Batch- 12 Feb 22

Date-14/02/22

Manual Testing

Manual Part I-

- **Process/ Methodologies/ Model**
  1. SDLC process
  2. Waterfall model
  3. V model
  4. **Agile Model** – 90-95%
- **Different Testing**
  1. Sanity Testing / Smoke Testing
  2. System & Functional Testing- 17 to 18
  3. Retesting & Regression Testing
  4. User acceptance testing – 2
  5. Testing Terminologies- 3

Manual Part II

Documentation Part- Test Policy, Test Strategy, Test Methodology, Test plan, TCD, TCE, DR, TSR,TCR

Project Management Tool- JIRA / HPALM

Database Testing

SQL- Oracal 11g – commands

API Testing / Webservice testing

SOAP/REST- SoapUI tool

REST- Postman Tool

UNIX- Commands

Live Project

1. Investment Banking Domain
2. Telecom Domain

Syllabus Completion Plan

Manual Part 1- 5 week

Database Testing- 1 & ½ week

Manual Part 2- 1 & ½ week

JIRA/HPALM-1week

API Testing – 2 week

Unix- 3 days

Live Project – 2 Weeks

Class- 3month – 12 weeks – 12 week

Extend- 1 & 2 week

Database – No of Tables- no of rows & column

Register

FN LN DOB MB Email ID

Login

UN PW

Ex. Frontend

FB

Register

FN

LN

DOB

MB

Email ID

Submit – successful register

Create PW

UN-Mangesh

PW-Mangesh123

Submit – create UN & PW successfully

FB

Login

UN-Mangesh

PW-Mangesh123

Date- 15/02/22

Project Size/ Team (16-18)

Projects

Project / Main team- Application new feature/ module/ functionality add/implement

Support / Maintenance Team- Existing application issue/defect/end user quires/Ticket etc.

Team size-

- I have **worked in project team**
- Team size
- Project team size (16-18 peoples)
- Delivery manager/ solution master (1)- project against delivery to client
- Business analyst/ Product owner (1)- BA always communication with client for the requirements
- Project manager/ Scrum master(1)- Project work / task assign & work progress monitor

- Design / solution / system architecture (1)- application against design (developer more 10/12 year)
- Developers (8-10)- coding against the application
- Testers (3-4)- test against application

Support team/ maintenance team (5-7 people)

- Project manager / support manager (1)- project tickets assign & monitor work progress
- Developers (3-4)- coding against the tickets/issue
- Tester (1-2)- resolved tickets against test

- In your organization- project team & support team

- Project & support team decided – client

- Ex. HSBC project (client)- project team= zensar & support team = wipro (1cr)

- Ex. Macdonled project (client)= project team & support team = TCS (ST-3cr)

Interview Questions

1. Team size? Roles?
2. What is team size in the last project?
3. What is the size of the developer in your project?
4. What is the size of the tester in your project?
5. In which team you have worked?

Software

Collection of program

Set of instructions

Logic & control

It is a set of – program, procedures & services associated with that operating system (which make it run)

Software is collection of specialized program which takes user input & generate desired output

Testing

Testing is a process to checking whether given application is generating desired output

It is process where the application produce desired output = for the given input

i.e. finding how something work

Software Testing

It is process to check **completeness & correctness** of the application w.r.t. **client requirement**

Verification & validation of the software is called as testing

Verification- means whether the software correctly implemented or not

Validation – means the implemented software meets the customer requirement or not

Software development flow

Actual Happy flow of the application

// Shree Ganesh Cloth Shop//

Men **Women**

Men- Shirt Pant

Shirt- Brands, Size, Half, Full, Price

Pant- Brands, Size, Type, Price

Women- T-shirt Sari

T-shirt- Brands, Size, Price

Sari- Brands, Price

//online shopping application for above requirements

Company- BA

Technical Team- PM, Solution team, dev team, test team

All the team work parallel

PM-

Takes assurance of development & test team about project & also monitor

Solution Team- DA

Explain the way to implementation & duration of project

Dev Team-

Complete development- WBT- Unit testing & integration testing (Register + Login)

One way scenario hit – positive way ne check

Test Team-

- **Check, whether the developed application is accurate or not**
- **Check, whether the code is according to the client requirements or not**

Sanity Testing

Positive way scenario hit – application stable or unstable

Smoke Testing

Advance version of sanity testing

Positive way scenario hit- application stable or unstable

Finding root cause of defect

System & Functional Testing (17) (BBT)

Step by step check from start to end

Retesting

If found defect – assign – developer – developer correct / fix defect- again send to tester – tester perform the retesting

Regression Testing

Code change – other module / functionality effect – regression testing

Sign Off

UAT testing / client testing – 2

Sign Off

Production

Release / DM

End User/ Client

Software development process

Client (requirement)

BA (collect all the requirements)

Developer (design & develop as per the req.)

Testing

Final production

Client

- Client wants some application
- So client comes with the own idea/ requirement

BA

- BA is responsible person to collect the requirements
- BA make a document & send to the technical team

Technical Team (PM,ST,DT,TT)

- Once we got the document from the BA then technical team understand the exact requirements of the client, analysis, assign to the experienced person

Dev team

- Dev team understand the client requirements
- According to the requirements developer starts working on it / developing the code

Test team

- Test team understand the client requirements to prepare test case design
- Once the development is done, developed application is sent to test team
- Tester tests developed application with every possible way
- Positive testing
  Provides valid data & check how application behaves for valid data
  Ex. mobile number text box/field – valid data- enter only 10 digit

- Negative testing
  Provides invalid data & checks how application behaves for invalid data
  Ex. mobile number text box-
  Invalid data- less than 10 digit mobile number
  Invalid data- more than 10 digit mobile number
  Invalid data- combination of char, digit, special symbol. Decimal

Final Product

After all the testing, final product is ready

Positive Testing

Flipkart

Login page

UN- Mangesh

PW- Mangesh123

Negative Testing

Incorrect UN & PW

UN- Mangesh123

PW- Mangesh

Click on the login button

It display validation message

“please enter valid UN & PW ” in red color

Syllabus of the Manual Testing

Project team

Definition

Software

Testing

Software Testing

Happy flow software development – in – out

Software development process

Software quality assurance

Customer (Nilesh)--------KIA SELTOS--------KIA (Mangesh)

Customer (Nilesh)-----Shree ganesh online application-----Software company(BA-Mangesh)

- Involved – BA & client / customer
- It is communication happen between the Client & BA
- Check the quality or to measure & monitor the process of development & testing

SQA depends on **different factor**

1. **To meet client Requirements**

- Identify client requirements & purpose
- Which type of application client wants
- Which is the domain of client requirements
  Ex. telecom, banking, healthcare, insurance, e-commerce…..etc
- What is the purpose of client requirements

2. **To meet client Expectation**

- Privacy- includes security
  For ex. banking domain application
  User data gather- aadhar card / pancard
  Very sensitive
  Customer wants privacy to all the user data
  - Big transaction

- Performance- includes speed – with & without load condition
  Application balance load properly
  It should work properly under heavy load
  Ex. FB, Instagram, Google meet

3. **Cost of product**

- MNC company – per hour cost client have to pay
- Depends type of project
  Normal project- required less resources
  Critical/complex project- required more resources
- Resource allocation
- Size of the project
- Time to complete – duration of the project
  Ex. BA- client- 1 year for project completion

Client wants- 6 month – BA- team / resources / size project increase – cost project increase

4. **Timely deliver**

- At the time of information gathering & documentation- duration of project or time to complete the project is decided
- Company exceeds the deadline / delivery time, then company have to pay penalty, that penalty is called as escalation
- Penalty/ escalation – reduction in fund (project cost-100cr- 2 years- 2 & ½ years = penalty- 100 reduction fund)

5. **Maintenance** / support / service (duration- Client & BA)
   After delivery of the project, if any problem occurs / technical difficulty occurs in such case company has to fix it without any cost
   Provides
   KPO- knowledge process outsourcing – technical team
   BPO-Business process outsourcing – non technical team

   Ex. customer care call center
   BPO-
   KPO-

Project having two categories

1. Critical project
2. Normal project

The ratio of developer & tester – as per the project type – resource allocation

Critical project

Resource requirement ratio =2 (developer):1(tester)

Ex. **banking** – 2 developer & 1 tester

4 developer – 2 tester

Normal project

Resource requirements ratio 3(developer):1(tester)

Ex. insurance domain / **gaming project**

3 developer: 1 tester

9 developer: 3 tester

NASA project

Developer 1: tester 2/3/4/6/7

Process / Methodology / model

- Process / Methodology / model a **way to develop & test** the software or application
  1. SDLC (Basic model)
  2. Fish model
  3. Waterfall model
  4. V model
  5. **Agile Model (90/95%)**
- **Who will decide the process?**
- **Client decide**

Software Development Life Cycle

Two Types

1. **LCD- life cycle development**
2. **LCT- life cycle testing**

SDLC-

It is a process used by the **software company/organization/industry** to **design, develop, & test** the **high quality software**

In the SDLC there are **different stages**

1. **Information gathering / requirement gathering-** BA – collect the requirement from client – prepared BRS document
2. **Analysis-** BA analysis – BA prepared – SRS
3. **Design-** Designer – prepare the HLD & LLD
4. **Coding-** developer- developer will do coding LLD
5. **Testing-** tester will TCD & TCE
6. Support/ **Maintenance**

Information gathering / requirement gathering

- In Information gathering stage BA is responsible person to gather the requirement
- Information gathering it is nothing but requirement gathering
- In this stage BA will intact with client & collect the requirement related to client project/ business
- In this stage BA will prepared one document – BRS (Business requirement specification)
- BRS defines **project / business related requirement** of the application/software
- Ex. paytm- end user use – **registration**, login, recharge / rent / gas / send money / flight ticket……etc.
- BRS documents we don't get (designer, developer, tester)
- BA is taking requirements from client & prepared BRS document & BA simply its acts as bridge between client & technical team

Client **BA** Technical Team

Analysis

- In the analysis also BA is responsible
- BA again will communicate to the client & collect requirements from client, against these requirement related to functionality of the application
- BA again prepared a document SRS (software requirements specification)
- SRF- FRS(functional requirements document) & CRS (Client requirements document)
- SRS defines software / application / **functional requirements to be developed** & system (module/hole project) requirements that will be used
- Ex. Paytm – Recharge module- **Radio button- mobile no- circle / operator- amount- check box- submit button**
- **SRS will contains**
  1. Functional requirements (project – multiples requirements)
  2. Functional flow diagram – (step by step process)
  3. Use cases (specific requirements- 1 req.- registration)
     Description- details about the requirements
     Acceptance- does & don't about requirements
  4. Screenshot / snapshot / dummy model / prototype / wireframe

Functional flow diagram – (step by step process)

Registration

Login

2FA

Dashboard

Select Share

Buy share

- FFD- represents step by step stages of application
- Represents relation between the task
- Dependencies between the task

Functional requirements (project – multiples requirements)

FR- meeting the attributes which are required to complete specific functionality / task

For example- Register on banking app

FN- should accept characters only (upper case/ lower case), length- (48)| don't allow- digit, decimal, _, space, special symbol

LN- should accept characters only (upper case/ lower case), length- (64)| don't allow- digit, decimal, _, space, special symbol

DOB- should accept only integer/number, digit, format- DD-MM-YY / DD/MM/YY / MM/DD/YY | don't allow- decimal, _, space, special symbol, char

Email Id- should accept character, number, _, decimal, special symbols | don't allow- space

Mob. No- should accept digit (10 digit with country code)| don't allow char, space,_, decimal, special symbol….etc.

Submit Button- check button should be clickable

This is nothing but functional requirements

Use Case- (multiple use cases) / User Story

- Specific requirement / 1 requirement
  Description – Details about the requirements
  Acceptance criteria- Does & don't about requirements

Ex.

Description

As a user **I want** register kite application **so that** I can move to the password create page

Acceptance Criteria

FN- should accept characters only (upper case/ lower case), length- (48)| don't allow- digit, decimal, _, space, special symbol

LN- should accept characters only (upper case/ lower case), length- (64)| don't allow- digit, decimal, _, space, special symbol

DOB- should accept only integer/number, digit, format- DD-MM-YY / DD/MM/YY / MM/DD/YY | don't allow- decimal, _, space, special symbol, char

Email Id- should accept character, number, _, decimal, special symbols | don't allow- space

Mob. No- should accept digit (10 digit with country code)| don't allow char, space,_, decimal, special symbol….etc.

Submit Button- check button should be clickable

Use Case – Test Scenarios – Test case design

Use case- help to identify test cases

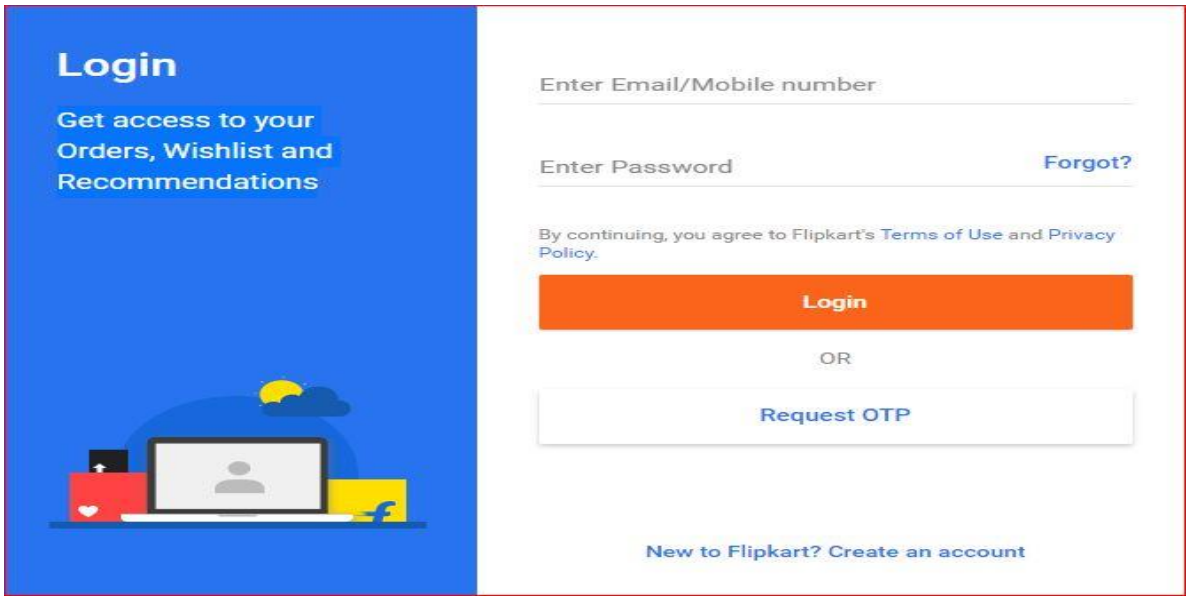
Use case- Combination= Input + Process + Output

Task- Use cases for online shopping

Snapshot

- **Application without functionality**
- Snapshot is format/ review/ prototype / screenshot / dummy module
- Snapshot provides idea to developer how software look like

- Snapshot is created by BA
- Uses Irise software for snapshot creation
- Irise - 8.11 version
- Visualization of functionality before development of product



When BA will **completed SRS document**. Then BA will **sent these document to developer & tester.**

Sent this document **throw mail** (outlook)

Developer & tester team – **understand the document**

If we have **doubt in SRS document**- then **developer & tester communicate to BA** (session)

What is difference between BRS & SRS

BRS	SRS
Business Requirement specification	Software Requirement specification
BRS document- it consist of complete scope of the project, project requirements	SRS document- all functional & nonfunctional requirements (Operating system/ browser)
BA- prepare the BRS	BA- prepare the SRS
From client BA collects all requirements and prepare BRS document	SRS is derived from BRS
Gathering client requirements	Gathering software & technical requirements

Uses cases are not present in BRS	Uses cases are present in SRS
Overall requirement	Details requirement
Ex. Investment banking Kite Register Login 2FA Dashboard Watchlist Order Position Fund Profile	Ex. functional requirments Register FN LN Pan Aadhar Mob.No DOB Email id Login UN PW Login Button Forgot PW

Design

- When BA sent SRS documents to designer
- Design stage – designer are working
- Designer will be prepared HLD & LLD
  HLD-
  Men Women

  LLD-
  Shirt- brands, half, size, price
  Pant- brands, half, size, price
- Design architecture (developer more 10 to 12 years)
- System architecture

HLD

- HLD is created by system architecture or design architecture
- Designing structural functionality of main module know as **external design**
- It includes what & how any main module work
- Include relation dependency of main module

LLD

- Defines static logic for every sub modeling

- Designing structural functionality of sub module know as **internal design**
- LLD design is created by front end developer

Task- See HLD & LLD particular application

Coding / Programming- developer

- Coding means **programming**
- One line is **code**
- Multiple line is known as **coding / programming**
- It is **set of programming language designed, written by programmer/ developer** know as **coding / programming**
- Java script, php, .net, python…..etc.
- **Developer** – are involved only – coding
- Developer will do coding on **LLD**

Ex.
Paytm
Recharge module – design architecture / developer
Radio button
Mobile number
Operator
Amount
Check box
Payment submit

Developer – there are 2 types

Front end developer-
UI- user interface
Functionally flow
Process
Developed by front end developer

Back end developer
Data management
Data gathering
Data security
Developed by back end developer

Full stack developer = front end developer + back end developer

Testing

It is a process to check completeness & correctness of the application / software w.r.t client requirements

Testing having 3 types

1. White box testing – Developers
2. Black box testing – Tester
3. Gray box testing – Tester

WBT – 2 testing

- It is code level testing approach to check or test **completeness & correctness** of program
- WBT is done by **developer**
- WBT only **developer are involved**
- WBT is also **called as**
    1. Code level testing
    2. Unit testing
    3. Glass box testing
    4. Clear box testing
    5. Transparent testing
- Once the **developer complete programming** or coding then **developer test** or check their **own code** & if any **error found** then **developer solve** it
- Developer test or checks **only positive way**
- Developer aware about the **internal coding / programming** of the application
- Developer test their **own code** & **make sure there is no error** / bug / defect
- Developer can't send the code to the **tester without doing WBT**

BBT- 18-19 testing

- BBT- as a **tester verify / validate internal functionality** of the application **depends on external functionality or external interface or front end**
- BBT is done by **tester**
- BBT testing **only tester are involved**
- BBT is also **called as**
    1. Build level testing technique
    2. System & functional testing

    (**Build**= Url=https://www.flipkart.com/)

- In BBT **overall functionality of application** is checked step by step from start to end

- In BBT- tester test or check both way like **positive & negative way**
- Tester **not aware about internal coding / functionality of the application** so to validate internal functionality depends on external interface

Ex. Login

UN- Mangesh

PW- Mangesh123

Login Button – Process = successful login

GBT

- GBT combination of **WBT & BBT**
- GBT testing is done by **tester**
- GBT only **tester are involved**
- To perform the GBT- tester need or should have programming language knowledge
- Whenever final software handover to the tester, tester check its functionality & if any error / defect / bug / fault in the output of the function in such case tester makes some changes in code itself instead of assigning to the developer
- Advantages
  Time
  Efforts save

What is difference between WBT & BBT?

WBT (White Box testing)	**BBT** (Black Box testing)
1. WBT is **performed by developer**	**1.** BBT is **performed by Tester**
2. WBT is 2 types - Unit Testing - Integration Testing	2. BBT is 2 types - Sanity Testing/ Smoke testing - System & functional testing - Re-testing, Regression testing etc.
3. In WBT check, Logic for Code, Condition statement, Loop statement, branches, etc.	3. Tester will do the check some coverage – Input domain coverage, error handling coverage, Backend coverage, etc.
4. WBT also called as **code level testing**	4. BBT also called as **System & functional testing**
5. It's a coding level testing technique	It's a build level testing technique
6. It is known as clear box |Glass Box| Transparent Testing	It is called as system & function testing
7. In WBT developers tested their own code	In BBT Tester test end to end functionality

8. Check for +Ve scenario	Check for +Ve & –Ve scenarios
9. Aware about internal structure	Not Aware about internal structure

Testing having 2 way – as a testing point of view

1. Test case design
2. Test case execution

Client – req.

BA- collect req.

BA- prepare BRS

BA- prepare SRS

After the completion of the SRS document

BA sent this document to the technical team (designer, developer , tester)

Developer	Tester
After getting the doc.	After getting the doc.
Understand the req.	Tester understand the req.
Design	Test case design
HLD/LLD	Test case review (self / peer / internal / external)
Coding	
WBT	

- Unit
- Integration

Build- Url

Test case execution

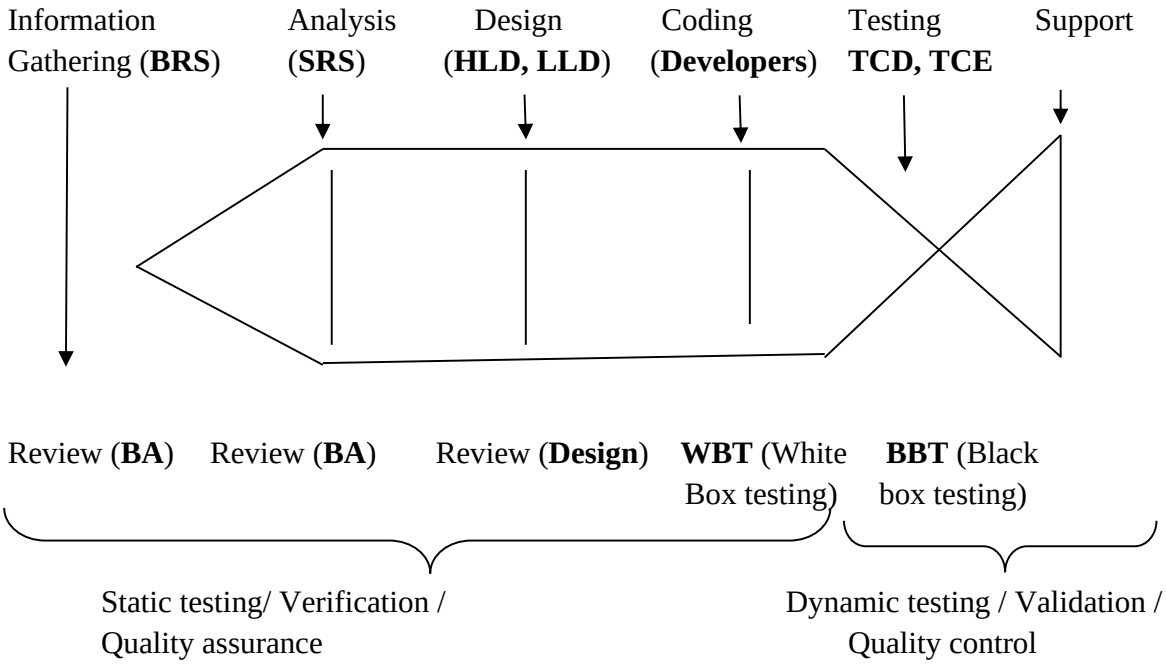
Review- Pass / fail

Defect

Maintenance

- After delivery of the project or application or software. If there is problem or any technical difficulty, in such case company has to fix it without any cost
- Maintenance
  Technical support - KPO
  Non-technical support – BPO
- Ex. customer care / call center

Fish Model



- Fish model it is an advance version of SDLC
- Fish model consist of verification & validation

Ex.

