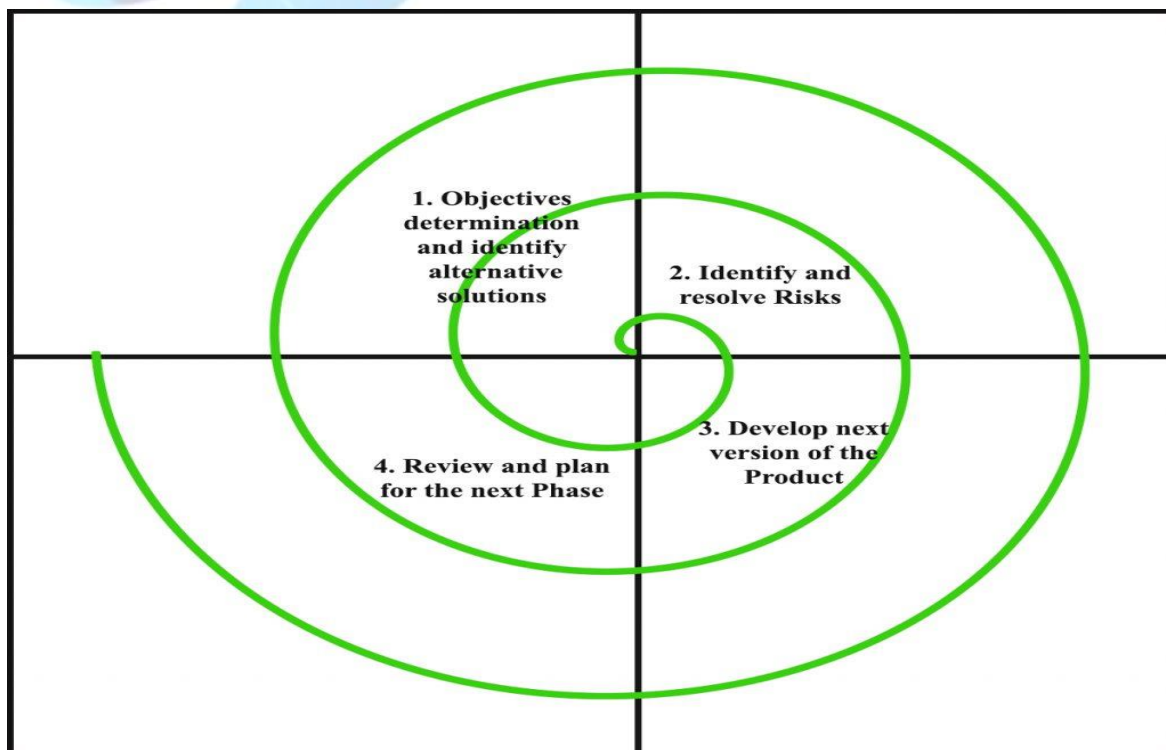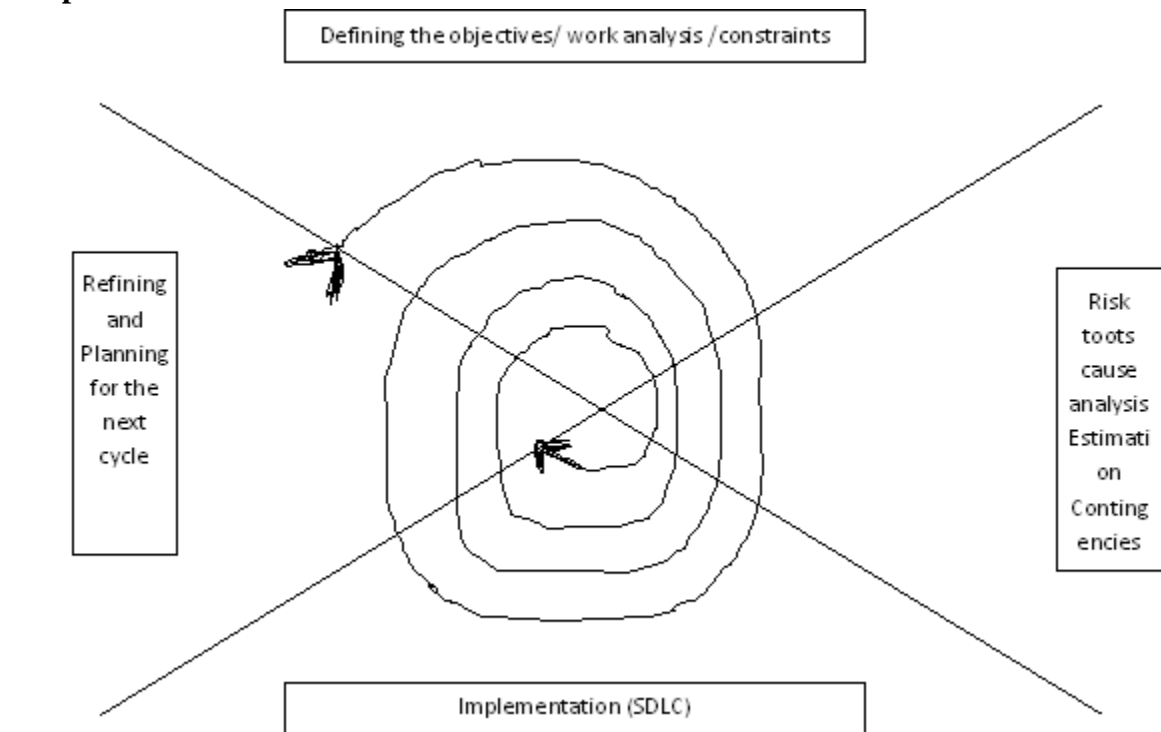
This model is going to overcome few of the disadvantages getting in the waterfall model.

In this model we will deliver entire project in **multiple cycles** and we are going to deliver few of the requirements in each cycle so that in **'n'** number of cycles we will deliver entire project.

If at all client asked us to change or update any of the requirements, we should need to incorporate and continue with the next cycle.

Advantages: whenever the project is highly risk based then it is the best suitable model

High visibility will be there for client on his product or project.
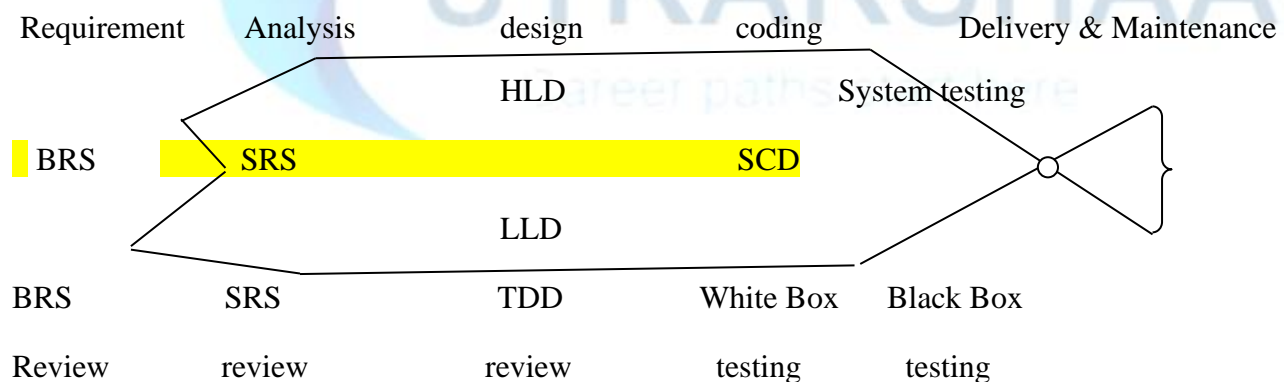
For every cycle client will be receive some part of the application based on that client may give feedback to the company for updating or changing the any of the features.

Risk management will be handled properly for each cycle, it will allow us to deliver in on time, if at all we did not get any major changes from the client.

Disadvantages:　　1. Time consuming model

　　　　　　　　　　2. Costly model

　　　　　　　　　　3. Risk root analysis is not easy model

5. Fish Model:

Requirement	Analysis	design	coding	Delivery & Maintenance
		HLD		System testing
BRS	SRS		SCD	
		LLD		
BRS	SRS	TDD	White Box	Black Box
Review	review	review	testing	testing

Advantages: as both verification & validation is done the outcome will be a quality product

Draw backs: time consuming model, Costly model

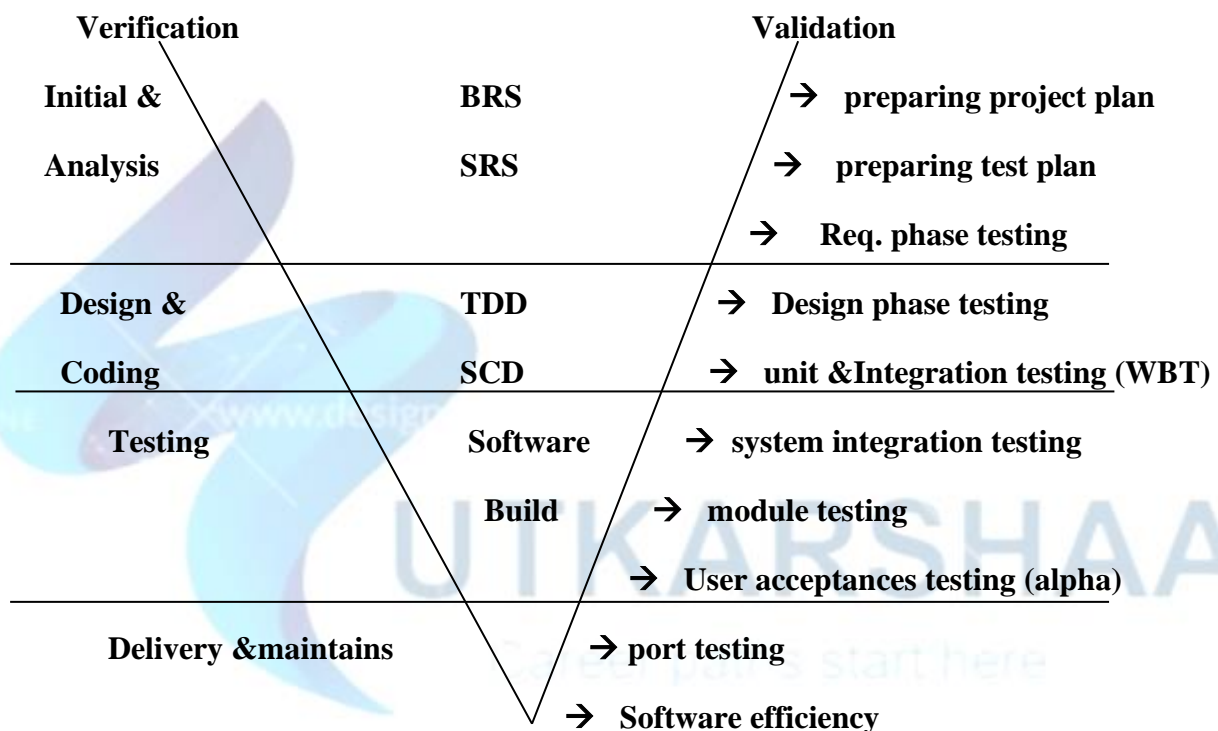5.　　　　**Verification & Validation model (V- Model):**

We can also call it as a verification and validation model.