Tata & Tesla

Pre launching – One charge – 100km – Quality Assurance

Actual Launch- One charge – 100km – Quality control

Review / Feedback

Review

It is process to check **correctness & completeness** of the **own document**

Verification

- Verification process is known as static testing / quality assurance
- Verification process is a actually review process

Requirements gathering

- In RG stage, BA collects all the client requirements & prepare BRS document

Review

- After the preparation of BRS document, whether it is correct or not because further SRS document is depends on BRS
- This checking is process is known as review process
- BA has to do review process

Analysis

- In Analysis stage, BA prepare SRS document from BRS document

Review

- After the preparation of the SRS document. BA has to check SRS document. Whether it is correct or not because all the further depends on the SRS document
- This checking is process is known as review process
- BA has to do review process

Design

- **S**ystem architecture divides design in to HLD & LLD
- After completion of design stage, design architecture has to do review process

Coding

- When developer complete the coding, developer has to check all the own code
- Developer has to compile code. Find error/ defect / bug & fix the error
- This process is called unit testing / code level testing

Static testing

- In the static testing it is process- the responsible authority do only review process

Validation

- Validation process is known as dynamic testing / quality control
- Dynamic testing focuses on quality of the product
- Overall / entire functionality of the application is checked step by step from start to end
- Validation includes BBT & GBT

Testing

- Tester test the overall functionality of the application step by step from start to end
- Execute positive & negative scenarios

What is difference between verification & validation?

Sr. No	Verification	Validation
1	This process known as **Quality Assurance**	This process known as **Quality Control**
2	It is process oriented, during the process - we are **providing assurance about QA**	It is product oriented , during or after testing we are **providing assurance about QC**
3	QA – **Software development life cycle**	QC – **Software Testing life cycle**
4	Also known as **static testing**	Also known as **dynamic testing**
6	Verification is **review process**	Validation is **End to end Testing process**
7	It is an **preventive Technique** – Main aim is **prevent the Defect/Bug**	It is **Corrective Technique** – identify the defects & provide fixes

Difference between Static & Dynamic Testing

Sr.No	Static testing	Dynamic testing
1	In Static testing BA, Designer & Developer are **checking** their own documents, design & code	In dynamics testing, Tester will **check functionality** of the application
2	In Static testing **BA , Developer & Designer are present**	In dynamics testing **Tester are present**
3	Static testing also called **Verification**	Dynamics testing also called **Validation**

4	Throw Static testing, we will define **quality assurance of the application**	Throw dynamic testing, tester will define **quality control the of the application**
5	Static testing also called **In-progress testing**	Dynamic testing also called **End-progress testing**

Interview Questions

1. What is software testing
2. Manual Testing
3. SQA? Features?
4. What is SDLC? Stages in SDLC?
5. How Many tester & Developers in your Team?
6. What do you mean by SRS & BRS documents? Or Difference?
7. Black Box testing?
8. What is Gray box testing?
9. Are you be the part of WBT?
10. Did you worked as Gray box tester?
11. Testing start before development or after development?
12. What is software project?
13. What is snapshot
14. Team size? Roles?
15. Who is good tester?
16. Why you joined as a tester?
17. What is team size in last project?
18. What is the tester in your project?
    Answer- In my last we have 4 testers. In which I am working a Manual Tester, Test lead working as Manual/Automation/API testing. 2 Tester- 1 work manual testing & 1 automation testing
19. What is size of developer in your project?
20. In which team you have worked?
21. What are SRS documents & have you see SRS documents?
22. When are we starting the testing?

    Answer- Whenever we got SRS documents from BA throw Mail, then we are ready / understand SRS documents. If we have doubts in SRS, then we are conducting the meeting with BA. When we have cleared the doubt in SRS documents, we start TCD & TCE

23. What are SRS documents will contain?
24. What is difference between WBT & BBT?
25. What is difference between Static Testing & Dynamic Testing?
26. What is difference between Verification & Validation?

Methodology / implementation / process of SDLC

Process a way to develop & test the software

1. **Water model**
2. **V model**
3. Agile model = 90/95%

Who will decide the process?

- If client has IT department – then client will decide the process
- If client not has IT department- then company / BA will decide the process

Difference between Waterfall, V & Agile model?

Waterfall	**V**	**Agile**
Requirements are **fixes from client**	If requirements are changed CR- client have to **pay extra amount**	Requirements CR are vary but **no extra amount is taken from client**
Used in – Small project	**Used in – Big/Large project**	**Used in – Big/Large project Complex project / Critical**
It is step by step or Continuous process	Simultaneous process	Flexible process

Project –

Shree Ganesh Online Shopping

Waterfall – Fix requirements – 10cr – requirement can't change further

Men Women

Men

Shirt

Pant

Women

Sari

T-shirt

V model

Initial condition – requirement are fix- 10cr – but client want some change requirement – requirement – **extra amount pay**

Project –

Shree Ganesh Online Shopping

Men Women

Men

Shirt

Pant

T-shirt

Women

Sari

T-shirt

Agile

Requirements **CR are vary** but **no extra amount is taken from client**

Project amount- Water fall (10cr) + V model (10cr+extra amount per requirements)

Agile project cost= 15cr (no of requirements change any time)

Project –

Shree Ganesh Online Shopping

Men Women

Men

Shirt

Pant

T-shirt

Jeans

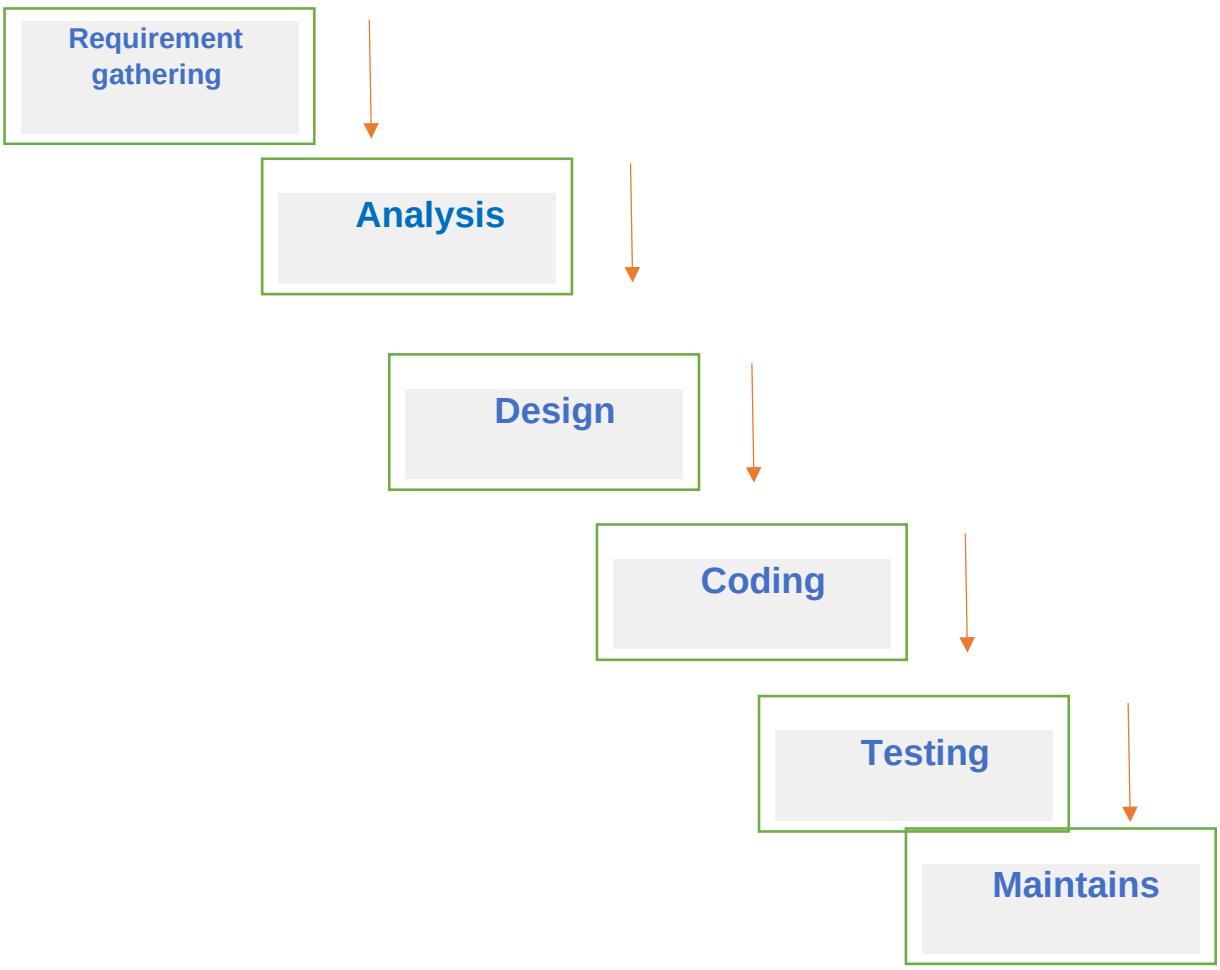
Women

Sari

T-shirt

Shirt

Waterfall Model



- Waterfall model it is **step by step implementation of SDLC**
- Waterfall model It is **linear sequential model** for software development & testing
- Sequential model means – **after completion of first stage** then **second stage** will be come
- Ex. Sequential means- **Testing will start** when **coding is completed**
- Ex. Sequential means- **Developer will start the coding** when **design is completed**
- In this waterfall model **client requirements are fixed**
- It is used **small project & small budgets** etc.
- Duration / delivery of the project in the waterfall model is **more than 3 month**
- 3 month – **5 to 6 model delivery**

Ex.

Client – 50 model

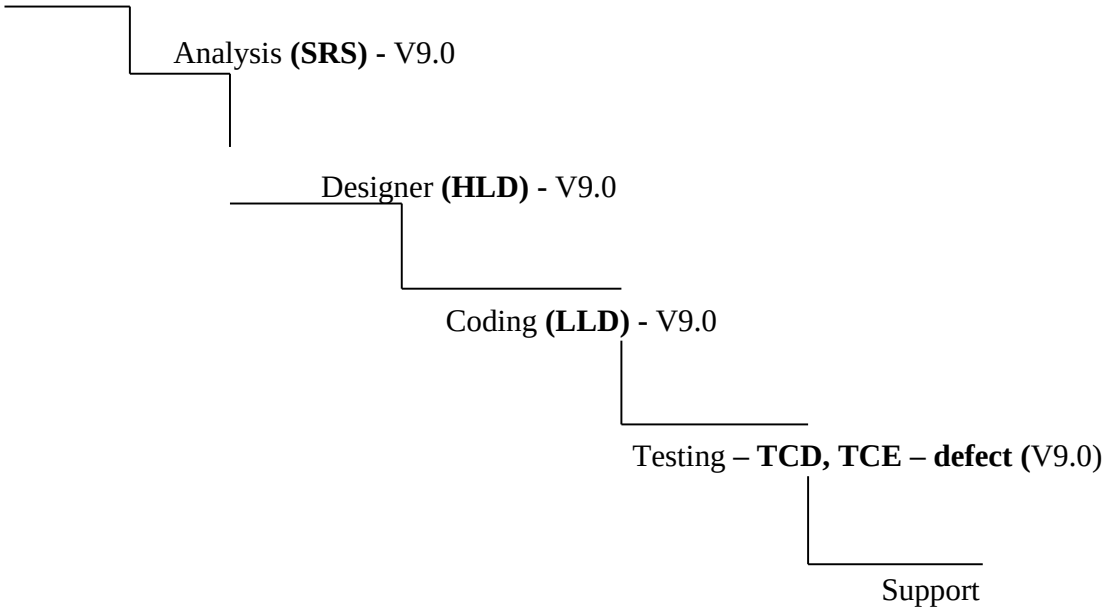
Duration / delivery / release = 3 month = 5 to 6 model – 1 release

Duration / delivery / release = 3 month = 5 to 6 model – 2 release

- Case 1
  If you want to **enter in to next stage**, the **current stage should be completed** & then only we can **enter in to next stage**
- Case 2
  Consider **coding stage has been completed** & now **testing stage** is there
  If any **defect occurs**, then tester **cannot revert back to the developer** or **coding stage**
  In such case, **tester has to assign/ lock the defect** in the testing stage & prepare report of it
  Then this **defect is fixed** (correct/resolve) in the **next version**

Information

Gathering (**BRS**) – V9.0 – New Requirement (V9.1) + Old defects (V9.0)



Disadvantages

1. Backtrack is not possible
2. Delivery to client more than 3 month

What is difference between Build / Release / Version?

V 1. 0

V 1. 1. 5

Build 1

Build- 2 to 3 model

Build 2- 3 to 5 model

Build 3- 1 to 2 model

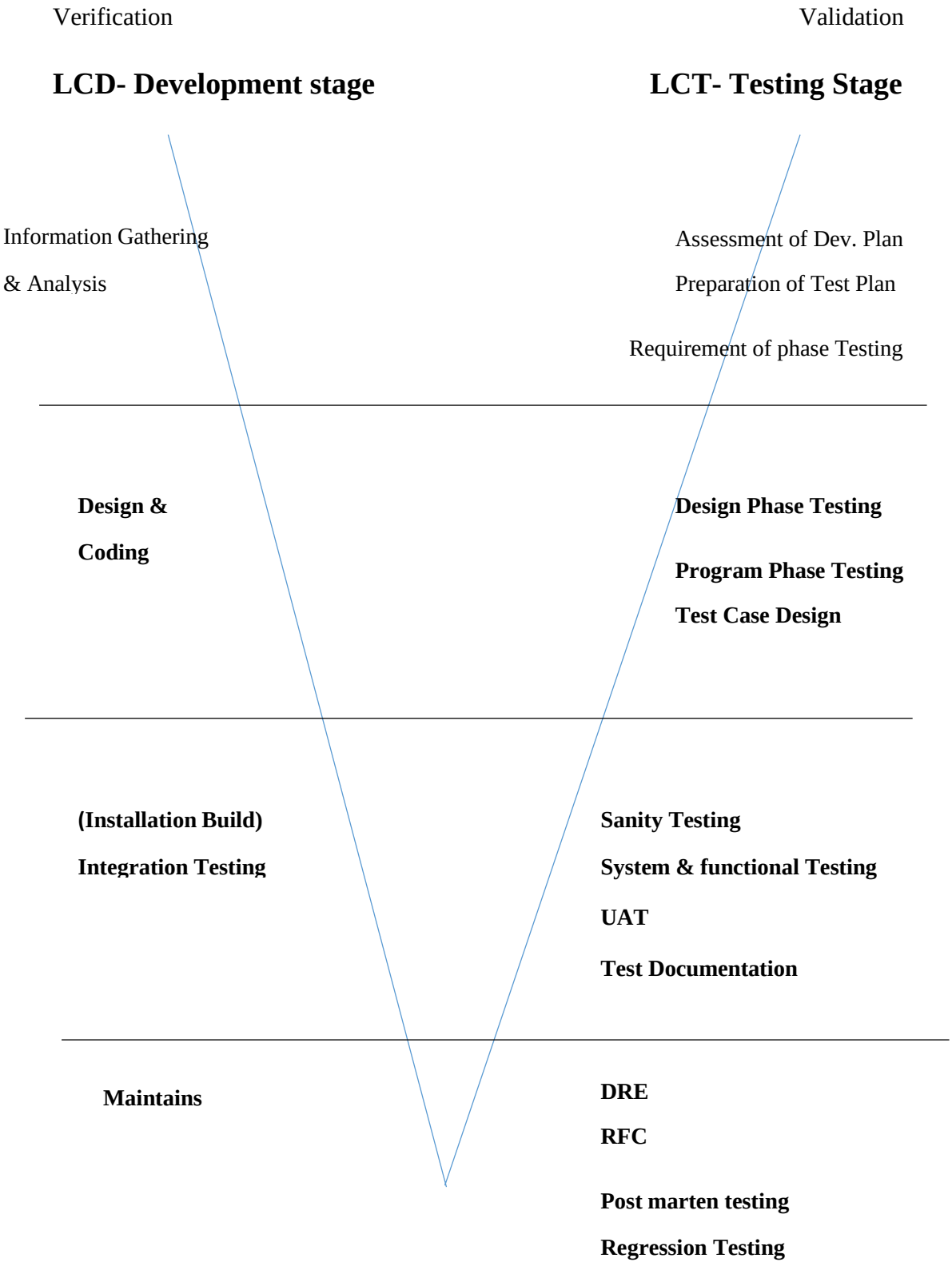
Release – 5 to 6 model develop -

V 1. 0 – 2 to 3 model

V 1.1- 1 to 2 model add – new req.

V 1. 1. 3 – 1 to 2 model – new req.

V Model


Verification
Validation
LCD- Development stage
LCT- Testing Stage
Information Gathering
& Analysis
Assessment of Dev. Plan
Preparation of Test Plan
Requirement of phase Testing
Design &
Coding
Design Phase Testing
Program Phase Testing
Test Case Design
(Installation Build)
Integration Testing
Sanity Testing
System & functional Testing
UAT
Test Documentation
Maintains
DRE
RFC
Post marten testing
Regression Testing

V Model

- V model it is known as **Verification & validation**
- V model, verification & validation **work parallel/ simultaneous process**
- V model, **development stage are mapped with testing stage**
- In v model, suppose, we have **completed 1st stage** & now we are in the **second stage** which is running. If any **change in requirement** (CR) comes for **1st stage or previous stage**, then we can **revert back** to the **previous stage or 1st stage** to **full fill CR** but client have to **pay some extra amount**

Ex. Shree Ganesh application – Initial condition – Fix requirement – development proceed – Req. change – extra amount pay

Req. change – accept BA- Ex. Testing- Req. its related development- BA accept – take charges

- V model used in the **big organization**
- V model, duration / delivery of the project is **3 or more than 3 month**
- Delivery **– 5 to 6 model** develop – release
- It is **plan driven methodology** – because **CR are rarely** come

Information Gathering & Analysis – refer SDLC

1. Assessment of development plan
2. Preparation of test plan **Detail cover in the Manual Part 2**
3. Requirements of Phase testing

Assessment of development plan

- Defining **objective of the project**- domain- Banking, telecom, insurance, health care….etc.
- Defining **steps to how we can achieve objective** project
- Strategy for **project development**
- Strategy for project **testing prepared**
- In testing point view – there are **automation testing & manual testing**.
- Among these 2, which **methodology** need to implemented is decided here
- Test responsibility matrix (**TRM**) is finalized in this stage
- Stage – top level people – PM/Test Manager sometime CEO,MB,VP

TRM – Mapped – development stages with testing factor

Testing factors- 15 to 17 testing factors – testing – 10 to 11

Ex.

BCCI – Format – Test/Day/T20

IND vs SHRI

BCCS

BCCI- Rahul/ Rohit

Preparation of test plan

- **TRM is implemented** in the Preparation of test plan
- **PM/TM** is **responsible** for **TRM implementation**
- PM/TM prepare a **test team (5 tester)**
- PM/TM assign – **Test lead**
- PM/TM **distribute work** to all tester
- **Test estimation** is created in this phase
- **Estimation**- means how much time it will take to complete particular assigned task (start & end time)
- In this phase
    1. **Job allocation** – 11 testing
    2. **Resource allocation** – 5 tester
    3. **Estimation** – time duration to complete particular task

Ex.

Coach- Rahul - PM

Captain- Rohit - TL

Team- IND vs SHRI

TRM- testing factors – 17-18 – Team India member – 17-18

Final team- 11 – TRM- 11 testing

Select- 11

Batsman-5

Keeper-1

All-rounder- 1

Bowler- 2

Spinner- 2

3. **Requirements of Phase testing** (Ex. 10-15 days)

- Phase means Unit

- Ex. Registration page – All country

FN

LN

DOB

Email Id

Mobile No- only accept 10 digit mobile number

Missing – country code requirements

- In this stage, estimated requirements of the phase finalized
  Ex. Whatsapp
  Introduce
- Test message
- Audio call
- Video call

Design & Coding

1. Design Phase Testing
2. Program/ coding phase testing **Detailed cover after agile**
3. Test Case Design **Detail cover in the manual part 2**

Design Phase Testing & Program/ coding phase testing

- Program phase testing – **code testing**
- Code testing is started from **small unit of program**
- **Developer** are involved in the program phase testing
- Check their **own code**- error – fix the error is like **unit testing**
- Is like **white box testing**

Test Case Design

- Tester understand **SRS document** & then **prepare test case design**
- Test case include
  Positive test case design
  Negative test case design
- **Tester** are involved

- It is like **BBT**
- Test case design will be executed in the **later stage**

Install Build/ Integration Testing

1. Sanity Testing
2. System & functional testing **detail after agile**
3. User Acceptance testing
4. Documentation **detail in manual part 2**

Install Build/ Integration Testing

- Delivery / duration – 3 month generally 5 to 6 module are developed
- Ex. for running project. If organization receives new requirements from client to develop new module
  If client wants to add this new module into the existing flow then we can called as integration testing
  Combine depend module
- In such case, developer comes into picture, where developer work on the new req., developer code for new req. & perform unit testing/WBT also
- So, when new module is ready, they add/ integrate new module in the existing module / application flow

Ex.

Amazon E-commerce

Registration- Login- Home- Fashion- Mobile- Electronics- Flight- Beauty-…..etc.

Existing flow

Mobile- View Product- Buy Product/add – place order- payment – delivery

New Req. – Exchange old mobile

Mobile- View Product- Buy Product/add – Exchange- place order- payment – delivery

Device Company

Module

Year

IMEI

Price

New mobile – 25k

Exchange mobile- 7k

Final price payment – 18k

- Developer are involved in the integration
- Integration testing 2 types
  1. Front end integration
     Front end developer add or combine all dependent module by using **CALL** function
  2. Back end integration
     Back end developer combine all the tables together using **JOIN** function

1. **Sanity Testing**

- It comes under the **validation process**
- In sanity testing as a tester validate **basic & core functionality** of the application/ build
- Check **the happy flow of the application/build**
- Tester check application – **blocker**
- Sanity testing- only **blocker defect** / critical defects are assign to developer
- **Tester** are responsible for sanity testing
- Duration – **2 to 4hr**
- Check application / build – **stable or unstable**
- Sanity testing don't **execute test case**

Ex.

IRCTC

Plan my joinery – my booking - PNR enquiry

Refund history

Source to destination

Class

Date – Date format – DD-MM-YY MM/DD/YY

My Booking – reservation information – date- time- class- train number

No of station count – list – count – Latur – pune- 18 count – display – 17

Railway logo

Search train – not working – blocker – sanity testing – assign to developer

2. **System & Functional Testing (BBT)-17-18**

- After the completion of **sanity testing**, we carried out system & functional testing
- BBT
- In this testing, as tester to check / validate overall / entire application step by step from start to end
- End to end testing process
- Testing execute positive & negative test cases
- In this testing small defect, large defect, blocker are assign to developer
- Ex.
  Date format
  Logo
  No of station not proper

3. **UAT**

- After the system & functional testing we carried out UAT
- After system & functional testing we assure about **defect free build / application**
- In UAT – **client** – BA-PM-designer, dev team- test team – sit together & check the entire functionality of the application start to end as **per the given requirement**
- Ex. UI, main module, sub module, logo, font, color, alignment etc.
- After validation of product in UAT, product is sent to the production

4. **Test Documentation**

- Tester prepare & maintain testing reports
- Include- daily testing report, TCD report, TCE report, defect report, test summary report (TL)..etc.
- Tester send this report to – test lead
- Test lead send this report to- project manager
- Project manager send this report to– BA
- BA send this report to- client
  Ex.
  Daily testing report
  Name of module Test case execution Status (Pass/Fail) Comment

Maintenance – refer SDLC

1. **Defect removal efficiency**
2. **Request for change**
3. **Post mortem testing**

4 Environments

DIT – Developer

SIT – Tester

UAT – Client/BA/PM/DT/TT

Prod- Live

Defect removal efficiency

For

Ex.

Tester – Application – Paytm- No of error / defect – client

- It is a process of **calculating at which level tester performed testing / tester tested the application**
- DRE has **2 phase**
  1. Defect found by tester
  2. Defect found during UAT (client)

Defect found by tester

During testing, if tester found defect then some of the defect are fixed (resolved/correct) & some of defect are cancelled

Ex.

In SIT, consider tester found 100 defects

90 defects – were fixed

10 defects – were cancelled

(Due to technical issue, duplicate, not a defect)

So only 90 defects are consider

Defect found during UAT (client)

If client found some defect during the testing of the application

Ex.

In UAT, 10 defect were found

Consider

A= defect found by tester -90

B= defect found during UAT- 10

Defect removal efficiency

DRE= (A/A+B)

= (Defect found by tester / (defect found by tester + defect found during UAT))

= (90/(90+10))

=0.9

Based on this what of kind of testing has been performed is calculated

DRE (defect removal efficiency)	**Remarks**
0.8-1	Good Testing
0.5-0.8	Average Testing
Below 0.5	Below Average Testing

Duplicate-

Ex. Pytam- recharge- mobile number field doesn’t have length validation

Pytam- recharge- mobile number field accept less than 10 digit

Pytam- recharge- mobile number field accept more than 10 digit

RFC- request for change / CR – change in request

Ex. Ganesh online shopping

Men Women **Kids**

1. If customer wants to **some change in the product** / application at the **time release / development / testing** then it is considered as **request for change or change in request**
2. To handle this RFC/CR- there is one team is called as **"configuration management team"**
3. CMT= **involve** – BA / developer / DL/ tester/TT
4. So, in which **environment we did the changes** is decided by CMT
5. Change in request is also mentioned in the **SRS document at the end**. It is mentioned in **red color with * mark** then after that also we get the **updated SRS pdf**
6. So, for this change in request / RFC for change **client has to pay extra amount**

Post Mortem Testing

Ex. Ganesh Online Shopping

Production – desired output not generating – developer – each and every module

- It is used to check **critical functionality of the application**
- PM testing – **developer are involved**
- When whole **testing is done** & **product is ready for production** & if **product is not producing** the **desired output** then **developer has to check** all the modules in details & has to **perform WBT**
- In this testing developer has to **find out exact root cause of the defect** – where it is & what is the problem

Developer – code – server – DBA (developer) team – code drag – push – URL – Jenkins tool – deployment – production

Regression Testing – details after S & F testing

- Mobile number filed – not accept digit – developer – developer change the code
- RT – performed on modified build to ensure that defect will be fixed & side impact other sub module
- Build / URL – mobile number filed not work
- New build – modified

Agile Model

Content

Agile introduction
Difference between naming convention (SDLC vs Agile)
Different type of Agile
Agile architecture
Agile meeting
Agile Adv & Dis.
Agile daily work plan
Agile terms
Interview questions

Ex.

Waterfall – 3 month – 5 to 6 module -10cr -
V model – 3 month- 5 to 6 module – 10 + CR (amount)
Agile- 1 week / **2 week** / 3 week / 4 week – **1** to 2 module – 15cr

Agile introduction –

- Agile methodology / process / model is the **module driven methodology**
- In agile methodology **requirements changes frequently, so it is not a plan driven methodology**
- In agile methodology client can **request for change (RFC) in requirement at any point of the development stage**
- Client can RFC at any stage like **DIT, SIT, UAT & Production**
- If any CR, that will be **accepted at any point without extra amount**
- If any CR come from client then we will accept & check impact on current development, testing & production process
    1. **If impact is more on current development & testing**, then BA/PM will discuss with client or inform to client (ex. Kids)
    2. **If impact is less** then we will consider new CR (ex. +91)
- **Agile duration of delivery** is of **2 week** to 4 week (Fix- 2 week)
- Agile methodology is a **value driven methodology** (we are **giving priority to client**)
- Agile methodology, project is divided in to no of module & release
- As per the client priority- no. of module have to be developed, module wise delivery is possible in agile
- It is flexible process

Ex. agile – project is divided in to number of module & release

1 2 3 4 5 6

7 8 9 10 11 12

Release 1 - 2week- client wants module 1 & 3 develop

Release 2- 2 week – client wants module 2 develop

Difference between naming convention (SDLC vs Agile)

SDLC/Waterfall/V Model	**Agile Model**
Customer / client	Stakeholder
BA	Product Owner / Project Owner (PO)
BRS	Product Backlog
SRS	Sprint Backlog
Use case	User Story
Release- 3 month	Sprint – 2 week
PM	Scrum Master
Extra amount (V)	No extra amount
Developer	**Developer**
Tester	**Tester**
Designer	**Designer**
Delivery Manager	Solution Master

Different type of Agile
Different flavor in agile
Different framework in agile
Different sub model in agile
Different sub type in agile

Agile will contain different type/flavor/framework/sub type/ sub model

1. **Kanban- Support team**
2. Lean – Support team
3. XP – Extreme programming (only developer & no tester)
4. **Scrum- Project team/ main team – sprint wise delivery to client/ stakeholder with 2 week (Sprint 1)**
5. FDD- Feature driven development
6. DSDM – Dynamic system development method
7. Crystal

Agile

- Agile is the development methodology/process/model based on iterative & incremental approach
- Agile Introduction

Scrum

- Scrum is a framework used in agile methodology
- Scrum is used to implement agile methodology
- Scrum in an agile framework that can help teams work together

Sprint (2week)

- Sprint is part of Scrum framework structure
- Sprint is defined period for creating feature/module
- During a sprint, specific module / feature of the product / application is created
- A set period of time during which specific task must be completed
- Sprint duration can vary – 1 week to 4 week (2week)

Agile Architecture

SDLC		**Agile**
Information Gathering (BRS)	BA/PO	**Product Backlog** (1 Project- 2000 US)
Analysis (SRS)	BA/PO	**Sprint Backlog** Sprint 1= 20US – priority of client Sprint 2= 15 US Sprint 3= 20 US
Use case 1. Description 2. Acceptance		**User story** Description Acceptance
Designer (**HLD** & LLD)		**Designer** (**HLD** & LLD)
Coding (**LLD**)		**Coding** (**LLD**)
Testing- TCD/TCE		**Testing- TCD/TCE**
Support / Maintenance		**Support / Maintenance**

1. **Stake holder**

- Stakeholder is nothing but a client / customer
- Stake holder comes with bunch of requirement / own ideas
- Stake holder is a member of the top most body of the company
- In agile methodology, stakeholder can request for change in requirement at any point of development stage / at any stage

2. **Product owner**

- PO collect / gather requirement from the stakeholder
- PO prepare product backlog
- **PO- team member sprint planning meeting**

3. **Product backlog**

- Product backlog is a created by PO
- In product backlog total/overall requirements of entire product / application
- It includes requirements of all modules

4. **Estimation**

a. Estimation

- In agile methodology the focus is on module wise delivery
- We get all requirement (2000) & those requirements are not for one particular module
  Ex. registration module- 20 req.
- So, in estimation requirements are **sorted for development**
- **Estimation is an important parameter of sprint planning meeting**

b. Estimation

- Estimation it is process to check how we can deal with problems
- Estimation of number module in the project
  So, we can assign number developer & tester according to the project
- Priority based module- client
  Depends on client requirement
- Estimation – involved PO/DL/TL/SM
- There are 3 main factor in the estimation
  1. Knowledge
  2. Efforts
  3. Complexity

Knowledge

- Knowledge about the domain of the project are checked
- Experienced & non experienced resources are taken into consideration

- After that team is formed, after formation of team each member of the team should have knowledge about domain of the project
- KT (Knowledge transfer)

Efforts

- Authority decide (PO/SM/DL/TL) how much efforts are required for project
- Authority decide how many resources (people) are required for project

Complexity

- Time, resources, cost
- Less resources- work load / delay in delivery
- Time
- Cost – escalation

Sprint Backlog

- Sprint backlog it is created by PO
- Sprint backlog contains user stories of that particular module (ex. registration)
- Sprint backlog contains detailed information of requirement, which are required for development in sprint (duration-2week)

User Story

- User stories is nothing but functional requirement
- User stories are decided in to estimation phase **(sprint planning meeting)**
- In estimation, sprint planning member decide which module have to develop & what are the requirement of that module (ex. registration), those sorted requirement are include in the sprint backlog
- So those user stories are functional requirement for the module are to be developed in sprint
- **User stories have criteria**
    1. Description criteria – details about requirement
    2. Acceptance criteria – does & don't about requirement

Description – it is description about what user want to do (process) & what is his desired output

Acceptance – system generate correct output otherwise system show failure

Description criteria template

As a [user/person/actor]
I want [process]
So that [benefits/output]

Acceptance criteria template – refer functional requirement – ex. registration
Given [content]
When [specific action is performed]
Then [should occur]

Given [situation/preconditon]
When [user action 1,2,3]
Then [action1]

Ex.
As a user, **I want** to be securely login in to the system **so that** my information can only be accessed by me (ex. gmail, fb)

As a user, **I need** to search for products, **so that** I can find the once I want to buy

Test case design

- Test case are designed by tester
- Test case create based on the test scenarios – user stories

Agile Meetings / Ceremonies

- **In agile 5 types of meeting / ceremonies**
  1. Grooming meeting
  2. Sprint planning meeting
  3. Scrum meeting / daily stand up meeting
  4. Sprint review meeting
  5. Sprint retrospective meeting

Agile – Sprint (2week) - 1 or 2 module deliver- US-10 - Grooming meeting

Sprint duration – 7/3/22 to 21/3/22 (2week)

Grooming meeting (Tuesday/Thursday)- conducted by PO

- Grooming meeting/session is conducted, **before the start of sprint**
- **Purpose**
  1. To understand the **objective of the project/module (1 or 2)**
  2. To understand the **purpose of stakeholder requirements that particular module**

3. To understand the **user stories of that particular module** will **develop in the sprint**
4. **Product owner** share **essential information** & also **provides guidelines to development & testing team**
5. If we have **doubt about the requirement/user stories**, discuss with **product owner & clear doubt**

- **People involved** – product owner, scrum master, designer, dev team, test team
- **Duration**- 30 to 60 min (60min)

Sprint planning meeting – conducted by SM

- Sprint planning meeting is conducted by SM on the 1st day of the sprint (7/2/22-Monday)
- **People involved** – product owner, scrum master, designer, dev team, test team
- In the **grooming session**, we get overall **idea about the module** or **current sprint user stories** (ex. US-15), so in sprint planning meeting, **SM allocates** work/task & user stories to the **development team lead & testing team lead** as per requirements & priority of the stakeholder
- The development team lead distribute work / task to the development team members, according to experience of the employee, requirement & priority of the stakeholder
- The testing team lead distributes work / task to the testing team members according to the experience to the employee, requirements & priority of the stakeholder
- SM asks for **estimation** (how much time it required to complete particular task) / story point (**hr** basis or day basis)
- Ask for estimation = task & user story
  Day 8hr/week = 40hr = 2 week = 80hr each team member estimation
  Ex. 1 US estimation/story point = designer (2hr)+developer(14hr)+Tester(8hr)= 22hr
- Once the sprint planning meeting is done, **SM monitor each activity on daily basis**
- Duration 30 to 60 min(60min)

Daily standup meeting – conducted by SM – (time decide client location-9am-10am)

- It is daily / everyday status call or standup call meeting / scum call
- People involved – product owner, scrum master, designer, dev team, test team
- SM is a chairperson of the daily stand up meeting
- Agenda of this meeting is **discussion about the progress of the project** or **work progress of the designer, developer, tester**
- **3 questions ask in scrum meeting**
  1. **What we did yesterday?**
     It is the report of previous work which was completed by team members whatever they are tester or developer
  2. **What we will do today?**
     It is the work from pending work which team member have to complete by the end of the day (EOD)

3. **Discussion about road blocker/ blocker or issue?**
   This include – difficulty in existing test case, execution, error or defect in the code compilation, lack of cooperate from team

- Duration – 15 to 30min

1. What we did yesterday? (before development testing activity)
   Good morning all, so, yesterday I have understand US & I started with test case design of this particular module "Login". I have completed 60% of the test case of the particular module "login". I had some doubt, so I had connected with PO/development team

   What we did yesterday? (After development testing activity)
   Good morning all, after the completion of test case design. I have executed few test case & here is the result, so its working fine. I have found some defect, so that I have raised / assign / lock defect to that particular developer & I have communicate with developer for that particular defect

2. What we will do today?
   We will complete remaining part of the test case execution. So at the end of the day I will share test case sheet / result with all team

3. Discussion about road blocker/ blocker or issue?
   While executing test cases, there was a blocker, so due to this we could not test that module

Sprint Review Meeting / Demo meeting (conducted by PO/SM)

- Sprint review meeting is conducted on the last day of the sprint (Friday- 2:30-3:30pm)
- PO/SM take review of the work
- People involved – Stakeholder, PO, SM, Designer, DT,TT
- Development team or test team prepare demo of application (about functionality / US of the application) & delivery to the PO/SM/Stakeholder
- In this review, requirement / US are cross checked, some suggestions/comments & some additional changes are also told (CR)
- This review is taken to check completeness & correctness of the application as per the stakeholder requirement
- Duration- 60min

Sprint retrospective meeting

- Sprint retrospective meeting is conducted by SM on the last day of the sprint (Friday- 3:30-4:30pm)
- People involved- PO/SM/ Designer /DT/TT
- In this meeting we discuss about the sprint
  1. What went well – good
  2. What didn't go well- bad
  3. Difficulty faced by development & testing team
  4. What were the action plan for next sprint or module
  5. Overall experience is shared about the sprint (good/bad)
- This meeting help to avoid mistakes in next sprint
- Duration – 30-60min (60min)

Summary of all meetings

Agile Meeting	Purpose	Involved
Grooming meeting **(** Any time- 2week) (Before start of Sprint)	- US / requirement **doubt/ Clarity understand**	**1hr – PO,** SM, Development team, Testing team, Designer
Sprint planning meeting **(**1 times – Start day of sprint) (1 Sprint - 2week)	- **Current sprint = 20 US added / work decide** → SM, PO & Designer - **Estimation provide** Ex. 1US = designer (2hr) + developer (14hr) + Tester (8hr) → total = 24h	**30 min/ 1hr** – **PO,** SM, Development team, Testing team, Designer
Daily stand up meeting/ Scrum meeting (Daily – 10 am to 10.15am)	- **What you have done yesterday** - **What are you doing today** - **Issue/ roadblock**	**15 min –** PO**, SM**, Development team, Testing team, Designer
Sprint review meeting **(**1 times – End day of sprint)	- US → **Tester or developer demo/ review to Client/ UAT** / **PO**	**1hr – Stakeholder/ UAT** / **PO, SM**, Development team, Testing team, Designer
Sprint retrospective meeting **(**1 times – End of day sprint)	- Current sprint (Sprint 1 → 2 week complete) → Sprint 2 work **Good & Bad things discuses**	**30 min/1hr- SM,** PO**,** Development team, Testing team, Designer

Agile Advantages & disadvantages

Advantages

1. Check points

   1 2 3 4 5 6
   * * * * *

   Each application- main module – sub module
   V model – any defect at the time production – Post mortem – all module
   Agile – any defect at the time production- check point

2. Scrum meeting (daily activities work monitor)- refer above meeting point

3. Implementation automation
   We can implement automation in agile
   Now days – 75% manual & 25% automation testing
   Advantages
      Less resources
      High accuracy
      Less cost
      Less time

4. Sprint wise delivery
   V model – 3 month - 5 to 6 model – client wait
   Agile – 2 week – 1 or 2 module – client not wait long
   1 year- Agile release- 24 – 24 module
   1 year- V model release – 4 – 20-24

Disadvantages

- Previously developed software- new module develop – developer & tester should have a total knowledge about the flow of software, relationship, dependencies of the module
- If frequently change in requirement, then delivery will take time (client inform)

Agile daily wise plan

- Project team size (18 people) (10 developer/4 tester)
- Agile duration for delivery = 2 week (5*2=10) (Monday-Friday)
- 1 team- Recharge module – 3 developer + 1 tester
- 2 team- Electricity module- 2 developer + 1 tester
- **3 team- invest in stock module – 3 developer + 1 tester**
- 4 team – rent module – 2 developer + 1 tester
- Agile – Scrum- Sprint wise delivery – 18 US (4US- Assign 3 Team)

Grooming session (60min)- 18 US doubt celerity

1 week
Monday (Start day of the sprint)

- Sprint planning meeting (60min)
- Allocation
- Estimation
    1. Sprint 1 = 18US – Assign work / task to developer (4US)+ tester (4US)
    2. Estimation –
        1 US – developer (14hr) + Tester (10hr)
        2 US- developer (16hr) + Tester (12hr) etc.
        3 US – developer (14hr) + Tester (10hr)
        4 US- developer (16hr) + Tester (12hr) etc.
- 1 US – 3 developer – 1 US coding (5/6hr) – In-progress + **Tester – 1US TCD (5/6hr)- complete**

Tuesday

- Daily stand up meeting (15min-30min)
    1. What we did yesterday? (1US- TCD- complete)
    2. What we do today? (1 US- TCE)
    3. Roadblock / issue?
- 1 / 2 US- 3 Developer – 1US coding (2hr)- complete- build sent for 1 US to tester, 2 US coding – In- progress (5/6hr) + **Tester – 1 US TCE (7/8hr)- complete**

Wednesday

- Daily stand up meeting (15min-30min)
    4. What we did yesterday? (1US- TCE- complete)
    5. What we do today? (2 US- TCD)
    6. Roadblock / issue?

- 2US- 3 developer – 2US – coding – in-progress (7/8hr)- complete + **Tester – 2US TCD (5/6hr)- complete**

Thursday

- Daily stand up meeting (15min-30min)
  7. What we did yesterday? (2 US- TCD- complete)
  8. What we do today? (2 US- TCE)
  9. Roadblock / issue?
- 2 / 3 US- 3 developer – 2 US – build sent to tester, 3 US – coding (6/7hr) – In-Progress + **Tester – 2 US TCE (7/8hr)- complete**

Friday

- Daily stand up meeting (15min-30min)
  10. What we did yesterday? (2 US- TCE- complete)
  11. What we do today? (3 US- TCD)
  12. Roadblock / issue?
- 3 US- 3 developer – 3 US-coding (4/5hr)- In-progress + **Tester – 3 US TCD (5/6hr)- In-progress**

2 week

Monday

- Daily stand up meeting (15min-30min)
  13. What we did yesterday? (3 US- TCD- In-progress)
  14. What we do today? (3 US- TCD- Complete + 3 US- TCE)
  15. Roadblock / issue?
- 3 US – 3 developer – 3 US – coding (3hr) – complete- build sent to tester, 4 US – coding (3/4hr)- In-progress + **Tester – 3 US TCD (3hr)- complete, 3 US – TCE (4/5hr)- In** progress
- 3 US – TCE (2 defects)- assign that defect on JIRA & we will inform to developer + developer will fix defects (1/2hr) + **Tester will check defects (2defects) (1/2hr)- complete**

Tuesday

- Daily stand up meeting (15min-30min)
  16. What we did yesterday? (3 US- TCD- complete, 3 US- TCE – In-progress, 2 defect check)
  17. What we do today? (3 US- remaining TCE- complete)
  18. Roadblock / issue?

- 3 / 4 US – 3 developer – 4 US – coding – In-progress + **Tester – 3 US – TCE (4/5hr)– complete**

Wednesday

- Daily stand up meeting (15min-30min)
  19. What we did yesterday? (3 US- TCE- complete)
  20. What we do today? (4 US- TCD)
  21. Roadblock / issue?
- 4 US – 3 developer – 4 US – coding (5/6hr)-complete – build sent to tester + **Tester – 4 US -TCD (5/6hr)- complete**

Thursday

- Daily stand up meeting (15min-30min)
  22. What we did yesterday? (4 US- TCD- complete)
  23. What we do today? (4 US- TCE)
  24. Roadblock / issue?
- 4 US – 3 developer- 4 US – coding(1hr)- build sent to tester + **Tester – 4 US – TCE (5/6hr)**
- 4 US – Tester – TCE – found 7 defect- tester assign that defect on JIRA & inform to the developer – developer – only fix 1-5 defects fixed & developer say that defect no 6 & 7 is not valid defect (developer not accepting defect) + **Tester will check defect 1-5 – Retesting (3hr)**

Friday (Last day of sprint)

- Daily stand up meeting (15min-30min)
  25. What we did yesterday? (4 US- TCE- In-progress, 7 defects, 5 defect- retest)
  26. What we do today? (4 US- remaining TCE)
  27. Roadblock / issue?
- 4 US – 3 developer – 4 US- defect fix (4/5hr) + **Tester – 4 US- TCE (3/4hr)-complete,** 2 Defect found – developer – fix (1/2hr)- **Tester again check 2 defect – retesting (1 hr)**

Sprint review meeting (60min)

1. Tester give demo – 4US demo to client / PO/SM

Sprint retrospective meeting

- Current sprint (sprint 1)- good & bad discussion

Agile Term

Burn down chart

These graph that defines how much work will be remaining w.r.t time/date

Burn up chart

These graph that defines how much work will be completed w.r.t time/date

Velocity chart

Velocity term defines how much US we are deploying or delivering to client w.r.t. sprint

Epic – main module which contain multiple user story in it / large US
Ex. Paytm project- Recharge module **– 15 to 20 US**

Estimation / Story point –

- Defines time span of every US / how much time will required for development & for testing / how much time it will required to complete specific task / US
- Estimation will be decided by using voting method
- In my project- PO/SM/Team will roughly estimate / story point to every US
- In sprint planning meeting, if estimation wants to change then I will talk with SM / TL (daily stand up)
- While giving the estimation which parameters will be considers
  1. How much knowledge, we have knowledge about the US
  2. How to complex for testing that particular US
  3. How much efforts will be required

Ex. US 1- developer (14hr) + (8hr)

Sprint zero

Interview questions
Interview question-

1. What is difference between Water module & V-module & Agile Model?
2. What is Agile, Scrum & Epic
3. What is Burn down chart, Burn Up chart & velocity chart
4. What is agile methodology?
5. What is agile ceremony/meetings/event?

6. What you have **discuses in Last Retrospective meeting?**

   Answer- I last Retrospective meeting, I have discuses

   1. As developer not accepting the valid defects OR developer is not fixing the defects with the time, Testing will delay OR Tester will get more work on last day of sprint.
   2. Testing task(TCD, TCE) should be broken in Sprint – Broker task to TCD= 4hr & TCE= 8hr

7. What is the Agile? How it is performed in your organization/ what is agile daily plan/ Agile Scrum plan?
   Answer- Agile day wise plane have to explain
8. What is your approaches if one of the task (TCD. TCE) if not completed in current sprint? OR If some of defects in not fixed in current sprint, what is your approaches? OR if we found a defect in last hr of last day then what is your approached?