By using verification in all the phases we are going to verify all the documents and standards whether we are on right track to deliver right product to the customer.

Testing team involvement will be there right from starting of the project.

Parallel to all the phases we will involve and we will design various scenarios and test cases.

Once we get the build from development team we need to use designed test cases and go ahead with test case execution.

Verification   **Validation**

Initial &   **BRS**   → **preparing project plan**

Analysis   **SRS**   → **preparing test plan**

                                                    → **Req. phase testing**

Design &   **TDD**   → **Design phase testing**

Coding   **SCD**   → **unit &Integration testing** (WBT)

Testing   **Software**   → **system integration testing**

                Build   → **module testing**

                        → **User acceptances testing (alpha)**

Delivery &maintains   → **port testing**

                        → **Software efficiency**

Test software changes DRE = $\underline{A}$

$A+B$

Defect Removal Efficiency

A = defects found by the testing team (10 defects)

B = defects raised by the customer (3 defects)

As both verification and validation and test management process is concentrated definitely the Outcomes **quality product**.

Drawbacks: i. time consuming model

ii. Costly model

Thick client: if at all the client machine is having both the presentation logic & business logic than it has known has thick.

Thin client: if at all the client machine only the presentation logic than known as thin client.

MTC (Mobile Thin Client) – access the dedicated machine (Citrix gateway)

Verification (un-conversional) :(starting to ending of sdlc) it is a process of checking conducted on each & every role in their organization. In order to conform weather they are doing their work according to the company guide lines or not.

Validation (conversional) :(developed product) Validation is a process of checking conducted on the developed product are its related part in order to conform whether they are working expectations or not.

Agile Methodology (Model)

(90%)

AGILE MODEL

For any projects/products which are **not having stable requirements** then they have to approach **agile model** as it is most suitable and more visibility to the **client** on **daily basis.**

Scrum master is the person who is going to manage all the testing and development activities on daily basis, he will monitor and ensure everything is going on as per the sprint time lines.

1) Scrum capacity
2) According to scrum capacity pick the stories from backlog.

5 developers * 8 = 400 hrs/sprint

1 tester * 8 = 72 hrs/sprint

Scrum Master

Total capacity = 472 hrs/sprint

30 minutes * 10 days = 300 minutes = 5 hrs* 6 = 30 hours

472-30 = 442 hrs

300 hrs

50 hrs (scrum master pick user story from backlog)

6 story points deliver 10 days (60 story points)

 daily standup

        80 hrs

     640 hrs

Equity (Story) – 8 days – story points (roughly 1 story point consider as 1 day/1 developer)

Testers - 2 days – (2 story point)

Hi User name (Story) – 3 story (2- developer and 1- tester) 24 hrs

1 PO

1 SM

8 Story – 6 story achieve (deliver) (2 story pending) (spill over) – next sprint add

Product owner is the person from client side and he will cross check all the activities from both development and testing team are working according to the client requirements.

All the requirements are splited into stories by the scrum master and he will derive tasks from the stories based on the priority to the client and depends on the time lines given by the team (Estimations) and he will include all those tasks into that sprint and according teams needs to work. Usually sprint duration will be nearly 2-3 weeks. This process will continue till end of all the stories given by the client.
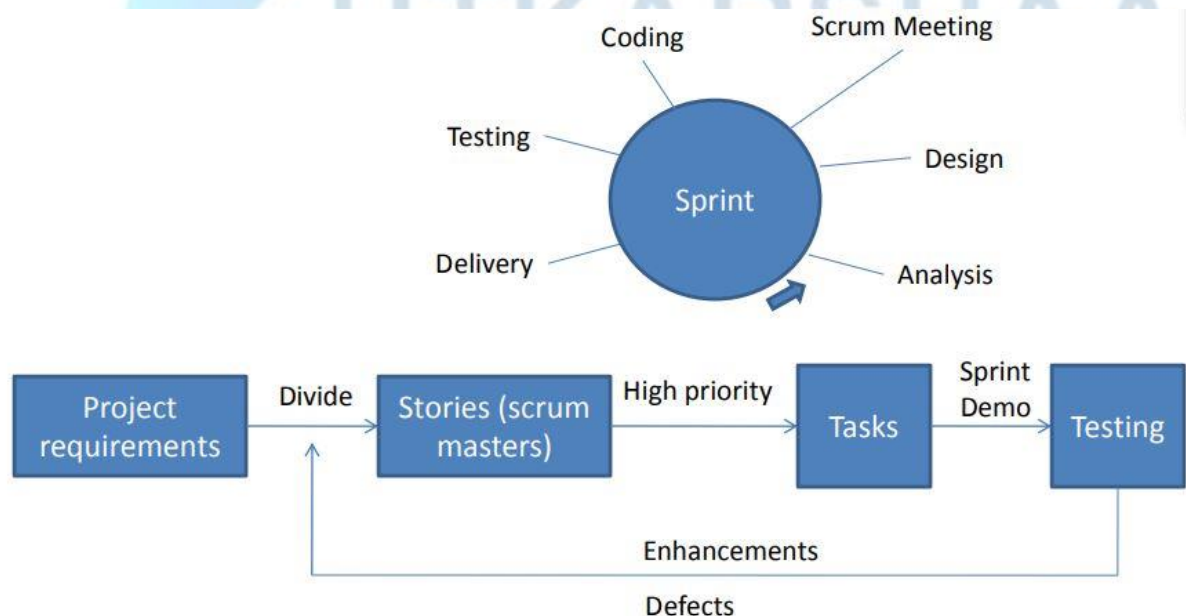
10 days sprint (2 weeks)

80 – 12 hours for scrum ceremony 68*10 = 680 hours (85) – 55 to 60 (deliver) 25 to 30 story (Spillover) – next sprint

Login form story – Estimation (story points) 1 - (8 hours) 3 (24 hours) 5 , 8

10 persons (team) capacity (7.5h) - 59 hours – 442 hrs

Sprint Meetings (Scrum ceremony)

1) **Daily standup - call 15 to 30 minutes**
2) **Sprint planning (2hours) 2 planning meetings in one sprint (4 hours)**
3) **Sprint review (1 hour)**
4) **Sprint retrospective (1 to 2h)**
5) **Sprint backlog refinement (2h)**

5 developers

2 QA

1 BA

1 PO

1 Scrum master

Capacity

Scrum Meeting/Daily Stand up meetings:

This is the meeting conducted by the scrum masters with development and testing teams for tracking the project/product development activities.

If at all anybody struck (blocker) with any issues they can update those issues and resolutions will be found in this meeting.

Here every body need to update tasks covered on that day and planning of tomorrow and regarding the issues.

Daily Stand up Agenda:
To discuss on Tasks worked on yesterday
Planned tasks for today
Impediments (Blocker) if any

Advantages

High visibility will be there to the client

On daily basis client will check this product or project features if at all he found to change anything then immediately he can update to the scrum master so that development teams will try to change those feature instantly in coming sprints.

It won't take more time for the development and testing activities.

Disadvantages

Poor documentation will be there

Both development and testing team will be allocated with some tasks. So that teams feels more busy with work.

Sprint Planning Agenda:
- Product owner(s) again explains the **prioritized** stories from the product backlog
- Team can ask question(s) on each story
- Team estimate each story with **story points**
- Team knows the dependencies between estimated stories
- Scrum Master pulls stories into the Sprint according to the priority of the Product Backlog and team capacity

Scrum

Scrum meetings (ceremony)

1) **Daily Sprint (15 to 30 minutes daily call, impediments or blocker for everyone) What we did yesterday, what we do today**
2) **Sprint planning (pick the story from backlog and prioritize the story, assign the story points) p- person d-day effort (1 story point consider as 1pd) 53 story point 2**
3) **Sprint Review(completed story presents in front of customer, QA session) last day of sprint - 3**
   Product owner
   - **Achievements**
   - **Login Functionality**
   - **Dashboard Functionality**

4) **Sprint Retrospective (What went well (congratulate to the Dev. /QA for achieving their task), what needs to be improved, action plan for what needs to be improved) (what did not went well.) last day of the sprint after review meeting or next day. 4**
5) **Sprint backlog (backlog refinement) – story explained by PO or BA. – 1 Before the Sprint Start (2-3 days before 2 hrs)**

(how many days sprint)
Scrum board phases
1) **To-do**
2) **Ready/Reopen**
3) **In Progress**
4) **Code Review**
5) **To Test**
6) **Testing**
7) **Acceptance**
8) **Master**
9) **On Hold**
10) **Done**

Types of Testing:

What is Functional Testing? (Mandatory)

Functional testing is a type of testing which verifies that each **function** of the software application operates in conformance with the requirement specification.