   Answer-

   1. In My Project, If Some task (TCE) is reaming, then I will try to completed with the time. Still not completed then I will inform to PM/SM and Developer that I will work on Saturday and Sunday will complete the task.
   2. In my Project, if defect is taking more time to fix then developer will inform to PM/SM and to tester that he will work on late night or work on Saturday & Sunday and fix the defects.

9. What is advance and dis- advance of Agile?
10. How you're deciding the Estimation?
11. What is Sprint zero?
12. What Scrum of scrum master?
13. What is Agile, Scrum & Sprint
14. What is Kanban & Scrum?
15. What is the US in the Agile?
16. What is entry & exit criteria for integration testing

    Entry criteria for Integration testing- Completion of unit testing is Entry criteria for Integration testing

    Exit criteria for Integration testing - Completion of Integration testing is Exit criteria for Integration testing **OR** Enter criteria for BBT testing is Exit criteria for Integration testing
17. What is agile architecture?

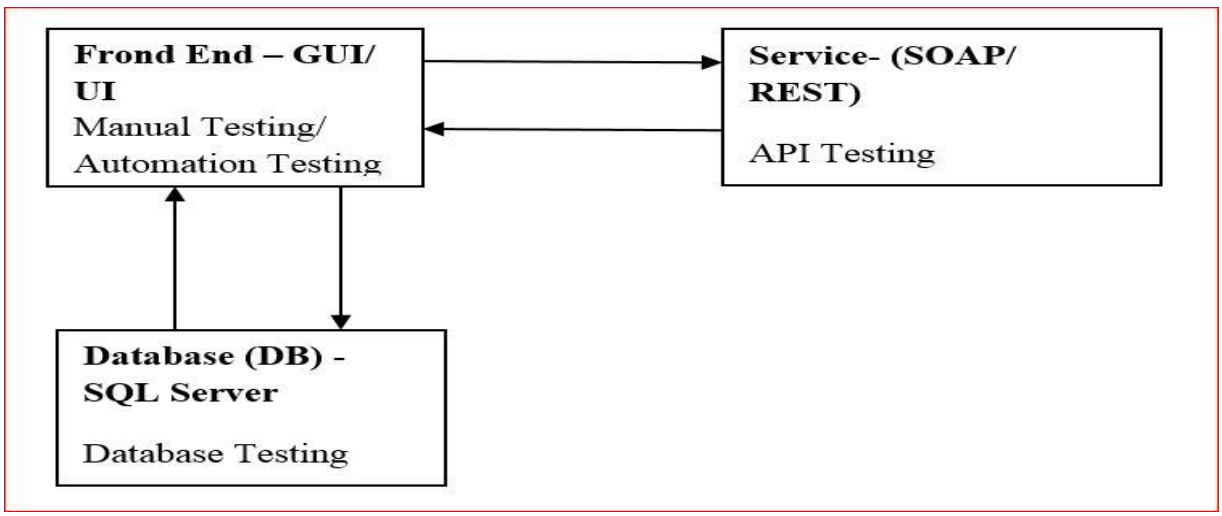
18. What is Agile & what is Sprint?
19. What things you have discuses in last Retrospective meeting?

Answer-

1. Last Retrospective meeting- In current sprint what good and what bad has been be happens that has been discussed.
2. In last Retrospective meeting, I have **given suggestion about**; developer will **add/ mentation date** of every build that will be related to testing.
3. In last sprint I have found more defects due to deployment process (**Dev environment to SIT environment**), Developer will check the deployment process
4. In last sprint I have found more defects due to developer not doing the Unit Testing documents. Suggestion I have given, developer will do unit testing properly.

Project / Application Technology

- Different project technology
  1. Front end- different language use – dot net languages / Java/ HTML/XML/Java Script….etc.
  2. Database – SQL
  3. Web service / API – Java language



Database Testing– SQL – Oracal 11g (Database)

Web service Testing – SoapUI tool

API Testing- Postman tool

Defect / bug tracking tool / project management tool– JIRA tool (aware about HPALM)

Front end- Manual testing / Automation testing

Automation – Selenium tool / Java language

Environments

1. **There are total 4 environments in any project**

DIT- only developer are working – dev

SIT – only tester are working – qa / non preprod

UAT- client / PO / SM / all team – preprod

Prod – Live

2. DBA (database administrative team)-(experience developer) or build team creates these **4 testing environment**

1. **DIT (development integration testing)**

- In this DIT developers are involved & developer develop code according to the user stories / functional requirement
- Having completed coding, they perform unit testing & integration testing
- Having completed the testing, code is pushed on particular location of the server (Github)
- The code is dragged by DBA team & they prepare build (Url) for that code
- Then code is deployed on the build (Url)
- Ex. Kite- registration - login (build)– 2 FA – Watchlist (build)
- Dev Url- https://dev.kite.zerodha.com/

2. **SIT (system integration testing)**

- Having deployed code on build, we get system generated mail as well as mail from development team / leader on JIRA/HPALM
- Mail contains
    1. Build is successful
    2. Build- https://qa.kite.zerodha.com/
       https://nonpreprod.kite.zerodha.com/

3. Credentials / test data – UN- Mnagesh / PW- Mangesh123
4. Design document
5. Unit testing document
6. Build version
7. Build release

- Test execution come in to picture whether we perform sanity/smoke/system & functional testing / retesting / regression testing etc.
- If defect found, we raise the defect to the developer
- Having raise defect, we send mail to the developer on JIRA

UAT (user acceptance testing)

- DBA team create UAT environment
- Stake holder , PO/SM/dev team / test team sit together & check entire functionality of the application step by step from start to end
- If it is defect free application, then it is send to the production / live
- Ex. https://preprod.kite.zerodha.com/ https://uat.kite.zerodha.com/

Prod / live

https://kite.zerodha.com/

Build create- job create

Which tool use for deployment purpose?

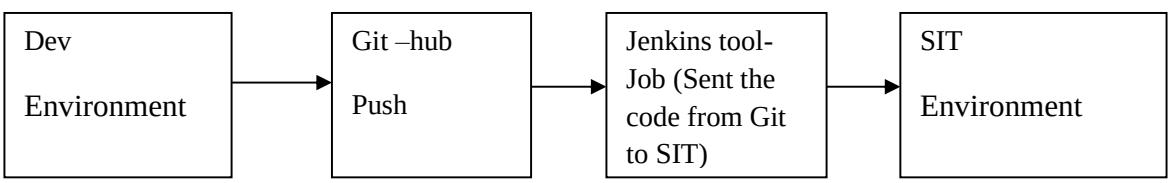
In my organization, we are using "Jenkins tool" for deployment

Dev	Git-hub	DBA	Jenkins tool-job	Sit
Env.	push	Drag		env.

- **How do you open or test application?**

  I work in SIT environment & we have SIT address & Tested SIT environment

- In My organization, we are using "**Jenkins tool**" for deployment

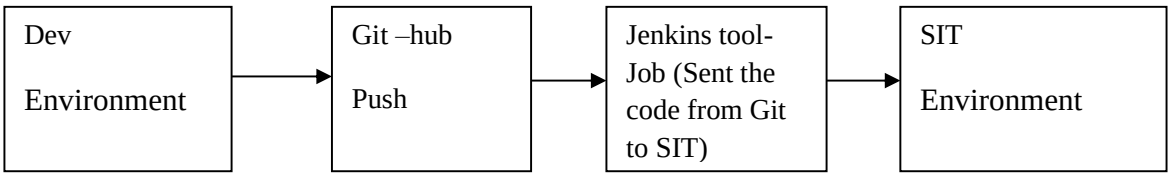


- **Dev Environments**- Unit Testing, Integration Testing
- **SIT Environments**- Sanity, Smoke Testing, System & function testing, retesting & regression testing
- **UAT Environments**- Alpha & Beta Testing

Interview question-

1. What are different technologies present in your project?
2. What are different environments in your project?
3. What is the URL for SIT environment? How you're opening the application?
4. How you get the build?

- **Answer- 1. When** Developer will complete the coding then he will sent build to SIT environment.
- Developer will sent attachment – unit testing doc, URL, tables names
- In My organization, we are using "**Jenkins tool**" for deployment



8. What is your approach if SIT environments are not present? OR if both developer & tester are working in same environments?

- **Answer-**Developer & tester are working in same environment. Some of these defect we will missed in testing.
- **If SIT environments is not present**, we will try to test the application in Dev **& UAT environment (remotes desktop access we will do testing)**

Types of the project

1. **Traditional project**

   In the traditional project – developing & testing is done in the same company or organization

2. **Off the shelf project**

   In the off the shelf project- development & testing team are from different company (i.e. developing is done some where & testing is done some where)

3. **Maintenance project**

   After delivery – include technical & non- technical support after the delivery of the project

---

Error, defect, bug, issue

Error

- A mistake in programming / coding is known as error
- Ex. loop is not closed, missing develop the particular functionality

Defect

- If error found by tester is known as defect
- Ex. logo is not display, alignment, font, color, functionality not working proper

Bug

- Tester raise a defect & assign to the developer, if the defect is valid & it is accepted by developer then it is called as bug
- Ex. mobile number field does not have length validation

Issue

- If application does not meet functional requirement is knows as issue
- i.e. when we raise the defect & assign to the developer – developer understand the issue but does not get any root cause

---

Defect density, defect removal efficiency, defect leakage, defect rejection ratio, defect age

Defect density

- No of defect identified per requirement / US
- No of defect found / size (US/no of req.)
  Ex. Us 01

Defect 20
Us-05
Defect 20

20/5= 4

Defect removal efficiency – refer V model

- A/(A+B)
- (fixed defect /(fixed defect + missed defect))*100
  A= defect identified during testing = fixed defect
  B= defect identified by the client /UAT= missed defect

Defect leakage

- No of defect found in UAT
- (No of defect found in UAT / No of defect found in testing)*100
  Ex.
  (10/90)*100
  100/9= 11.11%

  (50/90)*100= 500/9=55.55%

Defect rejection ratio

- (No of defect rejected / total no of defect lock)*100
  Ex.
  (10/50)*100
  100/5=20%

Defect age – 2 days

- Fixed date – reported date

Defect assign/report developer – 9/3/22

Defect Fixed – retest – 11/3/22

Testing

Development Environment – DIT

1. Unit testing
2. Integration testing

Testing Environment – SIT

- Sanity testing / smoke testing
- System & functional testing (BBT)- 17 Testing
- Retesting
- Regression testing

UAT environment

- Alpha testing
- Beta testing

Prod/Live environment- End user / production issue

Development Environment – DIT

- In dev env. developer is working or involved
- When developer will done the coding, then developer will perform testing like unit testing & integration testing

1. Unit testing

- Unit testing will perform by developer on every user story after coding
- Every user story against, developer will perform the unit testing
- For every user story check the coding is correct or not in the positive way only
- Unit testing contains- document, step to execution, flow of application, snapshot, table, login credentials
- Unit testing it will performed in sub module
- Ex. paytm – recharge module
  Recharge module- main module
  1 US – recharge icon
  2 US- radio button
  3 Us- mobile number
  4 US- operator
  5 US- amount
  6 US- check box
  7 US- recharge now
  8 US- thank you

2. Integration testing

- Integration testing perform by developer
- This testing is carried out after the unit testing
- Integration – means combine all sub module & prepare main module
- In integration – all the dependent module are integrated / added together to form one module
- For integration testing, developer should have knowledge about **functionalities, dependencies on other sub module, relation** because output of on module acts as a input to the other module
- **Integration testing –** it is process to check completeness & correctness of the flow of functionality whenever integration of module performed
- **Integration testing there are 2 types**
  1. Front end integration
     Combine all the dependent module by using CALL function
  2. Back end integration
     Combines all the tables together in database by using the JOIN function
- **Different approaches for integration testing**
  1. Top down approach
  2. Bottom up approach
  3. Bidirectional approach

1. Top down approach

- In this approach, **main module with functionality is available** for testing but s**ub module are not available** for testing
- Sub module are not available because
  1. It can be under the development
  2. Sub modules have defect/error
- So, to test this main module, developer creates dummy module i.e. developer creates dummy program in XML language is known as **stub program**
- XML (extensible markup language)is code language, which is used to communicate between two application
- XML language has request & response

Ex.

Email Login page
Email id
PW
Login

STUB

Email id
PW
Login

Is not available / under development

3. Bottom up approach

- In this approach, **sub module with functionality are available** but **main module is not available** for testing
- To test sub module, developer creates dummy main module i.e. developer create dummy program in XML language is known as **driver program**

- Ex.
  Email Login page
  Email id
  PW
  Login

  driver

  Email id
  PW
  Login

  Is not available / under development

4. Bidirectional approach

- It's a **combination of top down & bottom up approach**
- If developer wants to check functionality of the main module & he does not have sub module then he uses stub program
- If developer wants to check functionality of the sub module & he does not have main module in such cases he use the driver program

Driver / Stub

Email login page

Email id
PW
Login

SIT Environment

- What is ABC testing?
- Why we are doing these testing?
- Are you preparing any document? (TCD/TCE/Defect)
- When developer will the build – inform to tester throw mail (JIRA) & attached in mail – Unit testing document

- In SIT environment, **tester is working**

- Tester will do TCD/Review/TCE/defect rise/ inform to developer/ retest etc.

- SIT environment different **types of testing**

1. Sanity / smoke testing
2. System & functional testing (BBT)- 17
3. Re testing
4. Regression testing etc.

Sanity Testing

- Sanity testing it is a **first testing in SIT env.**
- In sanity testing, **tester are involved**
- When tester will performed first type of testing in the SIT env. , these testing are called as zero level testing / level zero testing
- Sanity testing is also called as
  1. **Zero level testing**
  2. **Tester acceptance testing**
  3. **Build verification testing**
- When tester will got Now build from developer (DBA) then tester will test / validate / check the **build stability** i.e. either **build is stable** for testing further these type of testing is called as sanity testing
- Sanity testing as a tester to check build is **stable or unstable**
- Main agenda is to check **basic and core functionality of the application**
- To check only **main flow of application** / **working flow/ happy flow**
- In sanity testing we will performed / **validate / check**
  1. **Basic & core functionality validation**
  2. **Tab validation**
  3. **Link validation**
  4. **Page validation**
  5. **GUI(graphical user interface) validation**

1. **Basic & core functionality validation**

- We check main flow of the application form start to end or page to page i.e. we check happy flow
- We check for **blocker / show stopper**, if we found, we raise that blocker / defect lock & assign to the developer on JIRA
- If we found defect, these defects are minor, small, critical we **note down those defect**
- If we found blocker, we **reproduce it 2 to 3 times** before lock/assigning/raising the defect
- Ex. Paytm- Recharge module – recharge for every mobile number – in this test- validate user can successfully recharge or not

2. Tab validation (Text box/ Text field)

- We check functionality of the tab, where we can enter char, digit, and special symbol….etc.
- We check whether this text box is accepting it or not
- Whether we enter any value in the tab by using screen keyboard or physical keyboard, those char, digit, special symbol etc…..

3. Link validation

- In this validation, sequence of interlink pages are tested
- Ex. if I click on the flight, then flight information page should be open, developer should provide the link of that page to the icon

4. Page validation

- Page validation is nothing but navigation validation
- We check, can we navigate from one page to other page
- We click on next or back arrow
- We check Page should navigate front & back
- We check pagination

5. GUI validation

- This testing test the interface with which user interact directly
- In this test tester check
    1. Display of application
    2. Logo & image – whether it is clear or blur
    3. Alignment
    4. **Dropdown behavior**
    5. Resolution
    6. Font
    7. Color
- This validation of visualization is called GUI validation

- When we performing these sanity testing, if we found blocker / show topper defect in the application, then tester will **reject the build**

- When we performed these sanity testing, if we found the defect (buggy build) then we simply **reject the build (ex. we found more than 20 to 25 defect)**

- After sanity testing we decide whether **build is stable or unstable**

- If we will **reject the build then we will inform to the developer throw mail** (outllook) (JIRA/HPALM)

- Then **developer will fix the blocker / defect** & sent us a **new build**, then tester will perform **sanity testing**

- In sanity testing, tester will required only **2 to 4 hr** for the testing

- In sanity testing, tester are **not writing the test cases & cant execute the cases**

- **Ex.**

  Paytm- recharge module- US – developer will coding / preparing the new build (V1.0)- developer will sent for testing- tester will do sanity testing (check build stability)- if core functionality not working – if we found a blocker – tester directly reject build(V1.0)- inform to developer by sent a

mail- developer will fix the defect & prepared new build (V1.1)- new build V1.1 will sent for testing- tester will do again sanity testing new build (V1.1)

Build – Friday – afternoon 1pm- 3pm testing – report sanity testing
Build stable- further testing perform – Monday
Build unstable- developer – correct- Monday- morning- new build for testing
Build stable- further testing perform

Mail

- Sanity testing has been performed successfully & these are my observation
- During sanity testing, I have found blocker
- I have raised that defect & I have assigned it to the developer alos

  Defect ID-
  Comment-
  Expected behavior
  Actual behavior
  Screenshots

- On this report, actually decide, next testing has to continue or not
- If we reject the build, then SM/PO- DL/TL/developer/tester – sit together & discuss the blocker

---

Smoke testing

-Smoke testing it is advance version version of sanity testing

- In sanity testing- if we found a blocker / defect then tester will reject the build

- Smoke testing- if developer will sent us a new build then test the build for testing. If we found blocker/defect then we will reject the build but we will **provide the exact root cause of the defect**

- Then developer will fix the blocker & again sent a new build, then tester will perform again smoke testing

- Smoke testing = sanity testing + root cause – Tester

Core functionality
Tab validation
Link validation + root cause
Page validation
GUI validation

- Smoke testing = sanity testing + package validation – developer

- Package validation – package is collection of object
- Tester conduct session with the developer (backend/frontend), then we check package for which parameter is not passing there etc. we get all the response in console & we get root cause of blocker/defect
- In smoke testing, also we will inform to developer (DL/TL/Developer)by sent a mail with root cause
- In smoke testing, tester cant write test cases for smoke testing
- Tester cant execute test cases for smoke testing
- Smoke testing, tester will required 2 to 4 hr.
- **Note- smoke testing – involved tester & developer**

In my project we are performing **sanity or smoke testing**, whenever we get new build

Interview question

- **Difference between Sanity & Smoke testing**

Sanity Testing	**Smoke Testing**
Validation- Basic & core functionality, tab, link, page & GUI	Validation- Basic & core functionality, tab, link, page & GUI and also **find out the exact root cause /troubleshooting and package validation**
Sanity is performed by tester	Smoke is performed by **tester** & package validation is done by **developer**
We do not write test cases & we do not execute test cases	We do not write test cases & we do not execute test cases
If we found defect in sanity we simply **reject the build**	If we found defect in smoke, we **reject the build** but we will **provide the root cause of the defect.**

Interview question-

1. What are your approaches, when you got the build?
2. What is difference between sanity testing & smoke testing?
3. What types of defect you have got in sanity testing/ Smoke testing?
4. Which testing you will perform in SIT Env./Testing, when you got the new builds?

   Answer- In My project, we will performed **Smoke testing or Sanity testing**
5. Are you writing test cases in smoke testing or sanity testing?

- **Answer-** No

6. Are you creating/log defects in smoke testing?
7. When developer is look into these defects, then what you will do? **OR** When you have rejected the build then what you will do?
8. Which testing, you will perform in your project when we get the new build? Why?

   - **Answer**- Smoke Testing or Sanity testing

     Why-

   1. Before going to system & functional testing, we don't get show stopper defect.
   2. It will be **save the time**, for developer if we will inform that where is defect & their troubleshoot.

9. Which testing you have performed in your Origination

- Answer- In **my Origination we performed Smoke testing or Sanity Testing**
- In Smoke Testing, we are checking or validating stability of the build.
- In Smoke Testing, we will check Core functionality, Tab/Page, link validation, GUI/UI, navigation validation, etc.
- In Smoke Testing we required 2 to 4 hr
- If we found defects in smoke testing, then we will reject the build and we will provide the root clause if defects.
- We will send us a mail to developer/DL/TL.

10. If we found defects in smoke Testing then what is your approaches?

- Answer- When we are randomly application **in smoke testing or sanity testing**.
- If we **found defects** then we will **reject the build**.
- Only critical defects are created in project management tool (JIRA/ HPALM) & inform to developer throw the Mail

11. Which testing will follow when you got new build?

a. Answer- In my organization, we are performing **Sanity Testing or Smoke testing**

- When we got **new build form developer** then we perform then **Sanity testing**
- In Sanity testing we are checking

  1. Validating the **GUI/ UI of application**
  2. Validating **core functionality (ex.** Recharge module- Mobile no. recharge)
  3. Validating **the link**
  4. Validating the **tab**
  5. Validating the page/**navigation**

System & Functional Testing

- S & F testing, we will perform after **sanity testing / smoke testing i.e. build stability**
- S & F testing it is also called **Black box testing**
- In S & F testing tester test/check/validate **entire / overall functionality** of the application step by step from start to end
- According to user stories – test scenarios are prepared – test case design prepared – **test cases executed in the S & F testing**
- S & F testing there 4 types of testing
  1. **Usability testing - (90-95%)**
  2. **Functional testing - (90-95%)**
  3. Security testing – (0-5%)
  4. Performance testing – (0-5%) – Jmeter & load runner tool

Functional testing

- Validate application / build **internal & external feature**
- Functional testing 2 types
  1. Functionality testing – Validate application internal feature
  2. Non functionality testing – Validate application external feature

1. Functionality testing

- In this testing, we check **completeness & correctness** of the application functional point of view
- Functionality testing is a process to check internal functionality of the application depends on external interface/ front end / external functionality
- In this testing, we execute the test cases step by step from start to end
- Validate internal functionality application
- It include (BIEBSC)
  1. **B**ehavioral / Behavior coverage testing
  2. **I**nput domain coverage testing
  3. **E**rror handling coverage testing
  4. **B**ackend coverage testing
  5. **S**ervice base coverage testing
  6. **C**alculation base coverage testing

1. **B**ehavioral / Behavior coverage testing

- In this testing, tester validating the behavior of the object / web element
- In behavior, we will validate property of the object
- In this testing, we simply check whether, the object property are working or not

Object	**Property**
Link	Click & unclick / Navigating or not
Check box	Check & uncheck
Button	Click & unclick / enable & disable
Radio button	On & off
Text box	Accept user input / enable & disable

2. **E**rror handling coverage testing

- In this testing, when we enter invalid data / blank data in the object (text box) then system/application is displaying / showing error message or not
- Validating the different types of error message which generated in web page, when we pass the invalid data / blank data
- Ex. Paytm – recharge module
  If we enter blank data - Please enter Mobile Number.
  If we enter invalid data- Please enter a valid Mobile Number.

3. **Backend coverage testing / database testing (details-2week)**

- Validating all **front end operation are stored in database or not**
- Data entered in the front end is stored in the database or not
- As a tester, we validate, whenever data has been entered in the front end has been stored in the database or not
- Validating front end with backend

Ex. frond end – register – successfully register

Backend- store – validate

- We check or validate data can be fetched or not

- So, we simply use the **SQL queries** & we get response in the form of table, where we check whether data is stored or not
- SQL query is used to **fetch data or check stored data**
- Ex. paytm – Recharge module- front end- successful & un successful

  Backend in database data either it is stored or not

  Table name- register

  Select all table

  SELECT*FROM tablename;

  SELECT*FROM Register;

  Output- all table information

4. Service level / base coverage testing

- Being a tester, we validate function sequence of the application
- Validating sequential order of functionality application
- Product owner prepare – **functional flow diagram** & maintain the sequence of module & sub module
- So, in this type testing, sequentially functionality of modules, application are tested as per the functional flow diagram
- Ex.

  Functional flow diagram

  Paytm

  Register

  Login

  Recharge module / icon

  Radio button

  Mobile number

  Operator

  Amount

  Check box

  Click payment

  Thank

5. Calculation base coverage testing

- It’s a part of functionality testing, during calculation base coverage testing, as a tester we validate / check arithmetic operation (i.e. addition, subtraction, multiplication, division)
- Flipcart

  Add to cart 5 item

  1 item price- 100

  2 item price – 100

  3 item price- 100

  4 item price – 100

  5 item price- 100

  ---

  add Total -500

  Remove to cart 1 item

  1 item price- 100

  2 item price – 100

  3 item price- 100

  4 item price – 100

  ---

  sub Total -400

  Same item add to cart – 4 item – 100*4=400

  1 item price- 100

  2 item price – 100

  3 item price- 100

  4 item price – 100

  ---

  mul Total -400

4 quantity – discount 10%

1 item price- 100

2 item price – 100

3 item price- 100

4 item price – 100

---

div Total -400 % 10= 360/-

6. **Input domain coverage testing**

- Validating the input which we are passing in to object (text box)
- Validating what type of input we will pass in object
- Checking what is size or length of the object & what is type of object (i.e. data type)
- Input domain coverage testing maintain
    1. BVA (boundary value analysis)-
       In this we check size or length of input value
    2. ECP (equivalent class partition)
       In this we check data types of input value
    3. Decision table testing technique
       Different input values combination sent- result

Ex. mobile filed – length max & min & data type accept

Ex. **register**

FN- length / size – SRS document – **2-48 accept**

Length – 48- 2 3 4 5 6 7 8 9 10 11……..48 – more time

BVA- Min min+1 min-1 Max max+1 max-1 only check 6 condition - pass-4 fail-2

FN-

Min = 2 = pass

Min-1=2-1=1= fail

Min+1=2+1=3= pass

Max- 48= pass

Max-1=48-1=47=pass

Max+1=48+1=49= fail

ECP- Data type- FN

Char – pass

int/digit- fail

special symbol- fail

space- fail

_- fail

Decision table technique – different combination sent- result – login page

Object	Rule/cond 1	Condition 2	Condition 3	Conditio 4	Condition2 5	Conditio 6	Cond 7
UN	Valid	In valid	Valid	In valid	Blank	Valid	Blank
PW	Valid	Valid	In valid	In valid	Valid	Blank	Blank
Submit	Press	Press	Press	Press	Press	press	press
Result	Home page	Error show	Error show	Error show	Error show	Error show	Error show

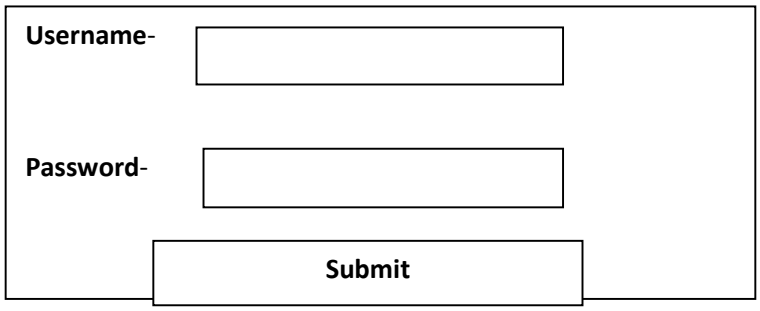
FN- data type accept

A. **BVA (**Boundary value analysis) –

- BVA we are **validating input Size or length into object / web element**
- Min & Max values of input on the objects
- Ex. US- Login Page –

  Username object accept – Mobile no. only

Password- combination of 4 to 6 charter which contains 1 Capital letter, 1 small letter, 1 no & 1 special charter



BVA	**Pass**	**Fail**
Username-	**10 digit no.(**Min & max)	11 digits, 9 digits no.(max+1, min-1)
Password-	4 digits no. (Min)	3 digits no (min-1)
	5 digits no. (Min)	3 digits no (min-1)
	6 digits no. (Max)	7 digits no. (Max+1)

B. ECP (Equivalent class partition)-

- Validating **data types of the input** which we will pass into object
- Ex.US- Login Page –

  Username object accept – Mobile no. only

  Password- combination of 4 to 6 charter which contains 1 Capital letter, 1 small letter, 1 no. & 1 special charter

ECP	**Pass**	**Fail**
Username-	Integer (0 to 9)	Charter, Fraction, Binary, etc.
Password-	String (A-Z, a-z, 0 to 9, Special charter)	Null/ Blank

C. Decision table testing techniques –

- Validation **different input combination values** sent into the object then what result
- Ex. US- Login Page –

  Username object accept – Mobile no. only

Password- combination of 4 to 6 charter which contains 1 Capital letter, 1 small letter, 1 no & 1 special charter

Object	Rule/ Condition 1	Rule/ Condition 2	Rule/ Condition 3	Rule/ Condition 4
Username	Valid	In-Valid	Valid	Blank
Password	Valid	Valid	In-Valid	Blank
Submit	press	Press	press	press
Result/ O/P	Home Page	Error message	Error message	Error message

Examples

E.g. Textbox should accept only 4 to 6 characters (take text box from above-pw)

BVA		**ECP**	
Size	**Result**	**Valid**	**Invalid**
Min = 4	Pass	0-9	Space
Max =6	Pass	a-z	_
Min+1 = 5	Pass	A-Z	
Min -1 = 3	Fail	Special character	
Max+1 =7	Fail		
Max-1 = 5	Pass		

Textbox should accept only 4 to 6 characters - ?

BVA		**ECP**	
Size	**Result**	**Valid**	**Invalid**
Min = 4	Pass	a-z	0-9
Max =6	Pass	A-Z	Space
Min+1 = 5	Pass		Symbol
Min -1 = 3	Fail		-_
Max+1 =7	Fail		Special character
Max-1 = 5	Pass		

Mobile Number should accept 10 digits

BVA		ECP	
Size = 10	**Result**	**Valid**	**Invalid**
Min = 10	Pass	0-9	a- z
Max = 10	Pass		A-Z
Min+1 = 11	Fail		Special character
Min -1 = 9	Fail		Symbol
Max+1 = 11	Fail		Space
Max-1 = 9	Fail		-_

Password – Text Box – Should allowed

8 to 14 digits | 1CAP char | 1 Small Char | Special Symbol |No space and _|

BVA		ECP	
Size	**Result**	**Valid**	**Invalid**
Min = 8	Pass	A – Z	Space
Max =14	Pass	a- z	-_
Min+1 = 9	Pass	Special Char	
Min -1 = 7	Fail	0-9	
Max+1 =15	Fail	Symbol	
Max-1 = 13	Pass		

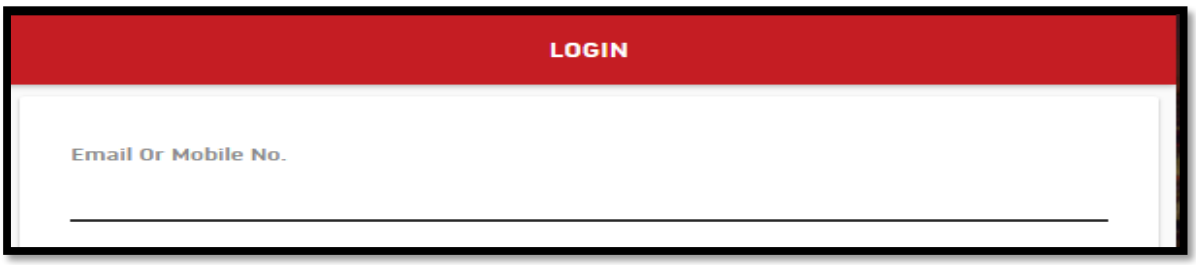
Check BVA & ECP for cycle stand having 100 cycles

BVA		ECP	
Size	**Result**	**Valid**	**Invalid**
Min =1	Pass	0-9	-_
Max =100	Pass		Special Character
Min+1 =2	Pass		Symbol
Min -1 =0	Fail		Space
Max+1 =101	Fail		A – Z
Max-1 =99	Pass		a- z

1. **BVA-**

- Ex. Dram 11 application Login page with mobile no. only

Size/Length	BVA
Min – 10 digits	Pass
Max – 10 digits	Pass
Min-1 = 9 digits	Fail
Max+1 = 11 digits	Fail

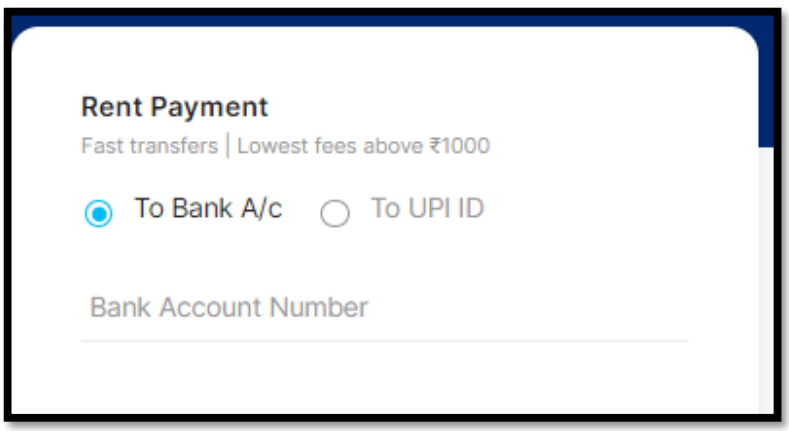


2. **ECP**

- Ex. Dream 11 application Login page

Data types	ECP
Integer	Pass (Mobile no.)
Charter	Pass (Email id)
String	Pass (Email id)

- **Ex.** For BVA & ECP



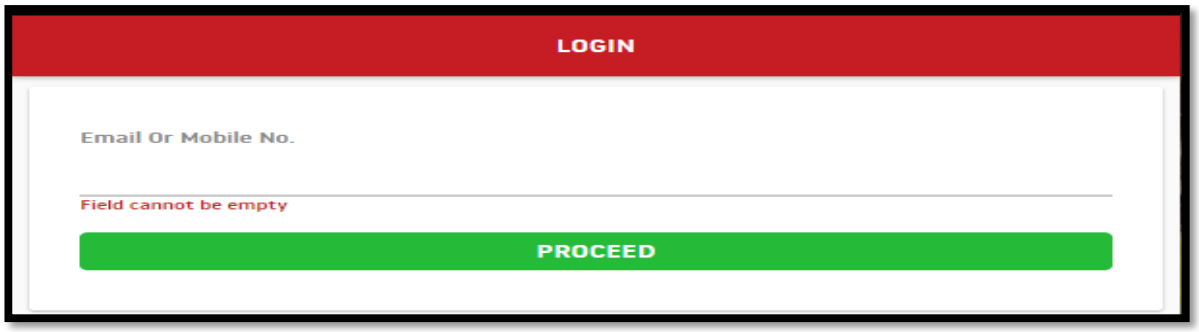
Size & Length	BVA
Max – 30 digits	Pass

Min- 11 digits	Pass
Max+1 digits	Fail
Min- 1 digits	Fail

Data Types	**ECP**
Integer	Pass
Charter	Fail

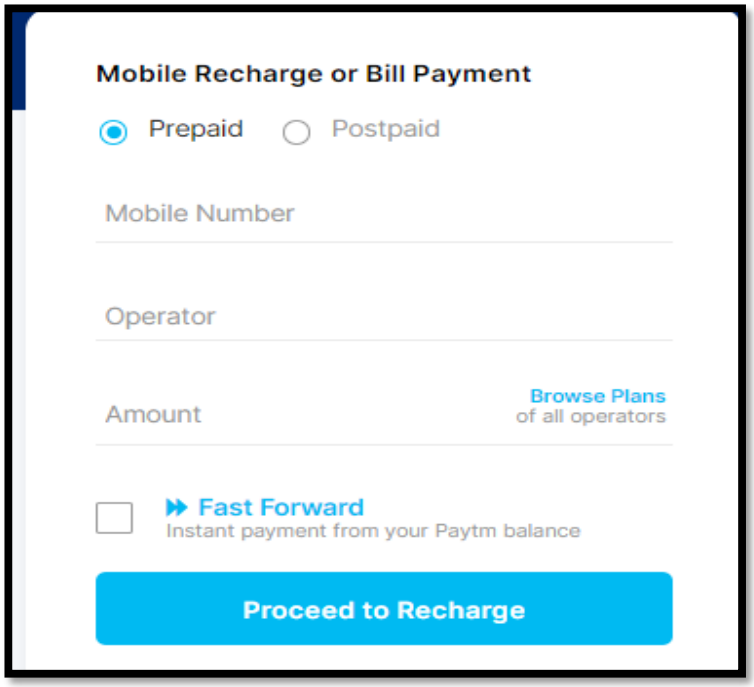
3. **Decision Table testing techniques-**

- Ex. Dream 11 login page



Objects	**Rule 1**	**Rule 2**	**Rule 3**	**Rule 4**
Email or mobile no	**9923475781**	**9422334455**	**xyz@gmail.com**	**pqr@outlook .com**
Result	**Pass**	**Pass**	**Pass**	**Pass**

- Ex. Paytm recharge module



Objects	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
Mobile no	Valid	In – valid	Valid	Valid	In-Valid
Operator	Valid	Valid	In-Valid	Valid	In-Valid
Amount	Valid	Valid	Valid	In-Valid	In-Valid
Result	Pass- Recharge	Fail- Error message	Fail- Error message	Fail- Error message	Fail- Error message

---

Non Functionality Testing

- It's a process of checking external functionalizes or external feature of the application
- Validate external feature of application
- In this testing, we check, whether the application is running on particular operating system (OS) & browsers or not
- Non functionality testing includes (RCCIISPG)
  1. **R**ecovery testing – (S & P)
  2. **C**ompatibility testing (S & P)
  3. **C**onfiguration testing – Not performed (P)
  4. **I**nstallation testing – Not performed (P)
  5. **I**ntersystem testing – (S & P)
  6. **S**anitation testing – (S & P)
  7. **P**arallel testing – Not performed (P)
  8. **G**lobalization testing – (S & P)

Recovery testing

- It is also called as **reliable / reliability testing**
- In this testing as a tester, we check whether application is **able recover from abnormal situation to normal situation or not**
- **Recovery point / requirements are given by stakeholder/client** (i.e. system should recover from start point or should resume from stopped point)
- Ex. Youtube – Movie download – 2 GB- download – 1200MB – suddenly internet off/ power issue- internet return – download – **start point 1mb** or **resume point 1200mb**
- **Ex. Phonepay- recharge – all info fillup – payment – internet lost – back- start point & resume point – decide – client**

Compatibility testing (ex. user expected platform- windows/ Chrome/safari/IE)

- In this testing, we simply check whether application is **supporting to users expected platform or not**
- We check, application is compatible with **users expected platform or not**
- In compatible testing there are 2 categories
  1. **Software compatibility** – OS support / Browser support
  2. **Hardware compatibility** – Parts, printers & external devices
- **In compatibility testing there 2 types**
  1. Forward compatibility testing
  2. Backward compatibility testing

Forward compatibility testing

Application	OS
Kite	Windows (if we have problem)

-In this testing, if **build/application is correct** or ok but **OS / Browser does not work properly**, then it comes under the FCT
- If there is issue in OS/Browser, then **IT support team**/technical team – work on it or resolve the problem
- In this, we get **less number of defect**

Backward compatibility testing

Application	OS
Kite (problem)	Windows

- If **OS/Bowser is correct / ok** but **build does not work properly** then it comes under BCT

- If there is issue on build, then **DBA/developer** – resolve the problem
- In this, we get **more no of defect**

Compatibility testing include

1. Operating system compatibility testing – we are not involved
2. Browser compatibility testing – **we are involved**

There are 2 types of browser compatibility testing

1. Cross browser compatibility testing (chrome/safari/IE/operamini/firfox/edge)
2. Version compatibility testing (chrome-99/98/97/96)

Cross browser compatibility testing

- In this testing as a tester test the build / **application on different different browser like** chrome, IE, safari, Mozilla, Firefox, operamini, edge etc.
- Validating either application / build is supporting to all browser
- Kite – Registration / Login – different browser – **Chrome/IE/Mozilla**
  Login- 50 test cases –
  Chrome- 30
  IE- 10
  Mozilla-10

Version compatibility testing

- In this testing as a tester test the build on **different different version of same browser**
- Validating either build / application is supporting to one browser with different version
- Ex. Kite – Registration / Login – **Chrome browser- V99/V98/V97**
  IE- latest 3 version
  Mozilla – latest 3 version

Configuration Testing / Hardware testing

- I am not part of this testing, I am involved in service based application / company, but I am aware, how it work. There is separate team, who does the testing
- During this testing, tester test whether application is supporting to different hardware's or not
- Validate of application either support to hardware devices or not
- Hardware – printer (laser printer/dot matrix printer)
- Ex. Paytm – Invoice download click – print icon – hardware (printer support to that paytm application)- print page will display

Installation Testing

- I am not a part of this testing, I am involved in service based application, but I am aware about it also
- It's a process of checking installation of software / application in to existing system of user as per the expected platform

Application / build	Customer expected platform	Set program execution Easy interface Occupied disc space Check uninstallation

Set program execution

- In this, he has to check whether all the setup file are present or not / package has all the files or not

Easy interface

- Installation process should be user friendly, so user can navigate easily so, tester simply check installation process

Disc interface

- Tester checks available disc space also & total disc space

Check uninstallation

- Tester check, whether installed software can uninstall from system or not

Intersystem testing

- It's a process of checking whether our application shares data or information with the other application or not
- Banking domain uses this type testing

Ex.
Withdraw money ATM-
User- account- ICICI – ATM
User – ATM- HDFC/Bank of India/ ICICI/ IDBI

Ex.
Paytm
Electric bill pay – MSCB
Recharge- Jio – Paytm- JIO app all user data –

Flight	Gas	Electricity
		
Recharge	Paytm	Movie
Train	Credit card	rent

Sanitation Testing / Garbage testing

- It is called as garbage testing
- In this testing, we try to identified any extra features in application which are not specified in the client requirement
- During the development, developer sense it, this particular feature should have been here in the development but which is not there in the stakeholder requirement. So even through developer adds that feature during development
- So, during this testing, tester find out extra added feature by developer which is not in the stake holder requirement
- As a tester you can lock / assign / raise the defect for this extra added feature
- Developer can suggest new feature to the client but we need to take permission of product owner. If client can agree on it. Then it will be consider new CR. So in such case tester can't raise / assign this as a defect

Ex.

Client requirement- Recharge page – 10 digit mobile number

Developer – mobile number field – 10 digit mobile number with country code

Tester – find defect – sanitation testing / garbage testing

Kite- Login to kite – req.

Developer – Login to the kite

Globalization testing

- It is a process of checking whether application supports different languages or not
- Globalization testing divided in to 3 types

1. Localization testing – Marathi/tamil/telgu…..etc.
2. International testing – Hindi/ japnis/ jarman…etc.
3. **Global testing – English**

Ex.

Application is support to the different different languages or not

1. Localization testing

- As a tester validate / check / test whether application supports to the local language or not
- Local language- Marathi, tamil, telgu….etc.
- Ex. FB / amazon

2. International testing

- As a tester validate whether application support to the official language of country or not
- International language- hindi/japnis/jarman…..etc.
- Ex. Whatsapp

3. **Global testing**

- As a tester validate whether application support to the global language or not
- Global language- English
- Ex. phone pay / paytm / kite ……etc.

Parallel testing

- I am not a part of this testing, I am involved in service based company. But I am aware about it also
- It is called as comparison testing
- It's a process of checking our product with other product

Ex.
Google pay – TCS
Phone pay – Capgemini
Amazon pay – Accenture

Accenture Company- amazon pay – parallel testing with other product

Usability testing

- In this testing we validate user friendliness of the application as real end user point of view
- It is also called as accessibility testing & GUI/UI testing
- How it look?
- We observe all the UI- font, color, alignment, resolution, visually etc.
- Actually application should take less number of event/steps to complete the task
  So in this testing we validate how many steps this application is taking & as well as how much time this application is taking to complete
- Ex. login page – Kite
  UN-
  PW-
  Login press button – immediately dashboard
- Ex. we enter mobile number – only enter 7 digit – instead of 10 digit - system show error message "please enter 10 digit mobile"

Ex.
User friendly
Recharge module – invalid information fillup – error message show – user friendly

Observe – UI (user interface)- Font, color, alignment

Application – min step – use – user friendly
Application – time

- There are 2 types of usability testing
  1. GUI (graphical user interface) testing
  2. Manual support testing

GUI (graphical user interface) testing

- Validating **look of application**
  We observe / validate – UI- font, color, alignment, resolution….etc.
- Validating **easy to use of application**
  We validate, on one click, next action should be perform immediately
- Validating **speed of interface in application**
  We validate, how quick application responds to the user actions
- **Ex**. Gmail vs Yahoo
  Gmail- easy user friendly interface
  Yahoo- not user friendly interface

Manual support system

- It is process of checking context sensitiveness of user manual input
- It is also called as regular expression testing

Ex.
Mobile – contact list
Manual – Mangesh
Search field- “M” – names start with “M” should be displayed

Ex.
Kite application
Watchlist – Search – TCS
Search field – “T”- names start with “T” should be displayed

Ex.
IRCTC application
Source to destination
Source filed – enter “P”- name start with “P” should be displayed

---

Security testing – (5%)

- We validate, whether application is secured or not
- We validate, whether user information data, operation are secured or protected or not
- Tester & developer are involved
- There are 3 types of testing
    1. Authentication – Tester
       We validate, user is valid / registered / authorized or not

    2. Access control – Tester
       We validate whether register / authorized user has permission to access & perform the specific operation or not

    3. Encryption & decryption
       Developer are involved & they perform it

       Encryption
       It is process of converting normal message in to meaningless message (unreadable format)

Mangesh 123- #######
****
@@@@

Decryption
It is process of converting meaningless message in to its original format

######- Mangesh@123

Performance testing

- In this testing, we check the speed of processing of our build/application
- Application is tested with workload & without workload condition
- Performance of the application measured in terms of speed of processing
- Performance test engineer (separate team) for this testing
- They use load runner & Jmeter tool for this testing
- Still I didn't get chance to work with them
- Types of PT
    1. Load testing
       Check the application ability to perform under the anticipated user load

    2. Stress testing
       Check the application under extreme work load & to see how it handle load

    3. Spike testing
       Check the application to sudden large load / spike in the load generated by user

    4. Scalability testing
       Determine the application effectiveness to support an increase in user load

    5. Volume testing
       Large no of data is populated in the overall application & monitor behavior

    6. Endurance testing
       Check the application can handle the expected load over a long period of time

---

Retesting

Tester – TCD – review – TCE – defect – assign to the developer – developer fix the defect – reassign – tester – retest – working fine – close / not working fine – reassign to developer

- Retesting it is a part of **functional testing**
- Whenever the **developer fixed a bug**, tester will **test the bug fix** is called as **retesting**
- Tester close the bug it is worked otherwise re-open and sent to developer
- Retesting defines **check / validating the same functionality** by **passing multiple test data or**
- Re execution of same functionality with multiple test data to validate functionality of the application is known as **retesting**
- **Ex. paytm- recharge mobile functionality – test data = BSNL, Airtel , JIO, Vodafone, idea…..etc. and different state**
- **Ex. fiplkart- login page- mobile – test data - BSNL, Airtel , JIO, Vodafone, idea…..etc. and different state**
- **Ex. fiplkart- login page- email – test data – gmail, yahoo, outllok…..etc.**
- For retesting we will get test data from PO/developer/database
- PO- provide through mail

- **Retesting is performed only 2 times**

1. Before lock the defect- we do retesting
2. After developer solve / fix defect (after getting fix)- we do retesting

- Before lock / assign the defect, we validate whether the defect is valid or not
- Before raising the defect, we have to check whether its **good defect** or **bad defect**
- While executing test cases if we found defect then we have to execute test cases with multiple test data & still there is defect then it's a **good defect**
- While executing test cases with **only one data** & if we found defect, so it is considered as **bad defect**

- **Process**

  We do retesting (multiple test data)– we got the defect – we lock / assign / raise defect on JIRA to the particular developer – defect id generate to that defect – developer fixed that defect – developer again assign to tester – tester retest the previous failed test case with multiple test data & ensure it work well or not – if it is not working fine then tester again assign to developer

Regression testing (Retesting + side impact check to other module)

- Tester – defect (BSNL)– lock – developer – fix – reassign – tester – retesting – (defect – Recharge mobile- BSNL) – Regression testing (JIO/vodafone/idea/airtel)

- **Developer fix the defect & assign to tester, we get new build & we execute test cases which was earlier failed with multiple test data on the new build or ensure whether defect has been solved or not** as well as we validate the impact of new code on interconnected module, main module & sub module or we simply validate, whether it is hampering other module or not

- In regression testing we validate defect has been working correct or not & there is not side impact on interconnected module

- Regression testing is the process in which we are testing modified build / newly corrected build to ensure that they are working fine & also to check their impact / side effect on other working module

- Ex. paytm- recharge module – mobile no- object- Build (V1.0) (500 line code)- testing – retesting- **Vodafone, idea, Jio, airtel it is working but BSNL is not working**- tester assign that defect to the developer – developer will fixed the defect – **modified / newly** corrected build (V2.0)(550 line code) & send modified build to tester- on modified build, we will perform regression testing (retesting + regression testing(check side impact)) – retesting – BSNL is working or not & check side effect / impact on Vodafone, idea, Jio, airtel

- Regression testing is performed 3 times

  1. During SIT – whenever we found a defect / when we got the defect
  2. After SIT – whenever build is moving from SIT to UAT env.
  3. After UAT- whenever build is moving from UAT to prod env.