This testing mainly involves **black box testing**, and it is not concerned about the source code of the application.

Every functionality of the system is tested by providing appropriate input, verifying the output and comparing the actual results with the expected results. This testing involves checking of User Interface, APIs, Database, security, client/ server applications and functionality of the Application under Test. The testing can be done either **manually or using automation.**

What is Non-Functional Testing? (Optional)

Non-functional testing is a type of testing to check non-functional aspects (**performance**, usability, reliability, etc.) of a software application. It is explicitly designed to test the readiness of a system as per non-functional parameters which are never addressed by functional testing.

A good example of non-functional test would be to check how many people can simultaneously login into a software.

Non-functional testing is equally important as functional testing and affects client satisfaction.

Functional Vs. Non-Functional Testing

Parameters	Functional	Non-functional testing
Execution	It is performed before non-functional testing.	It is performed after the functional testing.
Focus area	It is based on customer's requirements.	It focusses on customer's expectation.
Requirement	It is easy to define functional requirements.	It is difficult to define the requirements for non-functional testing.
Usage	Helps to validate the behavior of the application.	Helps to validate the performance of the application.
Objective	Carried out to validate software actions.	It is done to validate the performance of the software.

Requirements	Functional testing is carried out using the functional specification.	This kind of testing is carried out by performance specifications
Manual testing	Functional testing is easy to execute by manual testing.	It's very hard to perform non-functional testing manually.
Functionality	It describes what the product does.	It describes how the product works.
Example Test Case	Check login functionality.	The dashboard should load in 2 seconds.
Testing Types	Examples of Functional Testing Types • Unit testing • Smoke testing • User Acceptance • Integration Testing • Regression testing • Localization • Globalization • Interoperability	Examples of Non-functional Testing Types • Performance Testing • Volume Testing • Scalability • Usability Testing • Load Testing • Stress Testing • Compliance Testing • Portability Testing • Disaster Recover Testing

Build acceptance testing (or) Build verification (or) Sanity Testing (Smoke Testing):

It is a type of testing in which one will conduct over all testing on the released build in order to conform whether it is proper for conducting detailed testing(or) not.

To do the same one will check the following:

I.   Whether the build can be properly installed into the environment or not

II.   Whether one can navigate to all the pages and applications or not

III. Whether all the important objects available or not

IV.   Whether the required connections are properly established or not

In some company's they will call this type of testing is **smoke testing** also. But in   some company's soon after the build is developed just before releasing the build developers will check whether the build proper or not that is known as **smoke testing** and once the build is released the test engineer will once again will check the same that is known as build acceptance testing & sanity testing.

Regression testing: It is a type of testing in which one will perform testing on the already tested functionality again and again usually you do it 2 scenarios.

> ➢ Whenever the defects are raised by the test engineer, after rectification the next build is released them the test engineer will check defect functionality as well as the related functionality once again.

> ➢ Whenever some new features are added to the application, next build is released then the test engineers will check all the related functionality of those new features once again in order to conform whether they are working fine or not.

Note: testing the new feature for the first time is new testing but not regression testing.

Re-testing: It is a type of testing in which one will perform testing on the same functionality again and again with **multiple sets of data** in order to come to a conclusion whether it working fine or not.

Note: re testing starts from the first build to last build

> ➢ Regression testing starts from **second build** and continue up to last build

> ➢ During regression also retesting will be done usually people call this as re and regression testing.

Alpha testing: It is a type of user acceptances testing conducted in the **software company** by their test engineers in the presence of the user (Stack holder, customer) just before delivery.

Beta testing: It is a type of user acceptance testing in which one (either end users or third party testing experts) will perform testing in the **customers place** just before actual implementation.

Gamma testing:

It is the final stage of the testing process conducted before software release. It makes sure that the product is ready for market release according to all the specified requirements. Gamma testing focuses on software security and functionality. But it does not include any in-house QA activities.

Static testing: It is types of testing in which one will perform testing on the application are its related factors without performing **any actions**.

Ex: GUI testing, document testing, code review etc.,

Dynamic Testing: It is a type of testing in which one will perform the application or its related factors by performing some actions.

Installation testing: It is a type of testing in which one will install the application into the environment by following the guide lines provided in the deployment document in order conform whether those guide lines are suitable for installing the application in to the environment or not.

Ex: installation process

Compatibility testing :(suitable**)** it is a type of testing in which one will install the application into the multiple environments prepare with different combination in order to conform whether the application is suitable to those environments or not.

This type of testing comes under system level testing.

Here we are going to check entire application is supporting multiple software, hardware, browsers.

Compatibility will give us result our application is supporting all the browsers, operating system, environments, based on the client request.

Note: what are all the environments we have to check in the compatibility given by the client. Usually, this type of testing is important for product rather than project.

Monkey testing: It is a type of testing in which one will perform **abnormal action intentionally** on the application in order to check the stability (standby) of application.

Exploratory testing: It is a type of testing in which one will (**domain expert**) test the application without having the knowledge the application just by exploring the application.

Exploring: having the basic knowledge of some concept doing something and knowing more about it is knowing as exploring.

Usability testing: It is a type of testing in which one will check the user friendliness of the application.

It is the type of testing for finding the application graphical user interface (GUI) is user friendly or not.
Here we have to justify the entire application GUI using client or end user perspective whether it is easily understanding to the end user or not.
For doing the usability testing we need to have end user expectation on our application.
Note: UAT team is having high visibility on client or end user expectations then it will be easier for them to do the usability testing.

Ex; application open cursor wait starting point or not.

End to end testing: It is a type of testing in which one will perform testing on the end to end scenarios or the application.

Port testing: It is a type of testing in which one will install the application into the original customer environment and check it is suitable environment or not.

Soak testing / reliability testing: It is a type of testing in which one will perform testing one the application continues for long period of time in order to check the stability of the application.

Security testing: It is a type of testing in which one will check whether application is properly protected or not. To do the same black box test engineers will implement following types of testing.

1) **Authentication/ Authorization testing: It** is a type of testing in which one will enter different combinations of user names and passwords and check whether only the authorized people are accessing people or not.
2) **Direct URL testing: It** is a type of testing in which one will enter the direct some URL secure pages and check whether they are able to access them or not.
3) **Fire wall in case testing/ User prevails testing: It** is a type of testing in which one will enter into the application as a one level of user and will try to access the application beyond its limits and check weather is able to access or not.

Mutation Testing: It is a type of software testing in which certain statements of the source code are changed/mutated to check if the test cases are able to find errors in source code. The goal of Mutation Testing is ensuring the quality of test cases in terms of robustness that it should fail the mutated source code.

Ad-hoc (temporary) testing: It is a type of testing in which one will (test engineer) perform testing on the application in their own style after understanding the requirements clearly. Usually this type of testing is after formal testing.

We have to understand the functionality (requirement)

If we have some doubts then clarify all doubts from BA.

We prepare test cases for req functionality

We will wait till developer handover their build.

Once developer handover their first build then we start from

1) Sanity/ smoke/BV/BA
2) If we have found some bugs then report those bugs to developer
3) Developer fix those bugs and resend the next build to testing department.
4) Regression Testing.

No test case

Due to the lack of time, we could not able to manage the testing activities effectively then adhoc testing will be the testing playing major role.

Here we are not maintaining documentation like test scenario, test cases because due to **insufficient time.**

Ad-hoc testing is the testing done by the experienced testers based on experience they are having understanding on the requirements.

Note: ad-hoc testing will not give 100% quality application but as per the client request we have to do and deliver application to the client.

Ex: temporary random

I18N Testing: Internationalization testing is a non-functional testing technique. It is a process of designing a software application that can be adapted to various languages and regions without any changes.

Internationalization (i18n)

Internationalization is a **design process** that ensures a product (usually a software application) can be adapted to various languages and regions **without requiring engineering changes** to the source code. Think of internationalization as readiness for localization. Internationalization is often written **i18n**, where 18 is the number of letters between i and n in the English word.

Localization, internationalization and globalization are highly interrelated.

Localization (l10n)

Localization is the process of adapting a product or content to a specific locale or market. Localization is sometimes written as **l10n**, where 10 is the number of letters between l and n. The aim of localization is to give a product the look and feel of having been created specifically for a target market, no matter their language, culture, or location.

Globalization (g11n)

Globalization, in the context of the language industry, refers to a broad range of processes necessary to prepare and launch products and activities internationally. Sometimes written **g11n**, globalization is an all-encompassing concept which applies to activities such as multilingual communication,

global-readiness of products and services, and processes and policies related to international trade, commerce, education, and more.

Example:

"Google Search Engine" has been taken as an example to explain localization, internationalization.

Localization (l10n):

Initially, Google developed the search engine in its own native language (English), culture and desired local "look and feel". The following image shows the Google search engine designed for USA.

Mail   Images   ⠿   ⬤

Google

🎤

Google Search   I'm Feeling Lucky
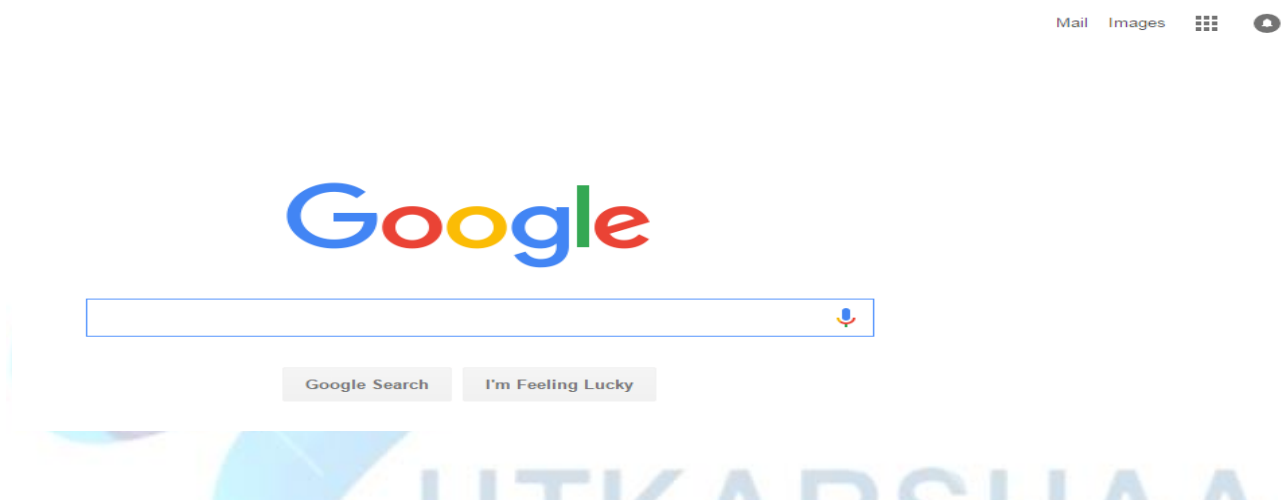
Internationalization (i18n):

As people are more comfortable in their native language, Google designed and developed its search products around the local geography, culture and language. Google is now available in 123 languages and has been localized completely or partially for many countries.