1. **During SIT**
   In SIT, we got the defect as well as if CR comes then every time we get the new build. So we have to validate whether due to CR & this defect fix & also validate other functionality hammered or not / impact of CR on other module
   In SIT- defect fix + CR = new build = regression testing (retesting + impact check other module)

2. **After SIT- all fix = all defect = regression testing**
   Check all other module & functionalities are working fine or not

3. **After UAT**
   In UAT, if client found defect then developer has to provide fix of it after that also we have to perform regression testing (retesting + side impact on other module) there
   i.e. in UAT- New suggestion + CR = New fix/build = Regression testing

In regression testing, we create the **regression suite** (bunch of test cases)

1. Failed test cases (ex. BSNL test cases)
   When we got the defect- developer – fix defect – send new build – tester test the failed test case with the multiple test data ensure defect is fixed or not & check side effect of other module

2. High priority test cases / core functionality test cases
   Paytm – recharge mobile module- recharge (BSNL,Vodafone, idea, airtel, JIO)

3. Extra feature test cases (ex. +91- add infront of mobile number)
   If there is CR, module are hampered or not is tested & also execute test cases for that new CR

4. If time permits we will consider remaining test cases (we execute low & medium priority test cases)

- For regression testing we required **2 to 4hr**
- In my project for regression testing we required 2 to 4 hr
  Ex. manual testing – module – 150 test cases – regression suite – 50-60 test cases

- There are 3 types of regression testing

1. **Unit regression testing**
   Testing only the change / modification done by developer

1 2 3 4

A B **C** D

2. **Regional regression testing**
   Testing modified module along with the impacted module
   Impact analysis meeting conducts to identify impacted module with QA & dev.

1 2 3 4

A B **C** D

3. **Full regression testing (80%)**
   Testing the main feature & remaining part of the application
   Ex. dev has done changes in many module, instead of identified impacted module, we perform one round of full regression testing

1 2 3 4

A B **C** D

Difference between retesting & regression testing (retesting + impact check)?

Regression Testing	**Retesting**
Definition	Definition
When we perform regression testing?	When we perform retesting testing?
Based on the project and availability of the resources, regression testing can be carried out parallel with retesting	Priority of retesting is higher that regression tester, so it is carried out before regression testing
During regression testing even the passed test case are executed (failed+passed+extra feature)	During retesting only failed test cases are re-executed
Regression testing is performed to make sure the code changes have not affected existing feature	Re testing is performed to make sure that the defect has been fixed or not

UAT (User Acceptance Testing)

- When tester will **complete the testing in SIT env**. then we will performed **UAT testing**
- UAT testing is also called as CAT **(client acceptance testing)**
- After SIT, during UAT, client conduct the no of session / meetings with us PO,SM, all team are the part of this meeting / session
- **During UAT**
  Whichever testing client does related to their application, related their all requirement this testing is known as UAT
- Collect the feedback from client
- During this UAT testing client verifies-
  1. Whether application is as per the **expected platform or not**
  2. Whether their all requirements have been **full filled or not**
  3. Whether all **new suggestion & CR have been add/ executed or not**
  4. Whether **technical approach is correct or not**
  5. Whether application is **providing desired output or not**
  6. So during UAT client validate / verifies **each & every functionality** or entire / overall functionality of the application step by step from start to end
  7. The **objective** of UAT is to confirm application is **ready for release or not / production**
- Q messenger & webex we use for communication
  Slack, Microsoft teams & hangout is used for internal communication
  Defect assign- JIRA

- **During UAT, we fallow this process**
  1. Test team / tester share desktop between client / PO/SM/all team (dev team/test team)
  2. Client select user story & we have to execute test cases of that user story
  3. We execute test cases with multiple test data given by the client (i.e. business data)
  4. Client validate start to end functionality
  5. If it is as client satisfaction then we update the result – done
  6. Sometime client suggest changes or new CR also
  7. In case during UAT, if client found a defect then developer / tester has to check it's a defect or not in their env. if defect valid tester assign that defect to the developer & developer has to provide fix as soon as possible
  8. No of times UAT happens & after that client updates the result, whether UAT has completed **partially or successfully**
  9. Once successful- **we get sign off** then application / **product is deployed for production**

- In UAT check & validating **system & functional testing**
- There are 2 types in UAT
  1. Alpha testing
  2. **Beta testing**
- In my project we perform the **alpha testing**

1. **Alpha testing**

- It is performed in **service base application**
- In this testing, developer, tester , scrum master , designer, client & PO are **involved**
- This testing is performed in **controlled environment** (both tester & developer are available)
- Where, test team share **desktop between client/PO/SM/TL/DL/all team**. Client select US & we have to execute the test cases with multiple test data given by the client
- In this testing, client verify / validate overall functionality of the application step by step from start to end
- If client found the defect, then developer & tester check defect their own env. Then tester has to reproduce the defect in their env. to ensure whether **defect is valid or not**
- If defect valid, then defect is assign to the developer & then developer has to provide **immediate fix**
- This testing is performed at **development site**
- Client is involved more in alpha testing

2. **Beta testing**

- It is performed in **product base application**
- Testing is carried out as **per the client location**
- This testing is performed in **un-controlled env.**
- **Developer & tester** are **not part of this testing**
- **End users / client tester** are the part of this testing
- After the alpha testing, application will be deployed at client location & **beta testing is performed at client site in un-controlled env.**
- If **defect found** in **client location** then **immediate fix not possible** & **fix is provided** in the **next version**

Difference between alpha testing & beta testing?

Alpha testing	**Beta testing**
Alpha testing for the service based application Ex. paytm,cred etc.	Beta testing for product base application ex. khata book
Alpha testing – both developer & tester available	Beta testing- both developer & tester are not available
If found defect – immediate fixed	If found defect fixed on next version

Client interaction more	End user interaction more
It is performed in controlled environment	It is performed in un-controlled environment
Development site	Client location

I also I have performed the UAT testing
In both env. SIT & UAT we can't work
I have got chance to work UAT team, in another project I have worked in UAT or in same project but in another module

In one project, tester will not work in both env. (SIT & UAT)
I have performed UAT testing, when UAT tester is on leave (1 or 2 month)

UAT testing we can performed by accessing remote desktop

https://dev.kite.zerodha.com

https://qa.kite.zerodha.com

https://uat.kite.zerodha.com

Production issue

- After UAT, regression testing is performed
- After regression testing, application is deployed in to production / live environment
- When client found any defects in the production/live environment is called as **production issue or Hot fix**
- **Production issue occurred due to 2 reason**
  1. If production issues has **missed the defect from your company** these production issue are called **hot fix**
     Client mail to project team or escalation email (penalty apply)

A. **Production issue** – client mail to project team or escalation mail – Tester (SIT) will check / reproduce defect (production issue) in SIT env. – **If issue has been found in SIT env.** – Tester will inform to SM & all team & developer fix the issue ASAP- developer will fix ASAP & tester will test defect – if working fine- developer/DBA will deployed to client (production/live)
(SM ask to tester- clarification mail reply)

B. Production issue – Client mail to project team or escalation mail- Tester (SIT) will check / reproduced defect (production issue) in the SIT env. – **if issue has not found in SIT env.-** Tester will inform SM & all team & developer – defect not occurred or functionality working fine in SIT env. – SM will say to the developer **kindly check the deployment process / environment issue**
(SM- ask to DBA/developer – clarification mail reply)

2. If we found issue in production due to **missed functionality from client side these production issues are called "change in request"**
Client sent mail to project regarding these CR

A. If any CR / RFC from client, it don't impact on deployment or testing or development or production, SM/PO & designer interaction client about CR
1. If impact is more then we will inform to client
2. If impact is less then we will development & do testing and deployed to production

Production Issue

Company	Client
Production issue	Production issue
Hot fix	**CR**
Impact analysis	impact analysis
Developer- coding	developer- coding
Tester- checking production issue	tester – checking new CR
Deployment	deployment
Production	production

Testing terminology

- Monkey testing
- Ad-hoc testing
- Exploratory testing

Monkey testing

- When we have maximum no of test cases but we have less time for execution then we perform monkey testing or
- We have lot of test cases to execute but do not have enough time to do execution of test cases that time we perform monkey testing
- It is called as speed up testing
- It is also called as random testing
- This situation arise when developer find difficulty to solve bug, then that time developer take extra time then that bug is called blocker defect
- When blocker defect comes, that time we do monkey testing
- In this testing we execute
  1. High priority test cases
  2. Basic & core functionality test cases
  3. Positive test cases
- If time permits then we check medium & low priority test cases / negative test cases also
- If we found defect, we inform to developer & try to fix as early as possible, according to that update the result

Other way

- If we have more test cases (100/200 TCD) but we have less time for testing (1/2hr) **or** if we got the build at last moment/ time (1/2hr) or deployment will happens with SIT to UAT or UAT to prod (1/2hr)
- At these situation, we will performed monkey testing
- In monkey testing
  1. High priority test cases execution
  2. Core & basic functionality test case execution
  3. Positive test case execution

- In TCE, we found defect then inform to defect & ASAP fix
- Deploy to client (SIT-UAT) & inform that we have done TCE on only high priority test cases execution / core functionality execution
- But in monkey testing more defect are present (15/20defects) then inform to developer then developer will fixed the defect ASAP but it is not fixed in Friday / last working day
- Saturday & Sunday we will work – developer – will fix the defects & tester – will testing the defect

- Still if a defect is not fixed in Saturday & Sunday- in new sprint / next sprint we create a task or US "**Carried over bug**". Developer will fix defects & tester – will testing defect that sprint / new sprint

Ad-hoc testing

- When we have **knowledge of application** / we are aware about application but, we **don't have test case data** or we have **less test data** to test the application, then we perform ad-hoc testing or
- If we have less test data for testing & we have more knowledge about application / module / functionality, on this situation we will performed / we will fallow ad-hoc testing
- We validate application based on previous experience
- Ex.
  Flipcart
  We have payment tab functionality
  Test data – different type credit card / debit card
  Dummy credit card / debit card (SRS document info- credit card length)

Exploratory testing

- When **we don't have knowledge about application / we are not aware about application** but we have **more test data** (test cases/ sprint backlog) then we perform exploratory testing
- It is also called as **level by level** testing or **step by step testing**
- If we have less knowledge about application & we have more test data or if your teammates are on leave for 1 week & you have to test the application / build / US, these situation we will performed / we will fallow exploratory testing

---

Priority & Severity

- Priority & severity will defined **against the defect**

- Priority
  Defines defect impact on client requirement
  Importance of the defect **w.r.t client requirement**
  Types priority – Highest, high, medium , low, lowest

- Severity
  Defines defect impact on functionality of application/build
  Seriousness of defect **w.r.t functionality of the application**
  Types severity – Critical, High, medium & low

High Priority & High Severity
EX.

1. Login page not working
2. Core functionality is not working
3. Share buy or sell functionality not working
4. Paytm – Recharge – Mobile number filed not accepting mobile number

High Priority & Low Severity
Ex.

1. Client logo is not present in application
2. Home page spelling mistake / share name- Expected-TCS Actual-TSC
3. Functionality text overlapped ex. Yes bank share overlapped to Tata motor share
4. Drop down functionality not display proper – Ex. drown down

Low Priority & High Severity
Ex.

1. Rarely functionality – invoice download
2. Rarely functionality – coin functionality not working
3. Rarely functionality- promo code – is not working

Low Priority & Low Severity
Ex.

1. Spelling mistake in text present in button / link / dropdown
2. Expected-Thank message Actual- Thank you!

Who will decide the priority & severity in your current working project?

- **Severity** – We will be decide by tester
- **Priority** – Initial condition we will decide priority while creating a defect, but in stand up I will inform to SM/PO & they will decide

Interview question-

1. What is system & functional Testing?
2. What is system testing?
3. What is functional Testing?
4. What is different testing, you have performed in your project?
5. Which testing you will perform after getting a defects?
6. What is re-testing & Regression Testing?
7. What is UAT testing? Where you have done UAT Testing?
8. What is Production issue? Have you work/ handled production issue?
9. What are your approaches, if you got production issue?
10. What are your approaches, if you got defects in UAT?
11. What are your approaches, if you have less time for testing & you have more test cases to execute?
12. What are your approaches, if your another tester is on leave & your have do the test cases execution?
13. What is Error, Defect & Bug
14. What are different testing terminology?
15. What is Priority & Severity?
16. Give me the example of is high Priority & Low Severity & Low Priority & High Severity
17. What is ready & done state for US?

- **Answer- Ready –** US is completed for development & Testing ready for deployment.
- **Done-** US is completed deployed to client

18. Who will do the deployment of US?

- **Answer**- In my project, we will do **deployment at every Friday/ Monday**
- Deployment will be done by developer

19. What are your approaches, if you have less time for testing & you have more test cases to execute?

- **Answer-** If we have **more test cases to execute (50 to 60 TCD)** and we have **less time for execution (1 hr to 2hr) OR if we got build in last moment (last Friday) from developer OR When we have to move build from SIT to UAT/Prod and we have less time(1 to 2hr),** on these situation we will performed/ we will **follow Monkey testing approaching**

- **Monkey testing approaching,** we will execute **only high priority test cases/ test cases related to core functionality of application**
- If we **get defects** in these **monkey testing approaches** OR if we **got defects** in testing high priority test cases/ test cases related to core functionality of application
- On these situation, we will **work on Saturday / Sunday**
- Developer, Tester & PM will work on **Saturday / Sunday.** To fix that issue → Test → US deployment to client

20. What is your approach, if we have not completed the development & testing of US and you have missed the delivery time line/ deadline?
    - **Answer- Current sprint (sprint 1) -** 1US- development is completed but testing is reaming. These US will mark/ comments against US as carried forward→ Into Next sprint (Sprint 2) → In **Sprint 2**, we will performed only **testing activity**

21. What is your approach, if a defect has been not fixed in current sprint, what you will do?
    - **Answer-** If defects has not fixed in current sprint, then we will work on **Saturday / Sunday →** We will try to fix these defects
        1. If defects has been **fixed** in **Saturday / Sunday** then tester will test and deployment to client
        2. If defects has been **not fixed in Saturday / Sunday,** then we will moves these defect **into next sprint (Sprint 2) → US** – "**Carried over Bug**"

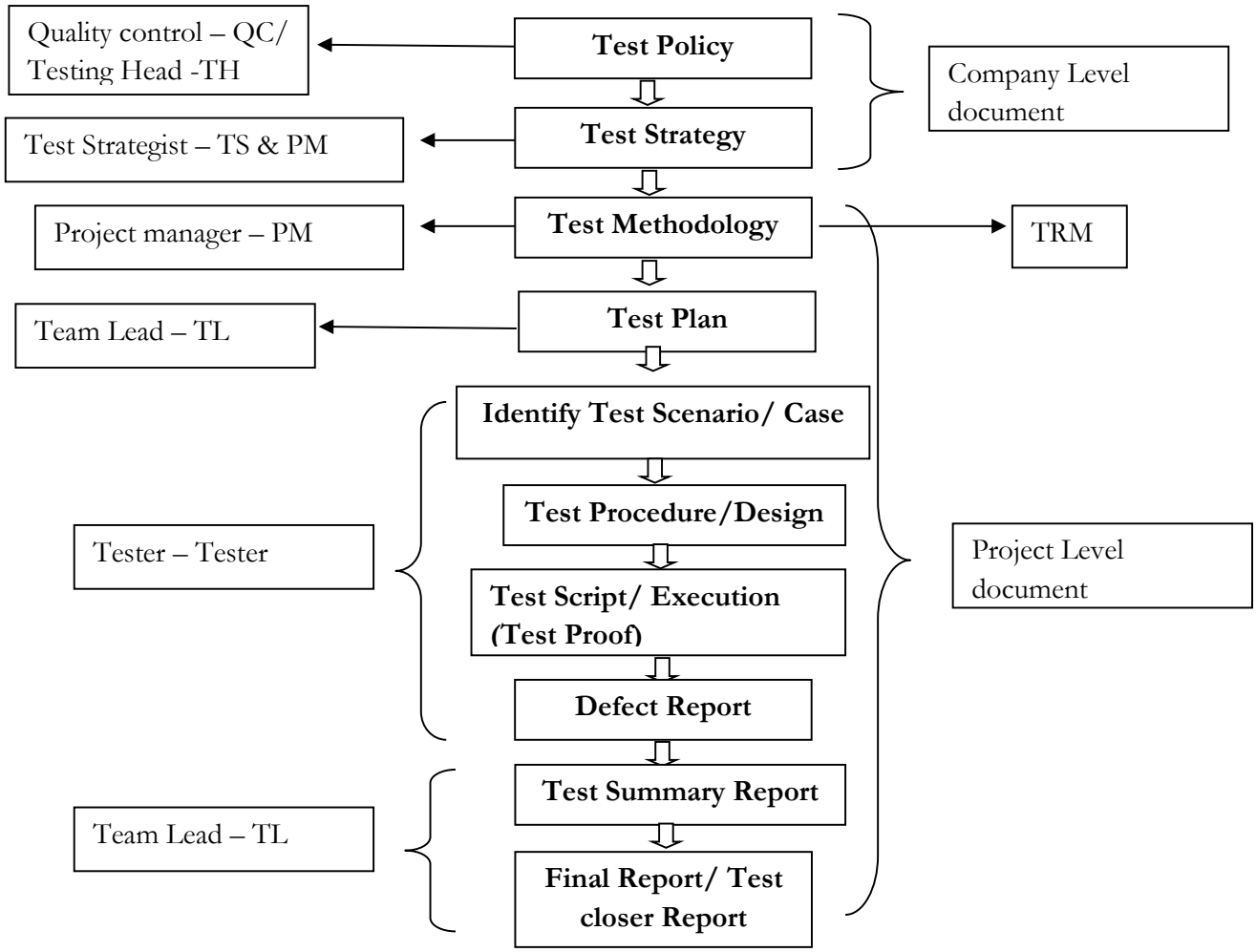
Manual Part 2

1. Organization test documentation
2. STLC
3. Test scenarios / Test case design / excel sheet format
4. Review
5. Tractability matrix
6. Defect life cycle
7. Testing principle

Test Documentation –

Q. What is your organization Test documentation?

- Test Document hierarchy



1. **Test Policy**
   - Test policy document **defines objective of the project / domain of the project**
   - Test policy document **prepared by quality control (QC) / Testing head (TH)**
   - Test policy document **company level document**

2. **Test Strategy**
   - Test strategy defines which **strategy / approaches** we can **apply for full fill the objective of the project**
     Ex. Project- Automation Testing- Java/C#/Python/..etc. + Selenium tool
   - Test strategy documents will be **prepared by Test strategist / Test manager**
   - Test strategy document **company level document**
   - In this document, **testing approach, available resources, & standards** to be followed for testing are given
   - In test policy document **objective of the project** is defined, so to **achieve that objective** & **fulfill objective of the project (Stakeholder requirement & expectation)** we need test strategy document
   - Test strategy document contains different approaches
     1. **Scope & Objective**
        Test Manager (about testing & their purpose)
        Purpose of project (objective & core intention of the project)
        Definition of project (which type of testing or possible testing activities should be conduct)

     2. **Business issue**

        It is nothing but cost analysis & it depends on
        Cost of the project
        Domain of the project
        Resource allocation
        100%- Project cost
        64% - development team & 36%- testing team

     3. **Test responsibility matrix (TRM)**
        It defines- it is a mapping between development stages & testing factors
        It is also called as test matrix
        Note- development stages first mapped with type of project, so form that we get the idea about what will get in the organization (i.e. development, testing, & maintenance)

Development Stage → **Project Type ↓**	**Information gathering/ Analysis**	**Design**	**Coding**	**Testing**	**Maintains**
Traditional	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Off the shelf	No	No	No	**Yes**	No
Maintenance	No	No	No	No	**Yes**

Development Stage → **Testing factor ↓**	**Information gathering/ Analysis**	**Design**	**Coding**	**Testing**	**Maintains**
System & functional Testing	No	No	No	**Yes**	**Yes**
Security	No	No	No	**Yes**	**Yes**
Performance Testing	No	No	No	**Yes**	**Yes**

4. Test deliverable

If defines we can't go to 2nd stage without completing 1st stage

Ex. test policy- test strategy- test methodology- test plan- test case design- test case review- test case execution- defect report- test summary – test closure report…etc.

5. Roles & Responsibility

Names of Job in testing team & their responsibility

R & R allocated based on the experience of the employee, knowledge about domain of the project, priority of module etc….

6. Communication & Status reporting

Require communication & status reporting between two consecutive jobs in testing team

7. Defect reporting & tracking

Required communication between testing team & development team

In short, we can say that, defect reporting & tracking activities are

1. Log the defects- by tester
2. Analysis the defect- by developer
3. Fixing the defect – by developer
4. Retesting of fixes- by tester
5. Regression testing to verify the impact on other module- by tester
6. Defect tracking till closure

8. Implementation of automation

Need of automation in our project level testing

Like manual testing, whether we have to implement it into automation testing or not is decided

If yes, then how much percentage of automation we have required

9. Test measurement (DRE)

DRE- Refer V model

10. Change & configuration management (RFC)

RFC- Refer V model

11. Training plan & training session

Need of training to tester before start of every project testing

No of KT (Knowledge transfer) session should be provided to the fresher & the other tester who come from the other project to this project by the senior tester

12. Risk (Problem) & Mitigation (Solution)

Resource risk- availability of testers are analyzed

Technology risk- software is analyzed

Etc…..