The Google search engine for "India" has been developed in English along with 9 popular and widely spoken languages of the country. By selecting any one of the specific language, the entire design of the page will be changed based on the language selected.
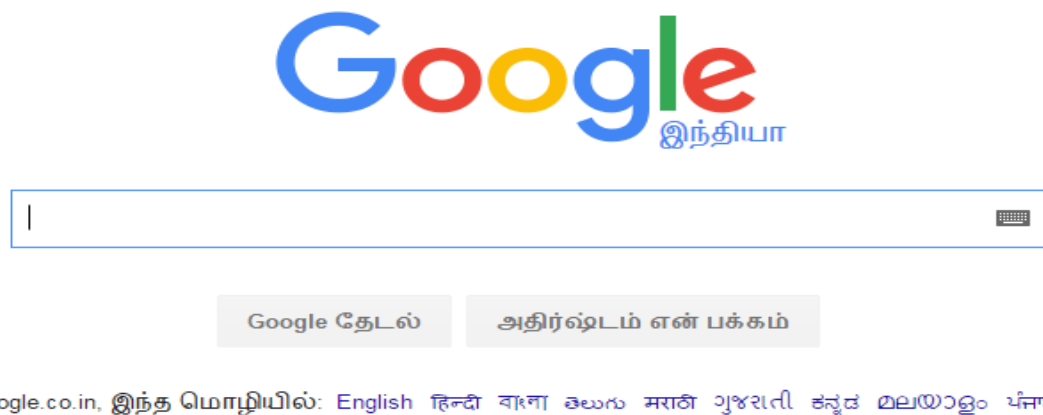
Thus in order to achieve the globalized standard and to make the software user-friendly, Internationalization testing has to be performed and it plays a major role.

Need For Internationalization (i18n) Testing:

If internationalization testing is not performed,

- It may have an adverse impact on product quality.
- The software may lack encoding of characters when converted to different languages.
- The software may crash or malfunction if the string is not supported in targeted languages.
- The font and font size may vary from language to language affecting the 'look and feel' of the application.
- It may not reach to intended audience.

Internationalization (i18n) Testing at Front end:

To perform internationalization testing, the testers have to focus on -**Language, Culture, and Region, Dates and Important events.**

- **First** testing should be done on a **user interface** such as alignment of texts, menus, and buttons, dialog boxes, images, and toolbar, prompt and alert messages.
- **Second** testing on **content localization** and **feature** should be performed on language specific properties files and on the region where the particular feature is enabled or disabled.
- **Third locale/culture awareness** testing should be performed on dates & number formats such as currencies, calendars, time, telephone number, zip code, etc.
- **Finally** testing on **file transfer** and **rendering** should be performed to make sure whether the file transfers are successful and to check whether the scripts supported to the website are correctly displayed.

The above steps, at a high level, covers front end testing.

Internationalization (i18n) Testing at Back end:

This process involves enabling the back end of a website to handle character encoding, form data submission, different languages and currencies and site search.

Testing at the back end requires an understanding of Content Management System (CMS) that is used to store, author and publish content on the websites. Thorough understanding of the database is highly important for testing at the back end.

Recent version of databases and Content Management System are already internationalized.
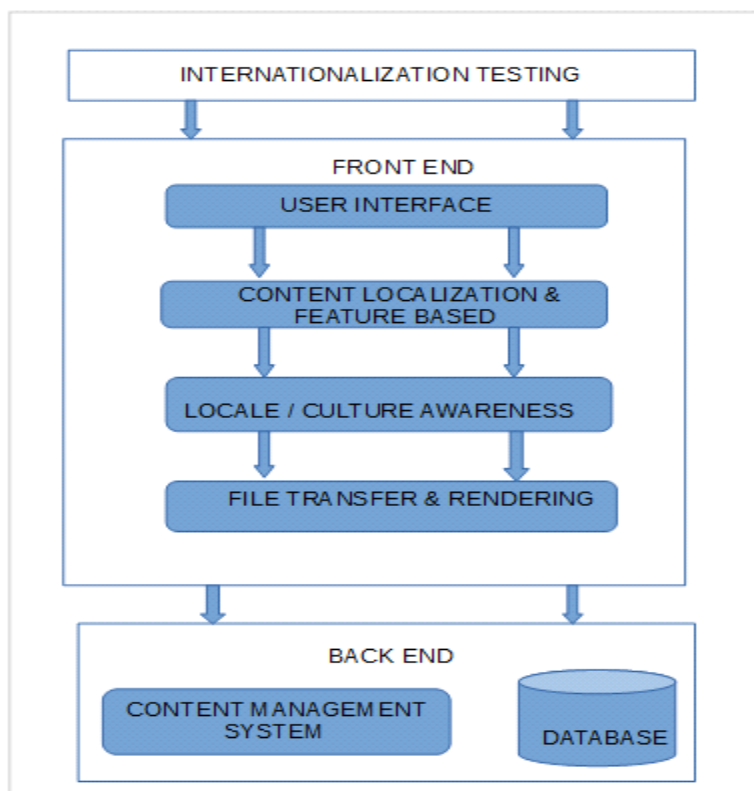
Advantages of Internationalization (i18n) Testing:

- Easier adaptation of software applications (or other content) to multiple locales
- Reduced time and cost for localization
- Single, internationalized source code for all versions of the product
- Simpler maintenance
- Improved quality and code architecture
- Reduced overall cost of ownership of the multiple versions of the product
- Adherence to international standards

Internationalization is a critical business process for any company producing multilingual products. It is important to consider the various markets a product will target and make the necessary adaptations early in product design and development.

SOFTWARE TESTING LIFE CYCLE (STLC): these are 6 phases

1. Test planning

2. Test development

3. Test execution

4. Result analysis

5. Bug tracking

6. Reporting

1. Test planning:

Plan: plan is a strategic document which contains some information that describes how to perform a task in an effective efficient and optimized way.

Optimization: It is a process on utilizing the available resources to their level best and get in the maximum possible output.

Test plan: It is a strategic document which contains some information that describes how to perform testing on an application in an effective, efficient and optimized way.

   Test lead will prepare the test plan.

Index / contents of the test plan:

   1.0 Introduction

   1.1 Objective

   1.2 Reference document

   2.0 Coverage of testing

   2.1 Features to be tested

   2.2 features not to be tested

   3.0 Test strategy

   3.1 Levels of testing

   3.2 Types of testing

   3.3 Test design Techniques

   3.4 Configuration management

   3.5 Test Metrics

   3.6 Terminology

3.7 Automation plan

3.8 List of automated tool

4.0 Base criteria

4.1 Acceptance criteria

4.2 Suspension criteria

5.0 Test deliverables

6.0 Test environment

7.0 Resource planning

8.0 Scheduling

9.0 Staffing & training

10.0 Risks & contingency (spiral model)

11.0 Assumption

12.0 Approval information

1.0 Introduction:

1.1 Objective: the purpose of the document will be clearly described here in this section.

1.2 Reference document: the list of all the documents that are referred by the test level will be listed out here in this section.

   Ex: system requirement specification project plan, SRS, Kick of meeting

2.0 Coverage of testing

2.1 Feature to be tested: the list of the entire feature that are within the scope and need to be tested will be mentioned here in this section.

2.2 Feature not to be tested: the list of the entire feature that is not planned for testing will   be mention here in this section.

   Ex: 1. Out of scope features

      2. Low risk features

      3. Features that are plan to be in corporate feature

      4. Feature that are skipped based on the time construction

3.0 Test Strategy: test strategy is a organization level term which is common for all the project in the application.

Test plan is a project level term which is specific for particular project in the organization.

3.1 Levels of testing: the list of all the levels of testing that are maintained in that company will be listed out in this section.

3.2 Types of testing: the list of all the types of testing that are perform in that company will be listed out here in this section.

3.3 Test design techniques: the list of all the techniques used during developing the test cases in that company will be mentioned here in this company.

Boundary value analysis + Equivalence class partition

Strictly concentrate on boundary

Password 8-15 chars

7 8 9 14 15 16   0 to 7, 8 to 15, 16 to 24

3.4 Configuration Management:

Configuration management determines clearly about the items that make up the software or system. These items include **source code, test scripts, third-party software, hardware, data, and both development and test documentation**. It ensures that these items are managed carefully, thoroughly and attentively during the entire project and product life cycle. It is sometimes referred to as **IT automation.**

The CM helps to improve performance, reliability, or maintainability; extend life; reduce cost; reduce risk and liability.

Example: *Suppose your company creates an avionics system that includes an LCD screen from a third-party company. Over the time, the specific LCD screen becomes obsolete and you need to find a suitable replacement to continue your production. The CM policy would identify who in which organization is authorized to approve utilizing a different but compatible version of the original LCD screen.*

3.4.1 Configuration Management Process
The configuration management process is comprised of 5 disciplines that will establish a product's baseline, and manage any changes over time.

3.4.1.1 Planning and Management*: This guides the design of the product, who has what responsibilities, and includes which **tools and procedures to use**. It identifies **third-party requirements** and the audit and review process.*

3.4.1.2 Identification*: Allows for setting baselines and identifying the configuration of assets (hardware) and/or software.*

3.4.1.3 Control*: Maintain control of configurations, change requests, and execution of changes to a system and its documentation.*

3.4.1.4 Status Accounting*: The process of recording and reporting configuration item descriptions and changes from the baseline over the item's life-cycle. If a problem arises, this allows for a quick determination of the baseline configuration and changes that have taken place over time.*

3.4.1.5 Verification/Audit*: Be able to audit what you have implemented and maintain positive control of all managed product*

3.5 Test Metrics: the list of all the metrics that are maintained in that company will be listed out clear in this section. (Noted in strategy document)

3.6 Terminology: the list of all the terms used in that company alone with meanings will be mention here in this section.

3. 7Automation plan: (regression testing presented automation) the list of all the areas that are plan for automation in that company will be listed out here in this section.

3.8 List of automated tool: the list of all the automated tools that are used in that company will be mentioned here in this section.

4.0 **Base criteria:**

4.1 Acceptance criteria: when to stop the testing on that application will be clearly mention in this section.

4.2 Suspension criteria: when to suspend the testing on suspend the build will be clearly mention here in this section.

If the suspension criteria are met during testing, the active test cycle will be **suspended** until the criteria are **resolved**. Example: If your team members report that there are **40%** of test cases failed, you should **suspend** testing until the development team fixes all the failed cases.
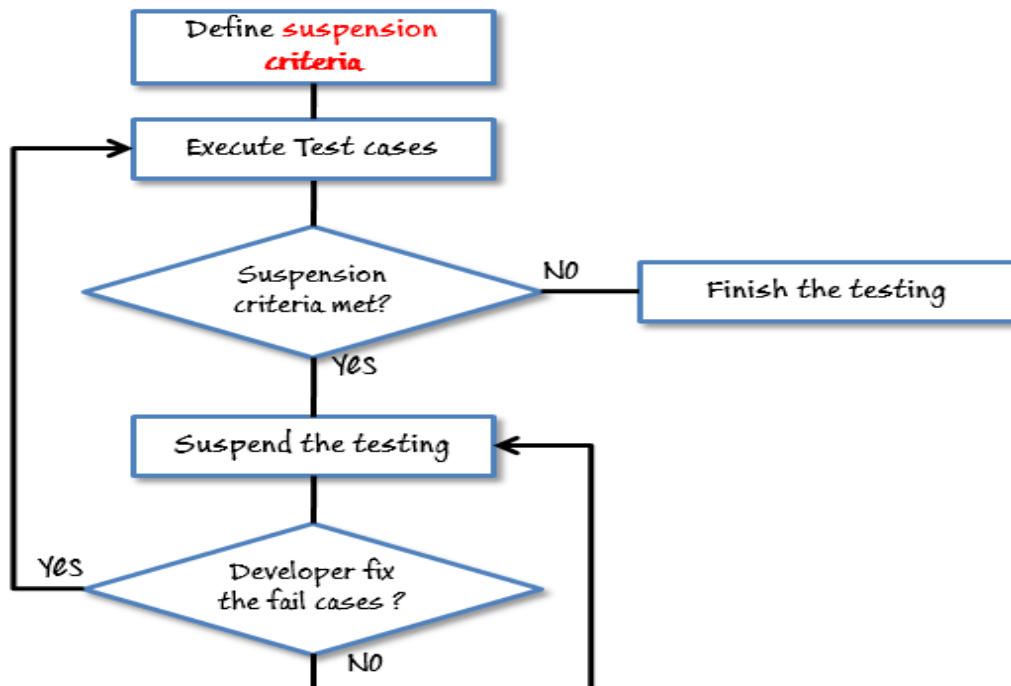
5.0 Test deliverable: the list of all the documents that are to be prepared and delivery during the process will be mentioned here in this section.

6.0 Test environment: the clear details of the environment that is about to be used for testing that application will be mentioned here in this section.

7.0 Resource planning: who has to do what will be planned mentioned here in this section.

05 June

(6 months)

Plan – (Knowledge transfer session – 15 to 20 days) – June

KT start 06 June – 26 June end date (30 June)

July -

05 December 2021(First build release)

8.0 Scheduling: the starting date and ending date of each and every task will be plan and mentioned here in this section.

9.0 Staff & training: how much staff need to be recruited and what kind of training need to be provide will be mentioned here in this section.

10.0 Risk and contingency: the list of all the potential risk that may occur during the process and corresponding solution plan will be mentioned here in this section.

Ex: i. employee may leave a organization middle of the project

ii. unable to deliver the project within the deadlines

iii. Customer may impose the dead lines

iv. unable to test all the features within the given time

Contingency:

- ➢ employee need to be maintain on bench
- ➢ what not be tested should be plan in case of imposed dead line
- ➢ Proper plan ensure
- ➢ Priority base execution

11.0 Assumption: the list of all the assumption that needs to be assumption by a test engineer will be listed out here in this section.

12.0 Approval information: who has approved this document and when it is approved will be mention here this status.

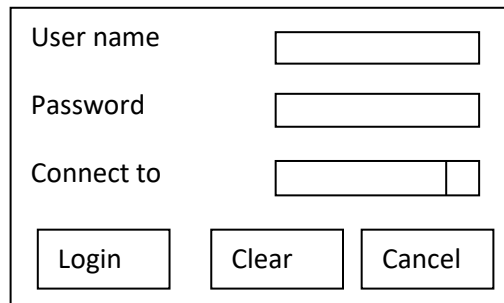**2. TEST DEVELOPMENT: (Business** analyst) (test case development)

Req document

| HLI(High level Info) |
| LLI(Low level Info) | → Use case
| Screen shot& Snap shots |
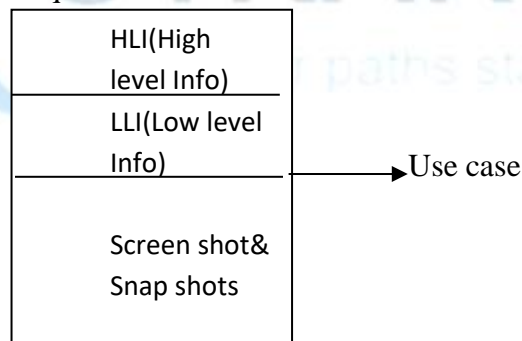
Use case: It describes the functionality of certain feature of an application in terms of actors (user), actions (perform operations over applications) and responses.

- Actors (normal user(end), admin user)
- Actions
- Responses

Snap shot:

User name	☐
Password	☐
Connect to	☐
Login	Clear	Cancel

OE user

invoke the login page of application (Action)

Login page should open (response)

     When
- **OE user Enter the username in username field and enter the password in password field.**
- **Select the preferred database from connect to field.**
- **Click on Login Button**

     Then

- **Navigate to Applications Home Page.**

Functional requirements (Explicit)

➢ Login screen should contain user name, password, connect to fields, login, clear and cancel buttons.

➢ Connect to field is not a mandatory field but it should allow the user to select the database option if required

➢ Upon entering user name, password and clicking on login button the corresponding page must be displayed

➢ Upon entering some information into any of the field & click on clear button all the fields must be clear and the cursor must be available in the user name field.

➢ Upon clicking on cancel button login screen must be closed

Special requirements or validations or Business rules (Implicit)

➢ Upon invoking the application login and clear buttons must be disabled

➢ Cancel button must be always enabled

➢ Upon entering some information into any of the field clear button must be enabled

➢ Upon entering some information into both username & password field's login button must be enabled.

➢ Tabbing order must be user name, password, connect to, login, clear and cancel

Use case template:

➢ Name of the use case

➢ Brief description of the use case

➢ Actors involved

➢ Pre-conditions

➢ Post conditions

➢ Flow of events

Use case document:

Name of the use case: login

Brief description of the use case: this use case describes the functionality of all the features present in the login screen.

Actors involved: normal user & admin user special requirements

There are 2 types of special requirements

➢ Implicitly requirements
➢ Explicitly requirements

Implicitly requirements: the requirements that are analyzed by the business analyst & his team which will add some value to the application & will not affect only of the customer requirement.

Explicit requirements: the requirement that are explicitly given by the customer are known as explicit requirements

Explicit requirements:

➢   Upon invoking the application login and clear buttons must be disabled

➢   Cancel button must be always enabled

➢   Upon entering some information into any one of the field clear button must be enabled

➢   Upon entering some information into any of the field & click on clear button all the fields must be clear and the cursor must be available in user name field

➢   Upon click on cancel button login screen must be closed.

Implicit requirements:

➢   Upon invoking the application, the cursor must available in the user name field

➢   Upon entering invalid username& password and clicking on login button the

Following error message must be displayed

➢   **"Invalid username please try again"**

➢   Upon entering valid username & invalid password & clicking on login button the following error message must be displayed

➢   **"Invalid password please try again"**

➢   Upon entering invalid username & invalid password and clicking on login button the following error message must be displayed

➢   **"Invalid username & password please try again"**

Pre-conditions: Login screen must be available

Post conditions: either home page or admin page for valid users and error messages for invalid users.