Test Methodology

- Test methodology document will be **prepared by TM/PM**
- Test methodology document it is **project level document**
- This document depends upon the **type of the project** & **project requirement**
- Test methodology = **Project type + Project requirement**
- Test methodology document – **PM will prepared the TRM**
- TRM defines **development stages** mapping with **type of project** / **test factor**
- During the documentation, PM focus on

1. Type of project

Development Stage → **Project Type ↓**	**Information gathering/ Analysis**	**Design**	**Coding**	**Testing**	**Maintains**
Traditional	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Off the shelf	No	No	No	**Yes**	No
Maintenance	No	No	No	No	**Yes**

2. Scope of the project

Identify scope of the application according to the need of project

3. Requirement of the project

Sanity / smoke

Functional

Performance …..etc.

4. Risk in project

New person- KT need to be given

Less person- work overload

If not having knowledge- exploratory testing

5. Preparation of test plan

Job allocation

Resources allocation

Estimation – start & end

6. **TRM finalization**
   Finalize TRM for current project

Test Plan – TL

- Test plan will contains **resource allocation, job allocation, & estimation** etc.
- Test plan document **prepared by Test lead**
- Test plan document it is **a project level document**

Test plan contains

1. **Test plan ID**- Unique number
2. **Introduction**- about project & test team
3. **Test item**- Module/ feature / services / functions
4. **Feature to be tested**- responsible module to prepare test case
5. **Feature not to be tested**- which once & why not?
6. **Approach**- required list of testing technique (Depends on TRM)
7. **Feature pass & fail criteria** – When a feature is pass & When feature is fail
8. **Suspension criteria**- Possible abnormal situation raised during above feature testing
9. **Testing tasks**- Necessary tasks to do before start of every feature testing
10. **Test deliverable**- required test document to prepare during testing
11. **Test env.** – required Hardware & software including testing tool
12. **Staff & training needs**- Names of selected test engineer
13. **Responsibilities**- Work allocation
14. **Schedule**- Date & time
15. **Risk & Mitigation**- problem & solution
16. **Approval** – Signature – **Test manager**

Identify test scenarios & test case design

- Tester will identify test scenario & test case design
- Test scenarios & test case design prepared by tester
- Test scenarios & test case design are project level document

Test case execution & defect report

- In TCE, if we found defect then we will raised to the developer
- Modified build we will performed retesting & regression testing
- Test case execution & defect report are project level document
- Prepared/ done by tester

Test summary report & test closure report

- Test summary report will contains
  TCD-
  TCE-
  Test case skip-
  Defect lock-
  Etc…
- Test summary & closure report done by / prepared by test lead
- Test summary & closure report document are project level document

---

User Story – Description & acceptance – Tester scenarios- Test cases

---

STLC- Software Testing Life Cycle

What is your organization testing process?

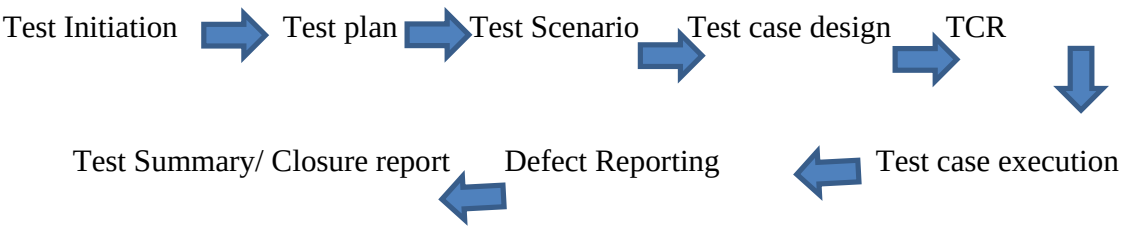
What is STLC?

STLC is a sequence of **different activities performed by the testing team** to **ensure the quality of the software**

It's a **systematic approach to validate application** based on the **stakeholder requirement & expectation**

It describes **step by step process performed by separate testing team** in order to **validate application**

STLC there are different stages



Test Initiation

- In my organization testing process is started with test initiation process
- In test initiation, Test manager are involved
- In test initiation TM focuses on

1. Requirement analysis
2. Scope of project
3. Risk analysis

Requirement analysis

- Requirement of the project means domain of the project
- Ex. banking, healthcare, insurance, telecom….etc.

Scope of project

- In this the strategy will be decided to test the module / functionality of module present in the project

Risk analysis / risk involve in the project

- Risk occur in project are
  1. Less resources
  2. Less test data
  3. Lack of knowledge

Test Plan

- In test plan stage, only **test lead will work** / are involved
- **Test lead** will prepared **test plan**
- These test plan will contain **resources allocation, job allocation & estimation**
- Test plan contains- **what to test, who will test, when to test & how to test?**
- **Job allocation**- means testing are selected in that particular project
- **Resources allocation**- resources means people are allocated for that particular project based on that selected testing
- **Estimation**- means start to end date of project

Test Scenario

- **Having analyzing requirement document** & **assigned user stories** as a **tester we identify test scenarios**
- Test scenarios describes **way to test functionality of an application** / build
- Test scenarios **derived from user story**
- Test scenarios will defines **"What to test?"**
- One test scenarios will **contain / include multiple test cases**
- Test scenarios always defines **"High level test cases"**
- Test scenarios always written in **"+ve ways"**

Test Case Design

- Test cases defines **steps to test functionality of an application**
- Test cases **derived from test scenarios**
- Test cases will defines **"How to test?"**
- Test cases always defines **"Low level test cases"**
- Test cases always written **"+ve way & -ve way"**
- Types of test cases design
    1. **UI test cases** – Font / Color, Design, alignment, resolution
    2. Validation test cases- error handing coverage
    3. **Functional test cases**- each functionality cover
    4. Usability test cases- GUI & Manual Support

Test Cases Review

- Test case review to check the written test cases by tester
- Test case review there are different
    1. Self review – tester check own test cases
    2. Peer review – teammates check the test cases
    3. Internal review – BA / TL check the test cases
    4. External review – Client check the test cases

Test case execution & defect reporting

- Having **review test cases** & when we **get build** we perform **"Sanity testing**" there
- If build **is stable**, we go for **"System & functional testing"**
- As a tester will start the **test case execution**
- So, during the test case execution, if we **found defect** we **assign it to the developer**
- If **we got the fix** then we perform **"retesting"** there & after it, if required we perform "**regression testing**" also
- After it, we have to **prepare test case execution report** & **defect report** & we have to send it to the **test lead**

Test Summary report & Test Closure report

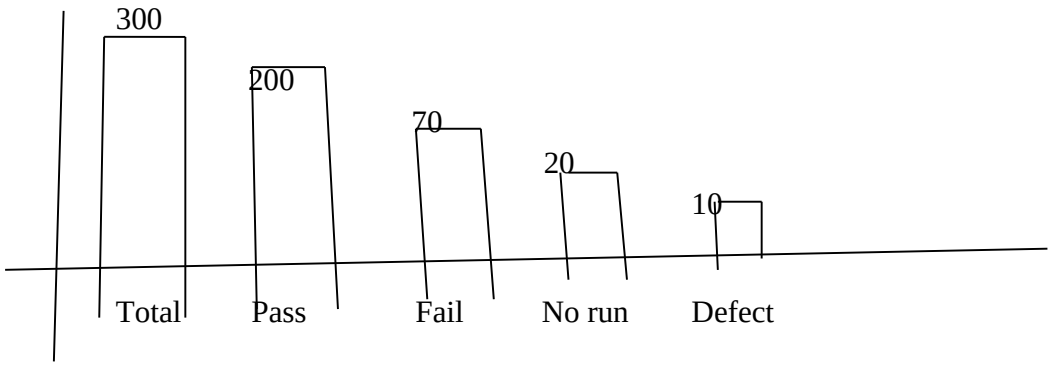
- It is prepared by test lead (sometime tester)
- In test summary report will contain, for every US
  How many test cases we have designed (TCD)?
  How many test cases we have executed (TCE)?
  How many test cases are skipped (TCS)?
  How many defect locked?

For Ex. test summary report

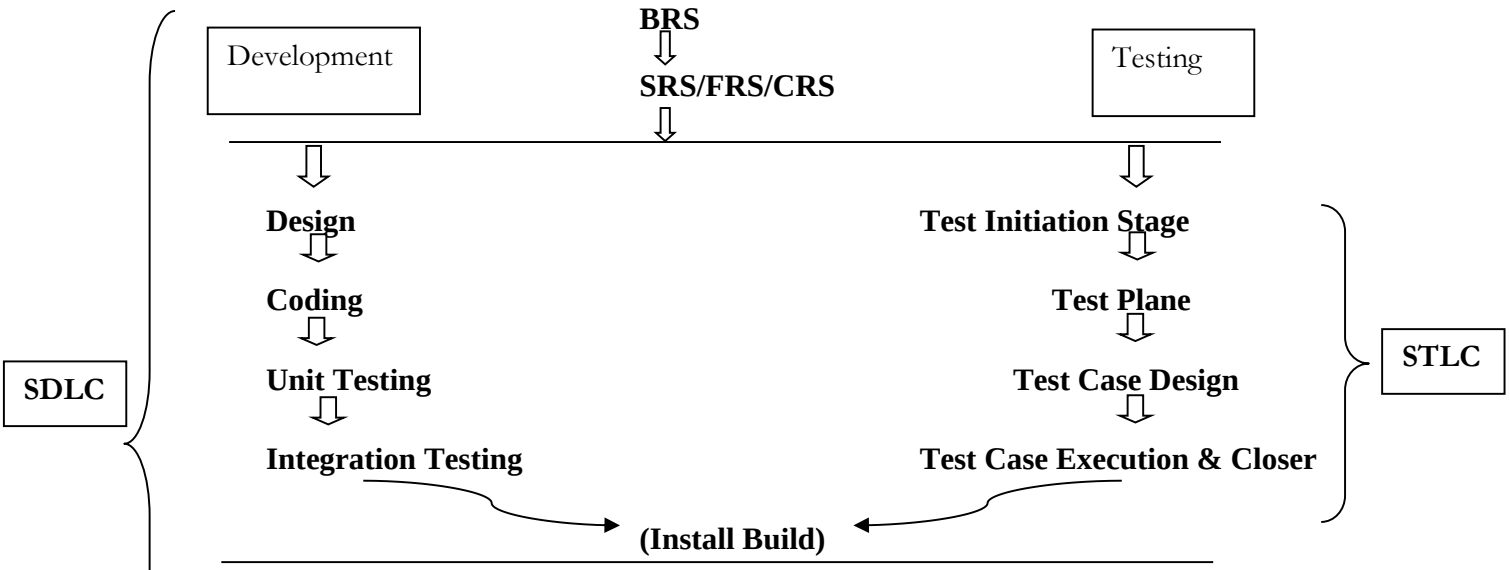
Tester Name: Mangesh	
Build No	
Login Credential	
Browser	
Total no of test cases	500
No of test cases executed	300
No of test cases pending	100
No of test cases pass	275
No of test cases fail	25
No of test cases skip	50
No of defect locked	25

Test closure report

- It is prepared by **test lead**
- Test lead check **all the report** & check all the **process are correct or not as per the** JIRA tool
- Test closure report provide detailed analysis of the **type of testing carried out**, **process followed**, & the **status of the defect found**
- To make the test closure report test lead use the **dash board tool**
- This report is in form of **graph**
- After this test lead prepare test closure report & **sent it to the SM/PO/TM**



Testing Process-



Level 0 Sanity/ **Smoke Testing (**Check the Stability of build)

Level 1 **BBT/ System & function testing (**Internal & External**)**

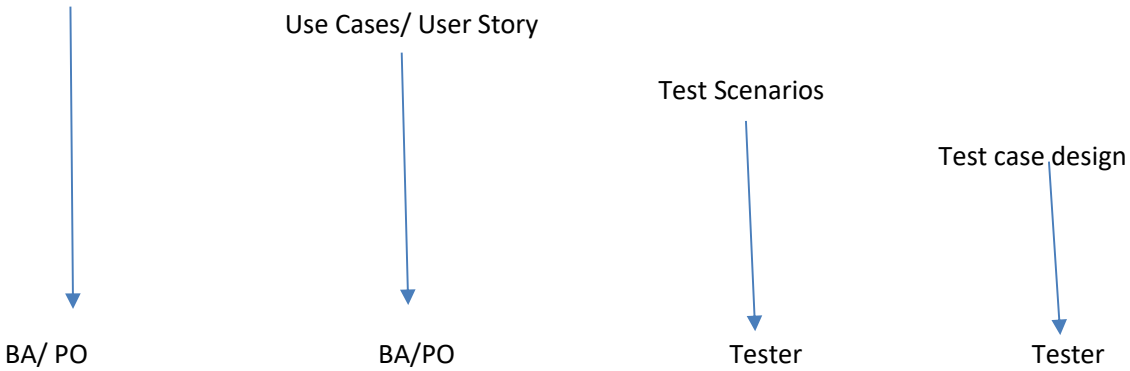
(If found **defect** sent to developer) (Inform throw JIRA/HPALM)

Level 2 **Regression Testing on Modified build**

Level 3 **Final Regression Testing**

Use Cases/ User Story / Test Scenarios / Test case design

SRS/ FRS/ CRS



SRS/FRS/CRS

- SRS/FRS derived from BRS
- SRS prepared by BA/PO
- SRS defines functional requirement to development & system requirement that will be used
  Ex.
  Functional req.- Mobile number filed
  System req.- should accept 10 digit
- SRS will contains
  1. Functional requirement
  2. Functional flow diagram
  3. Use cases
  4. Snapshot
- SRS will contains multiple use cases

Use case / User Story

- Use case / User Story derived from SRS
- Use case / User Story prepared by BA/ PO
- Use case / User Story defines single specific requirement / functional requirement
- Use case / User Story will contains
  1. Description criteria- details about the requirement
  2. Acceptance criteria – does & don't about requirement

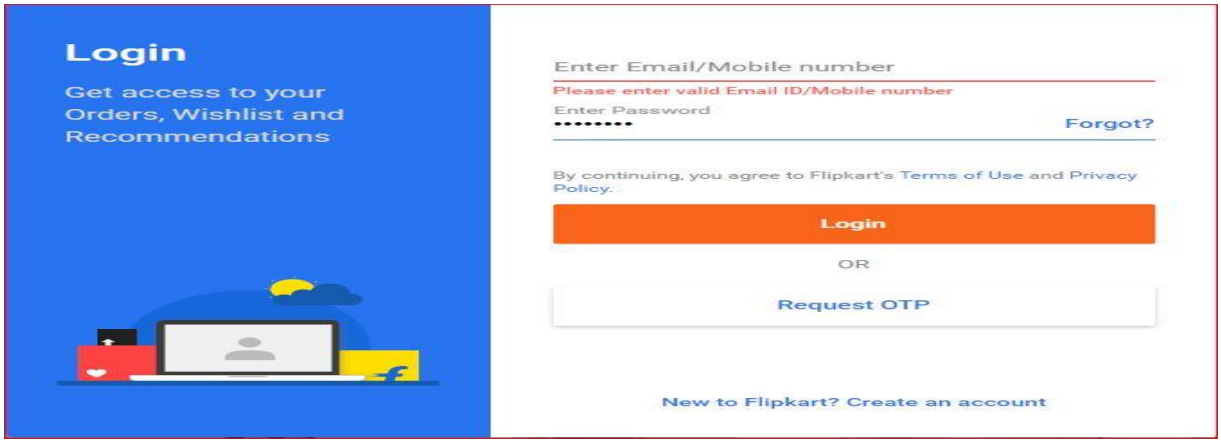
What is difference between Test scenarios & Test cases

Test Scenario

- **Having analyzing requirement document** & **assigned user stories** as a **tester we identify test scenarios**
- Test scenarios describes **way to test functionality of an application** / build
- Test scenarios **derived from user story**
- Test scenarios will defines **"What to test?"**
- One test scenarios will **contain / include multiple test cases**
- Test scenarios always defines **"High level test cases"**
- Test scenarios always written in **"+ve ways"**

Ex.

Login Page



Description Criteria

As a User

I want enter UN & PW to the login page

So that successfully login to the filpkart

Acceptance Criteria

UN – Should accept mobile number & email id

Mobile Number – should accept int, 10 digit | not allowed char, special symbol, space, decimal….etc.

Email id- should accept char, int, special symbol, decimal | not allowed space, _

PW- should accept 1 int, 1 char-capital, 1 char- small, 1 special symbol (@#$)…etc. | not allowed decimal, space…..etc.

Test scenarios for the login page

1. Verify login page by passing mobile no. in UN text box
2. Verify login page by passing email id in UN text box
3. Verify login page by passing data in UN & PW text box
4. Verify the login button functionality in the flipkard login page

Test Case Design

- Test cases defines **steps to test functionality of an application**
- Test cases **derived from test scenarios**
- Test cases will defines **"How to test?"**
- Test cases always defines **"Low level test cases"**
- Test cases always written **"+ve way & -ve way"**
- Types of test cases design
    5. **UI test cases** – Font / Color, Design, alignment, resolution
    6. Validation test cases- error handing coverage
    7. **Functional test cases**- each functionality cover
    8. Usability test cases- GUI & Manual Support

Verify login page by passing mobile no. in UN text box

TCD

1. Verify login page by passing idea mobile number in UN text box
2. Verify login page by passing vodafone mobile number in UN text box
3. Verify login page by passing Airtel mobile number in UN text box
4. Verify login page by passing JIO mobile number in UN text box
5. Verify login page by passing BSNL mobile number in UN text box
6. Verify login page by passing invalid mobile number in UN text box
7. Verify login page by passing blank/null value mobile number in UN text box

Verify login page by passing email id in UN text box

1. Verify login page by passing @gmail.com ids in UN text box
2. Verify login page by passing @redfimail.com ids in UN text box
3. Verify login page by passing @yahoo.com ids in UN text box
4. Verify login page by passing @hotmail.com ids in UN text box
5. Verify login page by passing @outlook.com ids in UN text box
6. Verify login page by passing invalid @gmail.com id in UN text box
7. Verify login page by Passing invalid @redfimail.com id in UN text box
8. Verify login page by passing invalid @yahoo.com ids in UN text box
9. Verify login page by passing invalid @hotmail.com ids in UN text box
10. Verify login page by passing invalid @outlook.com ids in UN text box
11. Verify login page by passing blank / null value in UN text box

Verify Login page by passing data in UN & PW test box

1. Verify login page by passing valid UN & valid PW in UN & PW text box
2. Verify login page by passing invalid UN & valid PW in UN & PW text box
3. Verify login page by passing valid UN & invalid PW in UN & PW text box
4. Verify login page by passing invalid UN & invalid PW in UN & PW text box
5. Verify login page by passing blank data UN & invalid PW in UN & PW text box
6. Verify login page by passing valid UN & blank PW in UN & PW text box
7. Verify login page by passing both blank Un & PW in UN & PW text box

Verify the login button functionality in the flipkard login page

1. Verify login button functionality clickable or not
2. Verify login button mouse over effect or not

UI test cases

1. Verify login page- font, color, alignment…..etc.

Test Case format – Excel sheet fields

- In my project, we are writing text cases in excel sheet
- These are field which in present in excel sheet

1. Test scenario ID
2. Test scenario description
3. Test case ID
4. Test case description
5. Test step
6. Pre- condition
7. Test data
8. Post condition
9. Expected result
10. Actual result
11. Status
12. Executed by
13. Executed date
14. Comments

Or

1. Test case ID
2. Priority
3. References
4. Test Scenarios
5. Pre- request condition
6. Test cases
7. Test step
8. Expected result
9. Actual result
10. Status
11. Comment

Or

Test Case ID

Scenario

Priority

Test case

Precondition

Test Step

Test step description

Expected result

Actual result

Status comment

Or

Test Case ID

Test Scenario ID

Scenario

Priority

Test case

Precondition

Test step

Expected result

Actual result

Status

Comment

Review

Test Case Review

- After the preparation of test cases, we have go through the test case review process to check completeness & correctness of test cases w.r.t. requirement
- Review – to check completeness & correctness of the document
- Importance of test case review-
  Its necessary to review test cases to verify
  1. Whether test cases are written with the intent to detect defect or not
  2. Whether understanding of the requirement is correct or not
  3. Whether impacted areas are identified
  4. Whether positive & negative scenarios/test cases are covered or not
  5. Test requirement coverage is correct or not
- **There are 4 types / ways of test case review**

1. Self review – tester self
2. Peer review – teammates
3. Internal review – BA / TL
4. External review- Client

Self review – tester self

- Tester will do the review of their own test cases these process called as self review
- We have to check our own test document to verify whether all the requirements have been covered or not

Peer review – teammates

- Tester have written test cases

- Test lead/ senior tester will review the tester test cases these process called as peer review
- This review conducted by TL/Senior tester
- We have to send excel sheet to them through the mail

Mail

Hi Roy,

I have written test cases of module “Login” kindly go through it & let me know of any changes are required. Please find attached test case design document of ‘Login’ module

Sent mail

Having review test cases they provide comments / feedback through the mail

Mail-

Hi Mangesh,

You are good enough, now we can arrange the internal review for the same

Or

Add or modify these test cases (mention comment of **test cases- comment column)**

Internal Review- BA

- In internal review, BA/PO will review test cases these process called as internal review
- This review is taken by BA/PO & sometimes with project team
- Test lead send mail invite to project team & PO
- So, we have to share our screen & we have to explain it after that we take feedback from them

Or

- PO will set up meeting with tester, test case review
- In internal review PO is responsible person to review your test cases

Mail

Hi Khushto,

I have a written test cases of module “Login” kindly go through it & let me know if any changes are required. Please find attached test case design document of “login”, module

Sent mail

Having reviewed test cases they provide feedback / comments through the mail

Mail

Hi Mangesh,

You are good enough, now we can arrange the external review for the same

Or

Add or modify these test cases (mention comments of test cases- comment column)

External Review – Client

- In external review, client team will do test cases review these process called as external review
- We conduct this external review when generally project is complicated
- Test lead send mail invite to all project team & client, SM
- PO/SM explain that, we have conducted the internal, peer, self review whatever suggestion we had given, accordingly TC have been updated by the tester

Review 1

Review 2 **excel sheet mentioned**

Review 3

- After this we have to share test case document in front of him.
- Client verify all the req. have been covered or not
- We take feedback, suggestions, comments from client

- After the meeting MOM are shared, which includes
    1. On TC, what we have discussed
    2. feedback, suggestions, comments from client
    3. New modification if any
    4. CR + technical implementation
- After this we have to modify TC accordingly to the feedback, suggestions, comments

Note-

- **In my Project**, we will cover **Internal Review/ External review**
- In review, if **we got suggestion/ Comments/feedback** →Tester will accept the suggestion/ Comments/feedback are **written in excel sheet in suggestion/ Comments/feedback column.**
- For suggestion/ Comments/feedback, **test cases modify & inform to review BA, Client/ UAT team throw Mail.**

Parameter in Test Cases review / while test case review we focus on–

- Test cases review **consider parameters**
    1. Test cases should covered **functionality/ Business logic related**
    2. Test cases is covered **all functionality**
    3. Test cases should be **not Duplicate**
    4. Test cases should be **standard format**
    5. Test cases should be **simple or understandable**
    6. Test cases **spelling mistake**
    7. Test cases **grammar**

Do you take any external review?