Flow of events Main flow:

Action	Response
Actor invokes the application	Application displays the login screen with the following fields: user name, password, connect to, login, clear & cancel
Actor enter valid user name, valid password and clicks on login button	Authenticates, applications displays either home page or admin page depending upon the actor entered.
Actor enters valid user name, valid password, selects a data base option & clicks on login button	Authenticates, application displays either home page or admin page depending upon action entered with the mentioned database connection
Actors enters invalid user name, valid password and clicks on login button	Go to alternative flow table 1
Actor enter valid user name invalid password and clicks on login button	Go to alternative flow table 2
Actor enters invalid user name, invalid password and clicks on login button	Go to alternative flow table 3
Actor enters same information into any of fields and clicks on clear button	Go to alternative flow table 4
Actor clicks on cancel button	Go to alternative flow table 5

Alternative flow table 1 (Invalid user name):

Action	Response
Actors enters invalid user name, valid password and clicks on login button	Authenticates application displays the following error message "invalid user name please try again"

Alternative flow table 2 (Invalid password):

Action	Response
Actors enters invalid user name, invalid password and clicks on login button	Authenticates application displays the following error message "invalid password please try again"

Alternative flow table 3 (Invalid user name &password):

Action	Response
Actors enters invalid user name& password and clicks on login button	Authenticates application displays the following error message

	"invalid user name &password please try again"

Alternative flow table 4 (clear click):

Action	Response
Actors enters same information into any of the fields and clicks on clear button	Application clears all the fields &makes the cursor available in the user name field.

Alternative flow table 5 (cancel click):

Action	Response
Actors clicks on the clicks on the cancel button	Login screen is closed

The guide lines followed by a test engineers soon after the use case document is received:

1) Identify the module to which the use case belongs to

> ➢ Security module

2) Identify the functionality of use case with respect to the total functionality

> ➢ authentication

3) Identify the actors involved in the use case

> ➢ normal users, admin user

4) Identify the inputs required for testing

> ➢ valid & invalid user names and passwords

5) Identify whether the use case is linked with any of the use cases

> ➢ it is linked with home page &admin page use cases

6) Identify the pre-condition

> ➢ login screen must be available

7) Identify the post condition

> ➢ either home page or admin page for valid for user and error message for invalid

> Users

8) Identify the function point & prepare the function point document

9) Understand the main flow of the application

10) Understand the alternative flow of the application

11) Understand the special requirements

12) Document the test case for the main flow

13) Document the test cases for alternative flow

14) Document the test cases for special requirements

15) Prepare the cross-reference matrix or trace ability matrix.

Functional point:

The point where the user performs some action in the application can be considered as functional point.

Testing process related document:
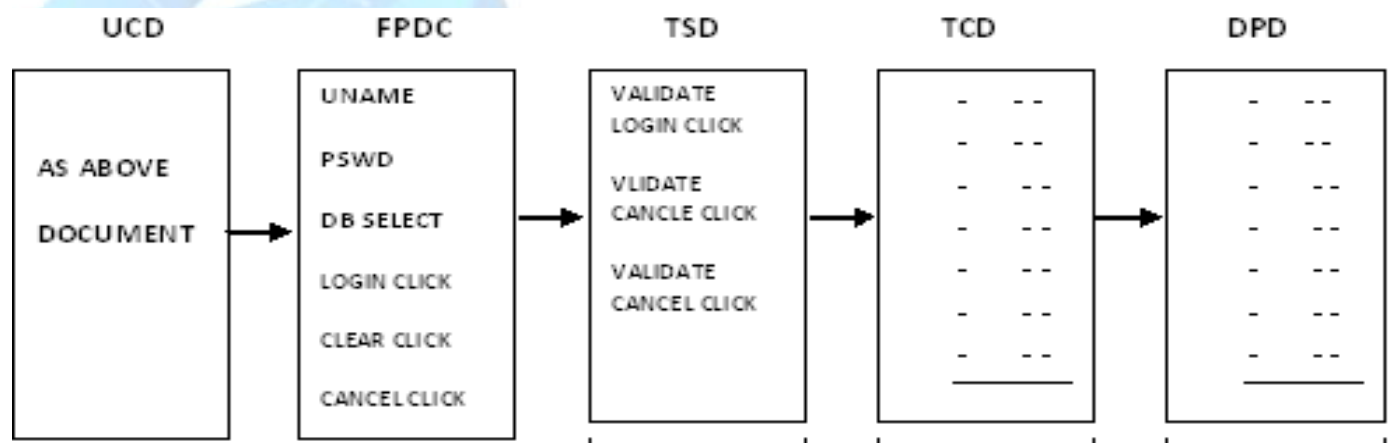
Traceability matrix:

Traceability matrix is a document which contains table of information used for traced back reference in any kind of confusion or questionable situation

Complete Traceability matrix (CTM):

UC_ID	FP_ID	TS_ID	TC_ID	DP_ID
EC-37	8	4	EC-39	1
32.0	24	8	52	2

| 82.4 | 52 | v28 | 100 | 3 |
| 100.0 | 70 | 35 | 250 | 4 |

Requirement traceability matrix:

T CDID	Req ID
1	1.0
2	1.0
3	1.1
4	1.1
5	1.2

Defect traceability matrix:

Defect ID

D Id	Tc Id
1	38
2	34
3	52
4	70
5	90

Types of test cases:

Test cases are broadly divided into 3 types

1. GUI Test cases

2. Functional Test cases

3. Non-functional test cases

Functional test cases are further divided into 2 types

I. Positive test cases

II. Negative test cases

Guidelines for writing the GUI test cases:-

➢ Check for the availability of all objects.

> ➢ Check for the consistency of the objects.

> ➢ Check for the alignment of objects in case of customer requirement only.

> ➢ Check for the spellings and grammar.

> ➢ Apart from the above guidelines any other idea we get with which we can test something in the application just by look and feel without doing any action, and then all those ideas also can be considered as GUI test cases.

Guidelines for writing the positive test cases:-

> ➢ A test engineer should have positive mind set.

> ➢ He should consider the positive flow of the application.

> ➢ He should use only valid inputs from the point of functionality

Guideline for writing negative test cases:-

> ➢ Test engineer should have negative mend set.

> ➢ He should consider the negative flow of application.

> ➢ He should use at least one invalid input for each set of data.

Project name	
Module	
Author	

Test case Template:

Req._ID	TC_ID	Category	Prerequisite	Description/test steps	Expected Value	Test Data	Actual Value	Result	Build No	Priority
1	T_1.1	Normal	Login Form should be available in local machine	User click on to Login Form	Validate Login form should open					Minor
1	T_1.1	Normal	Login Form should be available in local machine	Check all fields should be present on login form						

				1) Username name 2) Password 3) Connect To 4) Login 5) Clear 6) cancel						

Functionality:

1) Minimum and Maximum lengths should be set for all the text boxes
2) Password should be displayed in masked format(****) rather than showing actual text format
3) Login credentials in UPPER case should not be treated as invalid
4) Validation message should be shown when special characters are entered in the username field, or when invalid username and/or password is entered or the fields are left blank
5) Reset button should clear data from all the text boxes in the form
6) Login credentials, especially password, should be stored in database in encrypted format

Security:

1. When logged in user copy URL and paste in new browser window, it should redirect to Login page
2. Users should not be allowed to copy and paste Password from text box
3. Notification email for multiple device login - if user login from unusual device/machine
4. Entering Login credentials using virtual keyboard should be provided for banking application
5. After 3 or 5 unsuccessful attempts of login, user login credentials should get locked for specific period e.g. 24 hours
6. SSL certificate should be implemented/installed for Secured Website
7. SQL injection attacks & XSS should be verified for login
8. Two-way authentication through OTP on mobile/email should be tested for banking application

Session:

1. After logout if user clicks on back button user should not be able to login within same session, it should redirect to login page
2. If user logged in on multiple devices and Logout from one device then it should Logout from all platform/devices
3. Maximum Session out time should be set for Secured website

Browser:

1. If Browser cookies are cleared and user tries to login, the system should ask for credentials again
2. 'Remember Form Data' setting of the browser should not remember the password
3. Validate the login functionality when browser cookies are turned OFF

Test Case vs. Test Scenario: What's the Difference?

What is the Test Case?

A **TEST CASE** is a set of actions executed to verify a particular feature or functionality of your software application. A Test Case contains test steps, test data, precondition, post condition developed for specific test scenario to verify any requirement. The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the customer.

What is a Test Scenario?

A Test Scenario is defined as any functionality that can be tested. It is a collective set of test cases which helps the testing team to determine the positive and negative characteristics of the project.

Test Scenario gives a high-level idea of what we need to test.

Example of Test Scenario

For an ecommerce Application, a few test scenarios would be

Test Scenario 1: Check the Search Functionality

Test Scenario 2: Check the Payments Functionality

Test Scenario 3: Check the Login Functionality

KEY DIFFERENCE

- Test Case is a set of actions executed to verify particular features or functionality whereas Test Scenario is any functionality that can be tested.
- Test Case is mostly derived from test scenarios while Test Scenarios are derived from test artifacts like BRS and SRS.
- Test Case helps in exhaustive testing of an application whereas Test Scenario helps in an agile way of testing the end to end functionality.
- Test Cases are focused on what to test and how to test while Test Scenario is more focused on what to test.
- Test Cases are low-level actions whereas Test Scenarios are high-level actions.
- Test Case requires more resources and time for test execution while Test Scenario require fewer resources and time for test execution.
- Test Case includes test steps, data, and expected results for testing whereas Test Scenario includes an end to end functionality to be tested.

Example of Test Cases

Test cases for the **Test Scenario:** "Check the Login Functionality" would be

1. Check system behavior when **valid email id and password** is entered.
2. Check system behavior when **invalid email id and valid password** is entered.

3. Check system behavior when **valid email id and invalid password** is entered.
4. Check system behavior when **invalid email id and invalid password** is entered.
5. Check system behavior when email id and password are left blank and Sign in entered.
6. Check Forgot your password is working as expected
7. Check system behavior when valid/invalid phone number and password is entered.
8. Check system behavior when "Keep me signed" is checked

Why do we write Test Cases?

Here, are some important reasons to create a Test Case-

- Test cases help to verify conformance to applicable standards, guidelines and customer requirements
- Helps you to validate expectations and customer requirements
- Increased control, logic, and data flow coverage
- You can simulate 'real' end user scenarios
- Exposes errors or defects
- When test cases are written for test execution, the test engineer's work will be organized better and simplified

Why do we write Test Scenario?

Here, are important reasons to create a Test Scenario:

- The main reason to write a test scenario is to verify the complete functionality of the software application
- It also helps you to ensure that the business processes and flows are as per the functional requirements
- Test Scenarios can be approved by various stakeholders like Business Analyst, Developers, and Customers to ensure the Application under Test is thoroughly tested. It ensures that the software is working for the most common use cases.
- They serve as a quick tool to determine the testing work effort and accordingly create a proposal for the client or organize the workforce.
- They help determine the most critical end-to-end transactions or the real use of the software applications.
- Once these Test Scenarios are finalized, test cases can be easily derived from the Test Scenarios.

Test case vs. Test scenario

Here, are significant differences between Test scenario and a Test Case

Test Scenario	Test Case
A test scenario contains high-level documentation which describes an end to end functionality to be tested.	Test cases contain definite test steps, data, and expected results for testing all the features of an application.

It focuses on more "what to test" **than** "how to test".	A complete emphasis on "what to test" **and** "how to test."
Test scenarios are a one-liner. So, there is always the possibility of ambiguity during the testing.	Test cases have defined a step, pre-requisites, expected result, etc. Therefore, there is no ambiguity in this process.
Test scenarios are derived from test artifacts like BRS, SRS, etc.	Test case is mostly derived from test scenarios. Multiple Test case can be derived from a single Test Scenario
It helps in an agile way of testing the end to end functionality	It helps in exhaustive testing of an application
Test scenarios are high-level actions.	Test cases are low-level actions.
Comparatively less time and resources are required for creating & testing using scenarios.	More resources are needed for documentation and execution of test cases.

3. Test Execution:

In this phase the test engineer will do the following

1. They will perform the action as it is described in the description column

2. They will observe the actual behavior of the application.

3. They will write the observed value in the actual value column.

4. Result Analysis: In this phase the test engineer will compare the expected value with the actual value if both are matching they will decide the result is pass otherwise fail.

Note: If at all the test case is not executed because of any reason then the test engineer will keep block in the result column of test case template.

Result status

1) Pass
2) Fail
3) WIP (Work in Progress)

4) Blocked
5) Unexecuted

Work Flow for Jira Scrum Board

1) To Do
2) In Progress
3) Code Review
4) To Test
5) Test In Progress
6) In Acceptance (PO)
7) Hold
8) Done

5. Bug tracking: Bug tracking is a process of identifying, isolating and managing the defects.

Defect profile Template:

Test case id: The test case id based on which defect is found will be mentioned here in this section.

Defect id:

Defect Id	Test case Id	Issue discription	reproducable steps	Defect				defect		Defect status	Retified/fix		date closure
				By	Date	Build no	version no	Civiority	priority		by	date	

The sequence of defect numbers will be mentioned here in this section.

Issue description:

What exactly the defect is will be clearly described in this section.

Re-producible steps:

The list of a all the steps followed by the test engineer to identify the defects will be listed out here in this section

Reported by:

The name of the test engineer who has identified will be mentioned here in this section.

Detected date:

The date on which the defect is identified will be mentioned here in this section.

Detected build no:

The build number in which the defect is identified will be mentioned here in this section.

Detected version: (change management team given version no)

The version number in which defect is identified will be mentioned here in this section.

**Severity**: Severity describes the **seriousness** of the defect.

Severity is classified into 4 types

Severity   =   Priority

1. Fatal   =   Blocker (Highest)

2. Major =   Critical (High)

3. Minor =   Major (Medium)

4. Suggestions = Minor (Low)

5.   Lowest

Fatal defects: If at all the problems are related to the navigational blocks or un-availability of main functionality then such type of defects are treated as fatal defects.

Ex:
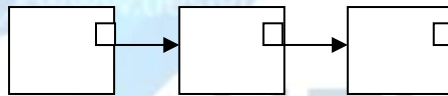
Value 1	[ ]
Value 2	[ ]
Sum =	[ ]

Major defects:

If at all the problems are related to working of the main functionality then such type of defects are treated as major defects
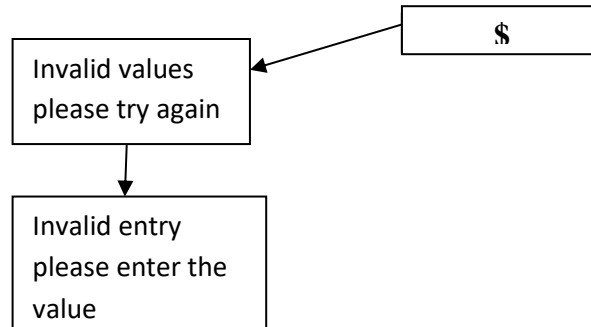
Value 1	[ ]
Value 2	[ ]
Value3	[ ]

Minor defects:

If at all the problems are related to the look and feel of the application then such type of problems is treated as minor defects.

Suggestions:

If at all the problems are related to the value of the application (user friendliness) such type of problems are treated as suggestions.

```
                                    ┌──────────────┐
                                    │      $       │
┌──────────────────┐  ◄─────────────└──────────────┘
│ Invalid values   │
│ please try again │
└──────────────────┘
          │
          ▼
┌──────────────────┐
│ Invalid entry    │
│ please enter the │
│ value            │
└──────────────────┘
```

Priority: Priority describes the sequence in which the defects need to be rectified. Priority classified into 4 types

1.   Blocker (Highest)
2.   Critical (High)
3.   Major (Medium)
4.   Minor (Low) – (Lowest)

     Usually fatal defects critical priority, major defects will be given high priority, minor defects   will be given medium priority and suggestions will be given as low priority.

But depending upon the situation priority will be changed by the development lead.

  Sometimes highest severity defects will be given low priority vice versa….

Case 1: least severity highest priority whenever the customer visits all the look and feel defects given highest priority.
Case 2: highest severity least priority

     Whenever some part of module or application released to testing department as it is under construction the tester will usually raise it is fatal defect but the development lead treat it as least priority.

High priority and low severity

1) **Logo of organization in home page is wrongly posted.**
2) **2 % extra interest on festival time after certain time still the message will be displayed on their portal.**

   High priority and high severity

1) **In ATM machine same bank ATM (Free) and other bank ATM having same charges per transaction.**
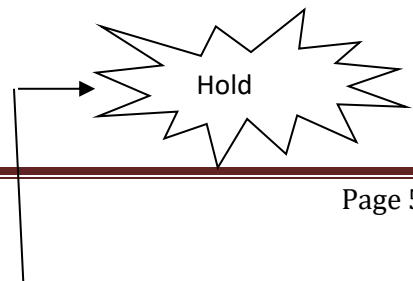2) **Valid credentials are not working properly.**

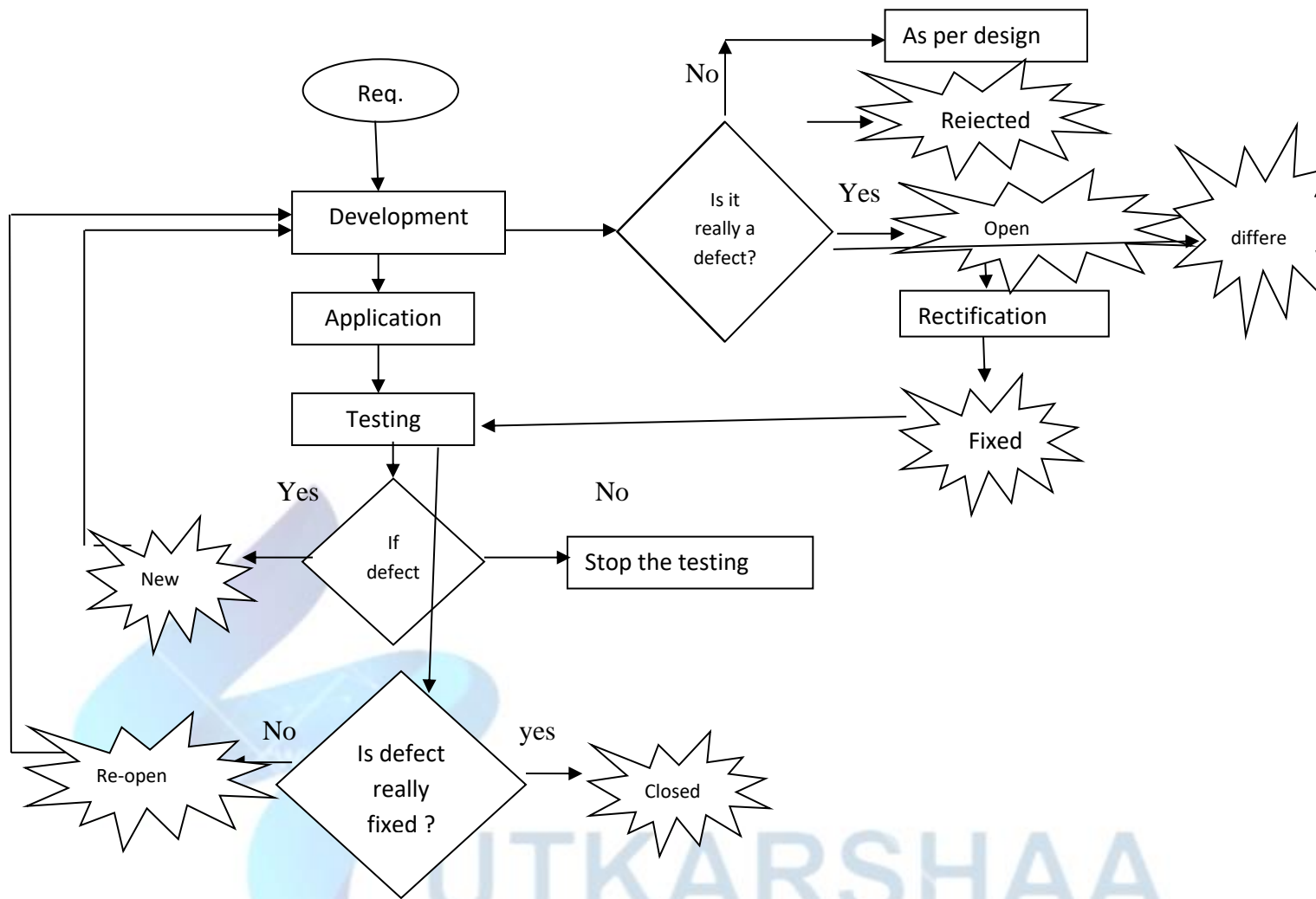**Low Severity and low priority**

1) **Spelling mistake inside the application blogs or messages.**

Example icici (iccic)

Error – white box tester/ developer – while development the developer execute their code.

Bug/Defect life cycle: black box tester.

Hold

New :

   Whenever the defect is newly identified for the first time then by the tester he will set the status as new

Open :

   Whenever the developer accepts the defect then he will set the status as open.

Differed:

   Whenever the developer accepts the defect and wants to rectify it later stages then he will set the status as differed.

Fixed: Once the defect is rectified then the developer will set the status as fixed or rectified

Re open and closed:-

Once the next build is released the tester will check whether the defect is really rectified then he will set the status as closed otherwise re open.

Hold :

Whenever the developer is confuse to accept or reject then he will set the status as hold

When the defect is in hold status they will be a meeting on the defect and if it is decided as a defect then the developer s well set the status as open otherwise tester will be closed

Rejected:

Whenever the developer feels that is not at all defect then they will set the status as rejected.

Whenever the defect is rejected then the test engineer will once again check it if at all they also feel it is not a defect then they will set the status as close or re-opened

As per design :(rare case)

Whenever developer feels the testers are not aware of latest requirement then they will set the status as per design.
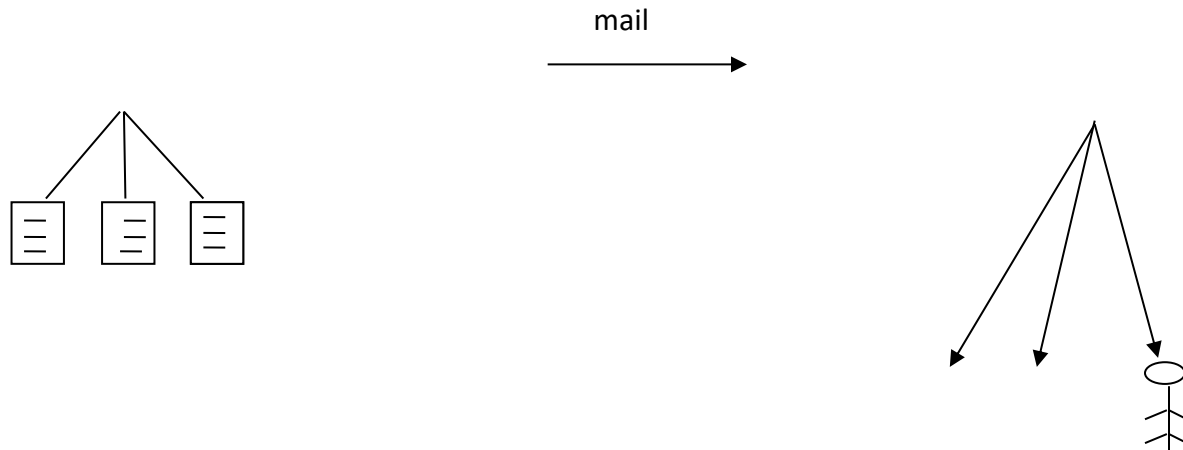
Whenever the defect is as per design the tester swill once again check it by going through the latest requirements if at all they also feel it is as per design then they will set the status as closed otherwise re opened.

6. **Classical bug reporting process**:

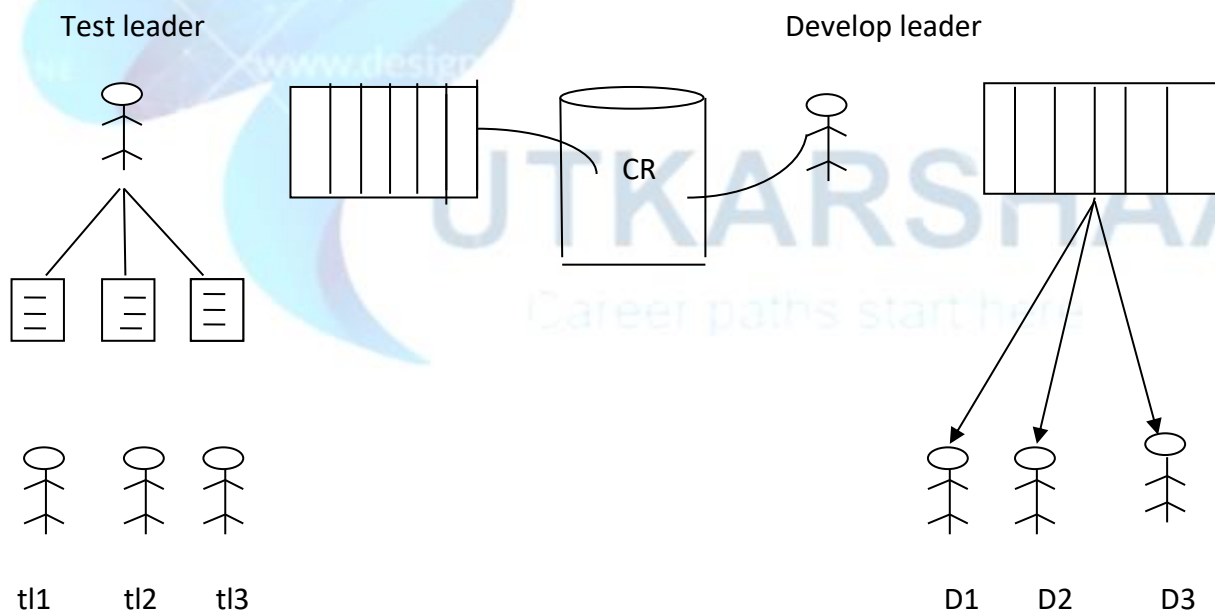Test leader   Develop leader

mail

Drawbacks: i. time consuming   ii. Redundancy

iii. No transparency   iv. No security

Common repository oriented bug tracking process:

This is not send mail it is only security purpose, only store one (server) box and take only authority developer.

Test leader   Develop leader

tl1   tl2   tl3   D1   D2   D3

Draw backs: i. time consuming   ii. Redundancy iii. No security

Bug tracking tool oriented bug reporting process: every software user used some tools

Ex: I tracer, He tracer it is a simple software. It is can't access people it is only used for bug tracking. In house tools are using only house purpose.

This is the latest process all company's is use now a days.
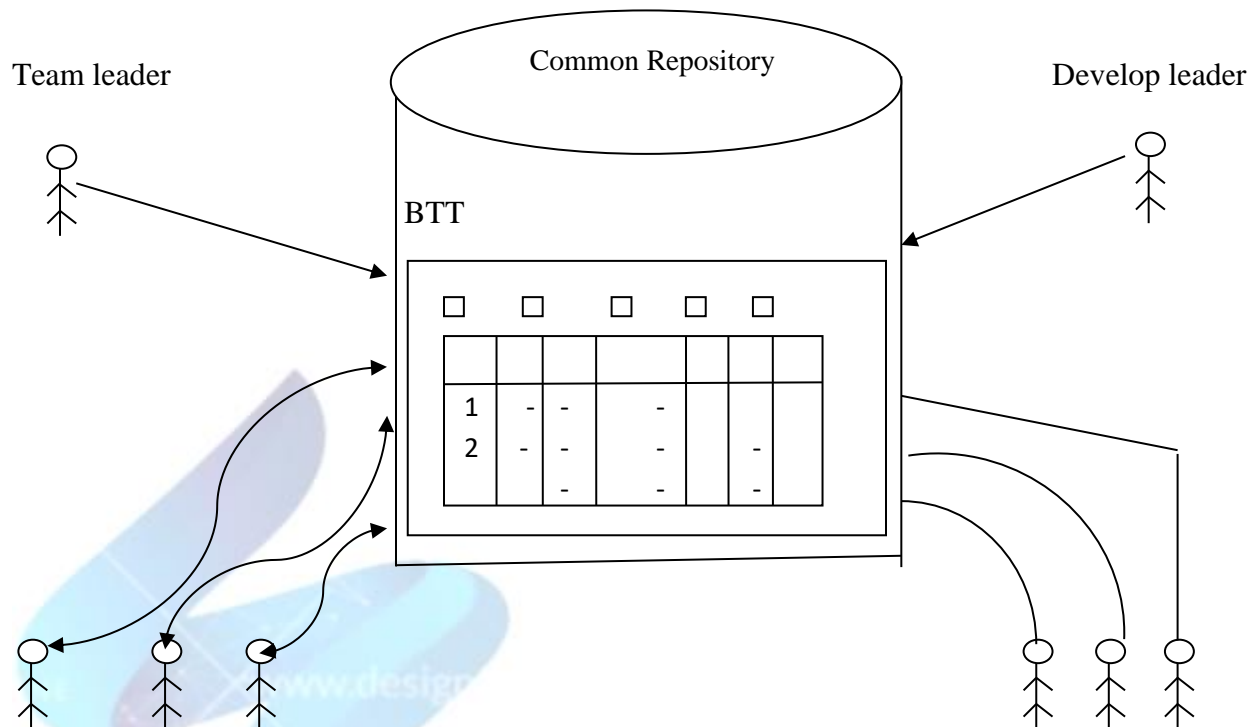
Draw backs: i. no security ii. No consuming iii. Full transaction

Note: quality center is a bug tracking tool, Bug tracking tool is a software application which can be access only by the authorize users they onetime do all the activities comfortably relative to bug tracking and reporting.

Test closure activity: Test leader will prepare one test summery; this is the final activity for the testing process usually done by the test summery report, which usually contains following information

> ➢ No. of cycles(builds) of execution

> ➢ Duration of each cycle

> ➢ No. of defects found in the each cycle

> ➢ No. of test cases executed in each cycle and etc…,