Yes, we have taken, actually **my projects was complex/critical**, so **many modifications & new CRs** were there. So, we **decided at organization level** to **review TC documents from the client.** It was essential, after the review, minutes of meetings used to share with us

RTM – Requirement Traceability Matrix / TM- Traceability Matrix

Traceability matrix- (Requirement Traceability matrix)- TM/TRM

- In general we **conduct review** but if **some requirement are missed out** then we are **cover in traceability matrix**
- Traceability matrix also called as **requirement traceability matrix**
- It's a high level document to **map & trace users requirement with test cases** to ensure that for each requirement adequate level of testing is being achieved
- In Traceability matrix- **Test cases & defects are mapped/Link to User story/ requirement**
- Traceability matrix defines **tracing of US, Test case & defect**
- It is document **prepared by tester**
- There are **3 type of Traceability matrix**
  1. Forward Traceability matrix
  2. Backward Traceability matrix
  3. Bi-directional Traceability matrix

1. **Forward Traceability matrix –**
   - Forward traceability matrix is prepared by doing **mapping between business requirement/ US & prepared test cases**
   - In Forward Traceability matrix **test cases (**Excel sheet) **will mapped / link to US/ requirement**
   - **Why we do forward traceability matrix?**
     For a single user story how many test cases I have written and not a single test case should be missed as per the requirement mentioned in the user story

Traceability matrix- (Requirement Traceability matrix)- TM/TRM

- In general we **conduct review** but if **some requirement are missed out** then we are **cover in traceability matrix**
- Traceability matrix also called as **requirement traceability matrix**
- It's a high level document to **map & trace users requirement with test cases** to ensure that for each requirement adequate level of testing is being achieved
- In Traceability matrix- **Test cases & defects are mapped/Link to User story/ requirement**
- Traceability matrix defines **tracing of US, Test case & defect**
- It is document **prepared by tester**
- There are **3 type of Traceability matrix**
    4. Forward Traceability matrix
    5. Backward Traceability matrix
    6. Bi-directional Traceability matrix

2. **Forward Traceability matrix –**
    - Forward traceability matrix is prepared by doing **mapping between business requirement / US & prepared test cases**
    - In Forward Traceability matrix **test cases (**Excel sheet) **will mapped / link to US/ requirement**
    - **Why we do forward traceability matrix?**

      For a single user story how many test cases I have written and not a single test case should be missed as per the requirement mentioned in the user story

3. **Backward / Reverse Traceability matrix –**
    - Backward traceability matrix is prepared by doing **mapping between defect & business requirement**
    - In Reverse Traceability matrix **defects will mapped / link to US/ requirement**
    - In Reverse Traceability matrix - **Created Defects will mapped/Linked US/ requirement**
    - **Why we do backward traceability matrix?**

To know which defect belongs to which user story

4. **Bi-directional Traceability matrix**
    - Bi-directional Traceability matrix it's a **combination of FTM & BTM**
    - In Bi-directional Traceability matrix **test cases (**Excel sheet) **& defects will mapped / link to US/ requirement**

- These **traceability matrix** will **prepared in excel sheet** or in **project management tool** itself (HPALM, JIRA, etc.)

Sample Format- (See excel sheet format also)

Forward Traceability matrix –

Requirement / US ID	**Requirement / US**	**Test case ID**	**Status**
1	Registration	TC01-TC19	TC01-TC13- Pass TC14-Fail TC15-TC-19-Pass
2	Login	TC20-TC27	Requirement is not clear waiting for comments from product owner
3	2FA	TC28-TC33	
4	Watch list	TC34-TC40	
5	Fund	TC41-TC55	

Backward Traceability matrix –

Requirement / US ID	**Requirement / US**	**Defect ID**
1	Registration	PW-01
2	Login	Requirement is not clear waiting for comments from product owner
3	2FA	
4	Watch list	
5	Fund	

Bi-directional Traceability matrix

Requirement / US ID	Requirement / US	Test case ID	Status	Defect ID
1	Registration	TC01-TC19	TC01-TC13- Pass TC14-Fail TC15-TC-19-Pass	PW-01 FRIEND-29
2	Login	TC20-TC27	Requirement is not clear waiting for comments from product owner	
3	2FA	TC28-TC33		
4	Watch list	TC34-TC40		
5	Fund	TC41-TC55		

Defect Life Cycle

Client req- BA collect- BRS/SRS- Sent- Developer develop- Tester – test the App- found defect – Assign defect to developer- developer fix the defect – retesting – its working fine- close / if not working fine – again assign to the developer

- During test case execution, if we identified deviation between expected result to actual result is called as defect

Or

- It is deviation between expected result & actual result when application is under testing is called as defect

Or

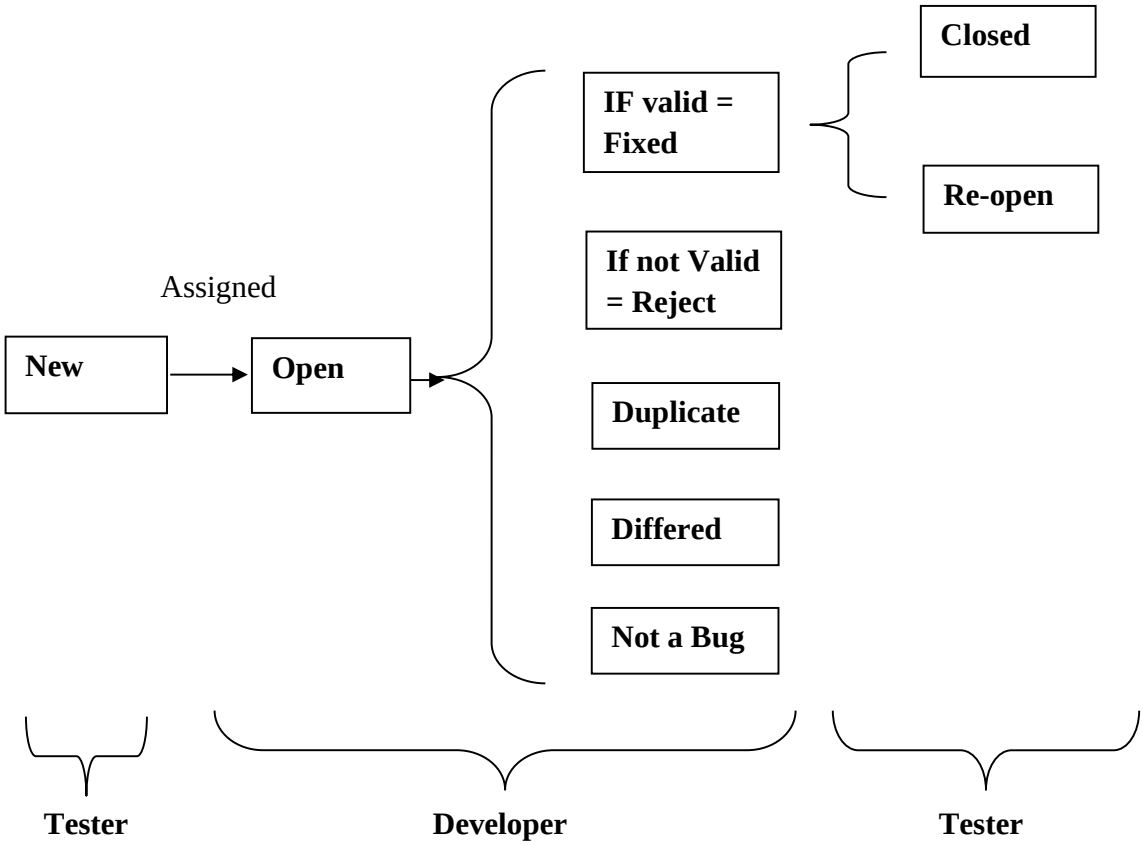
- In application while TCE, if we found error it is called as defects

- Basically defects we generated due to

1. Wrongly implemented functionality
2. Missed some functionality
3. Extra functionality
4. Due to env. problem / deployment problem

Defect Life Cycle

- DLC= reported date – close date
- DLC- the journey of the defects, from start to end it is called as DLC
- DLC contains different statuses- New, Open, Duplicate/rejected/ Not a bug/differed, Fix, Close / reopen



New

- During the testing, if we identified / found a defect, so we assign/lock/ raise that defect in to JIRA / HPALM tool
- It is posted for the first time so status of the defect is “NEW”

Assigned

- Once the defect is posted, we assign that defect to the developer, that times status of the defect is “Assigned”

Open

- In open state, developer analysis the defect & they check whether defect is valid or not
- Developer reproduce the defect in their env. if developer feels that defect is not appropriate then they transfer the defect in to rejected, duplicate, differed, & not a bug state

For

Ex.

1. **Rejected**
   - If defect is invalid or not a genuine they mark it is a rejected
   - Comments
     We are not able to reproduce the defect in our env.

2. **Duplicate**
   - If developer find that, the defect as same any other defect or concept of defect matches any other defect then they mark it is a duplicate
   - Ex.

Paytm- Recharge Module- Mobile number functionality

Defect-

Paytm>Recharge> Mobile number field does not have length validation

Paytm>Recharge> Mobile number field accept less than 10 digit

Paytm>Recharge> Mobile number field accept more than 10 digit

- Comment
  Already defect has been raised for the same functionality they mention the defect ID also

3. Differed
    - The state of the defect is changed to differed means **defect is expected to be fixed in** the next version
    - If a defect is valid but developer dont want to fix that defect in current sprint/ current module then developer will mark status of that defect as differed
    - There can be so many factor for this
        1. Priority of the defect may be low
        2. Lack of time for the release
        3. Impact of the defect on software is low
    - For this PO discuss about defect with client & after that PO/SM sit together & decide we have to fix it in this sprint or in the next sprint

4. Not a bug

- If the defect is not affecting the functionality of the application then it is mentioned as not a bug
- Ex.
  Defect – login
  Defect lock- registration

Fixed

- If defect is valid, then developer analysis the code defect of the application to identify the exact root cause of the defect & modify the code of the application to resolve/solve the defect
- Having fixed the defect, they verify code & after that they mark it is “fixed”
- They provide the proper comment also
    1. What was defect?
    2. What changes have been made?

  So after all these, they assign the defect to us (tester) that time status is “commented”

Close

- So, after this, we have to perform retesting. Whether we execute the test case which was earlier failed with multiple test data to ensure that defect has been solved or not
- **If defect solved**, the status of the defect “verified/approved”, after that defect is checked by PO/TL also & then by taking permission of PO/TL we close the defect. So here status of the defect is “closed”
  Comment

The defect has been raised related to this mobile number functionality, now it's working fine as expected, I am getting the desired output & defect has been resolved

If defect is not solved

Reopen

- If still defect is there, we simply reassign it to the developer that time the status of the defect is reopen
  Comment
  Defect is still there in the functionality, kindly look in to the that defect & we attach screenshots of the defect also

What is your approach when developer will not accept the defect?

- When we found the defect, we lock that defect in the JIRA tool as well as we write detail description there & we attach screenshot & the defect also
- I will give him first remainder in the triage call, second remainder in the daily stand up if still he is not accepting it
- I will reproduce that defect on my system as well as on my colleagues system if defect is present. I will inform the developer again
- I will share my screen & I will reproduce that defect in front of him. Still he is not accepting it. I will inform to test lead, test manager & I will send mail to scrum master, developer & all tester regarding such issue
- In daily stand up we will raise this issue

Testing Principle of Software Testing

There are 7 principles of software testing

1. Testing shows presence of defects
2. Exhaustive testing is not possible
3. Early testing
4. Defect clustering
5. Pesticide paradox
6. Testing is context dependent
7. Absence of error fallacy

Testing shows presence of defects

- Testing an application can present one or more defect exist in the application
  Ex. web application, desktop application, mobile application
- Therefore, it is important to design test cases which find as many defects as possible
- Tester validate application to make sure that the application is defect free
- Primary aim of the testing is to identify test cases with the help of various testing technique & testing should be traceable to the client requirement
- By performing testing we can reduce defects

Exhaustive testing is not possible

- To test all the module & their features with various possible combinations of input & precondition is not possible
- It is not possible to test all possible combination of data

Ex.

Restaurant billing application

GST

50/- 100/- 200/- 500/-1000/-1500/-2000

- So, instead of this, teams, PO/SM should prioritize testing based on the risk analysis to the product & business

Early Testing

- Testing activities should be started in the early stage of SDLC
- It is not necessary to develop the software for the testing
- It is required document only based on this we have to design a test cases

Defect Clustering

- We can detect number of defects, which are correlated to the small number of module. i.e. small number of module contains most of the defects in the software
- So, there various reasons of it
  1. Module can be complicated
  2. Coding can be complex
- According to pareto principle, 80% complication present in 20% of the module

Ex.

Project – defect

Main module- less defect

Sub module - more defect – 80% defect present in the 20% size of module

Interview question

Pesticide Paradox

- If we are executing the same set of test cases again & again then we will not able to identify new defect
- So, to get over this pesticide paradox, we need to review TC frequently & different TC need to be written

Ex.

Application version 1- design some set of test cases

Application version 4- you also use same test cases

Ypu have not got the more defect

Whatsup

Version 1 Updated TC version 4

Android

Lollypop updated TC Marshmallow

Project – 10 module

Version 1 – TCD

Version 2- Same TCD

Testing is context dependent

Ex.

Approach testing for banking, telecom, insurance, gaming

- We have number of domains- banking, insurance, healthcare, e-comerce, telecome, gaming…..etc.
- So there is a define way to test these application & it is based on need, functionality & feature
- So, to validate these application we need to take help of various testing technique, approaches & methods
- Ex.

Testing medical software	testing gaming software
Proper calculation in medical	gaming working on hardware
Reliable calculation	particular RAM
Small decimal point in drugs are	PC is hand or not
Create more problem	
Different approach	different approach

Absence of error fallacy

- Many other factors to be considered before making a decision of the software
- Testing not proper
  1. Test cases upgrade
  2. Software requirement
  3. Test software
- Were the executed test cases really designed to catch the most defect?

Or

- Where they designed to see if the software matched the client requirement?

As tester testing – not defect found / testing is not proper

Developer done coding proper

Test Factors:

To define quality S/W, quality analyst defines 15 testing issues. Test factor or issue means that a testing issue to apply on S/W to achieve quality.

The test factors are:

- Authorization:

  Whether user is valid or not to correct application

- Access Control:

  Authorized to access specific services

- Audit Trail:

  Meta data about user operations

- Continuity of processing:

  Inter process communication (IPC) during execution.

- Correctness:

  Meet customer requirements in terms of inputs and out puts

- Coupling :

  Co-existence with other existing S/W

- Ease of Use :

  User friendliness of screens

- Ease of operate :

  Installation, un installation, dumping, exporting etc.,

- File integrate:

  Creation of internal files (ex. back up)

- Reliability:

  Recover from abnormal situation

- Portable :

  Run on different plat forms

- Performance :

  Speed of processing

- Service Levels :

  Order of services

- Methodology :

  Follow standards

- Maintainable :

  Long time serviceable to customers

Test Factors VS Black Box Testing Techniques

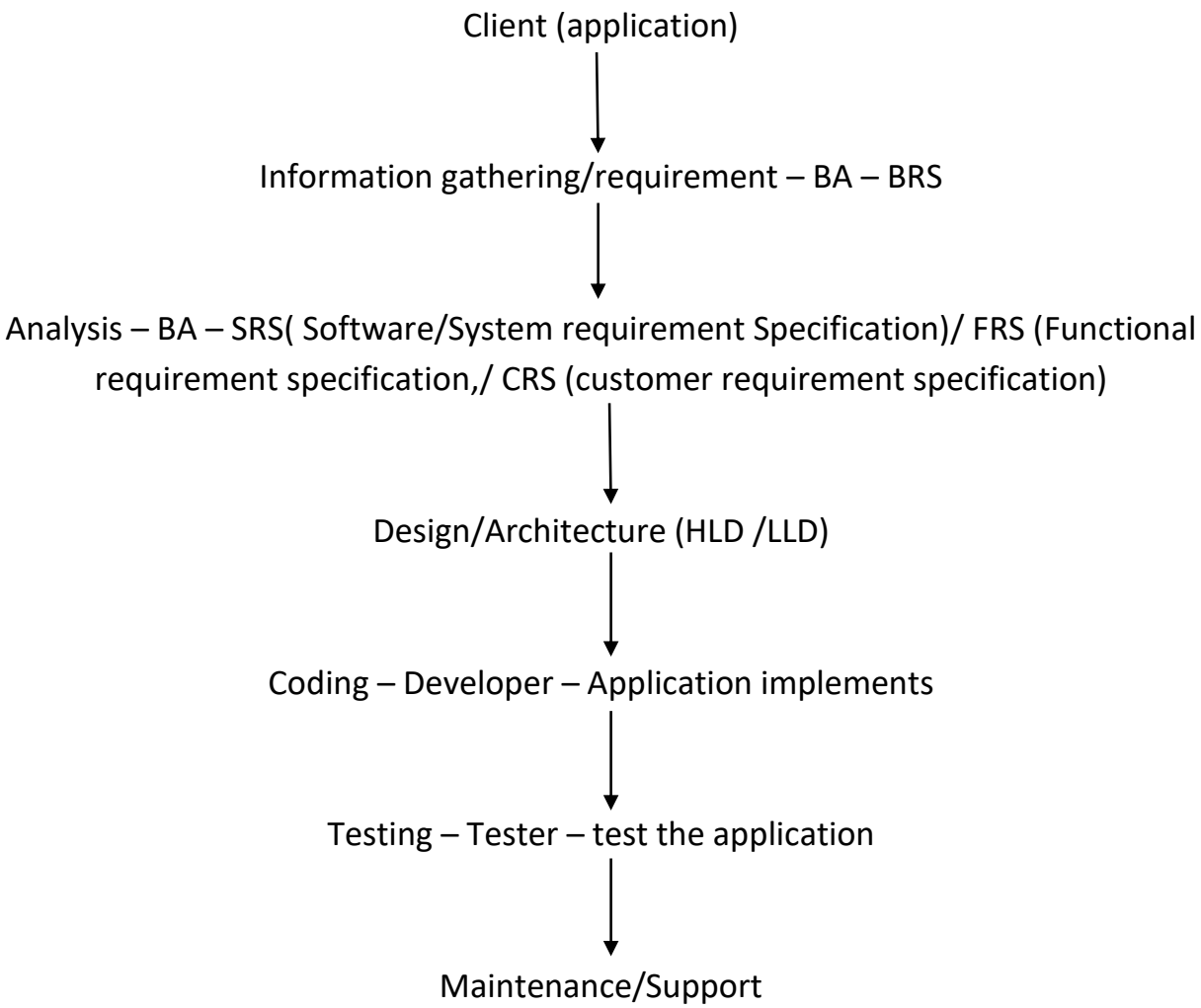
- Authorization : Security Testing, Functional or requirement testing
- Access Control : Security testing, if there is no separate team then Functional or Requirement testing
- Audit Trail : Functionality or requirements, error handling testing
- Correctness : Functionality or requirements testing
- Continuity of processing : Execution testing , operation testing (white Box)
- Coupling : Intersystem testing
- Ease of use : Usability testing
- Ease of operate : Installation testing
- File Integrate : Recovery, error handling testing
- Reliability : Recovery, Stress testing
- Portable: Compatibility, Configuration testing
- Performance: Load & Stress, Storage & Data volume testing
- Service Level : Functionality or requirements testing
- Maintainable: Compliance Testing
- Methodology: Compliance Testing

Manual 1

SDLC (Software development life cycle)

It is a process used by the software industry to design , develop and test the high quality software.

Phases of SDLC



BRS (Business requirement specification)

1 – It is a bridge between client and the BA

2 – Business related document

3 – This document generally consists of complete scope of the project, performance, requirement, and usability.
4 – It is a non technical document

SRS (Software requirement specification)

1 – SRS is derived from BRS
2 – IN SRS document all functional and non functional requirements are covered.
3 – It is a technical document

What is BRS and SRS?

BRS	SRS
Business requirement specification	Software requirement specification
This document generally consists of complete scope of the project, performance, requirement, and usability.	IN SRS document all functional and non functional requirements are covered.
BA people prepares BRS	BA people prepares SRS
From client BA collects the requirements and prepares BRS document	SRS is derived from BRS
All requirements are covered	All functional and non functional requirements are covered
Use cases are not present in BRS	Use cases are present in SRS.

Design – Solution designer
HLD- external functionality of the app
LLD – internal functionality of the app

Coding-

Developers will work on coding part to build a app

Testing

Tester is working on

TCD –Test case design
TCE – test case execution

Maintenance / support

We provide some warranty to client.

SRS document includes

1 – **Functional flow diagram**

Eg. Signup page – Login page – Home page – Profile page

2 – **Functional requirement**

Eg . Signup page (Requirement – First name , last name , mobile number , bdate, mail id, password , submit button.)

3 - **Use cases**

Use cases consist of Acceptance and description criteria.

4 – **Screenshot/Snapshot**

1- Snapshots are visualisation of functionalities before development of product.
2- Snapshot is made by BA using HTMl code and irise software.

Agile model/Methodology/process

1- The agile scrum methodology is combination of both incremental and iterative model for managing product development.
2- Agile is a continous process for development and testing.
3- Agile is value driven process.
4- If CR comes then we will check its impact on development, testing and production or its priority in the application.
5- Duration 2 weeks sprint

Sprint – A sprint is a set period of time during which specific work has to be completed and made ready for review.

I have worked in Scrum Agile methodology

Agile framework/ Sub module

1. **Xp – Extreme programming – continous development**
2. **Scrum – To give sprint wise delivery to client**
3. **FDD – Future driven development**
4. **Kanban**
5. **Lean**

Agile Architecture

Information gathering BRS	Product Backlog[200 req]
Analysis SRS	Sprint Backlog [20 req – 1 sprint] [20 req – 2 sprint] [20 req – 3 sprint]
USE cases / 1 requirement Description Acceptance	User Story/ 1 req
Design	Design
Coding	Coding
Testing	Testing
Maintenance / Support	Maintenance / Support

Peoples involved in Agile

Client - Stakeholder

BA - Product owner

DM - Solution master

PM - Scrum master

Designer - Designer

Dev team - Dev team

Testing team – Testing team

Agile Meetings/Ceremonies

Meeting	Purpose	Involvement of people
Grooming Meeting (Before start of sprint)	To clear the doubts about the requirement.	30min to 1hr Product owner (Chair person),Dev team, testing team, Design team, Scrum master (optional)
Sprint planning Meeting. (1st day of sprint)	1- Decide the user story. 2- Estimation (time slot to complete the user story) 1sprint – 20req	1 hr – Scrum master (chair person), product owner, testing team, dev team , Design team
Daily stand up (Scrum Meeting)	What you have done yesterday? What you are going to do today? What are the roadblocks or issues?	15min -30min Scrum master (Chair person),design, dev team, test team, BA (optional)
Sprint review meeting (end day sprint)	User story completed in testing- demo/review	1hr BA, client, testing team,Dev team, Design, PM.

Sprint retrospective Meeting	Good and the Bad things in Current sprint (discussion about our work)	30min PM, Testing team, dev team, design, BA (optional)

Advantages

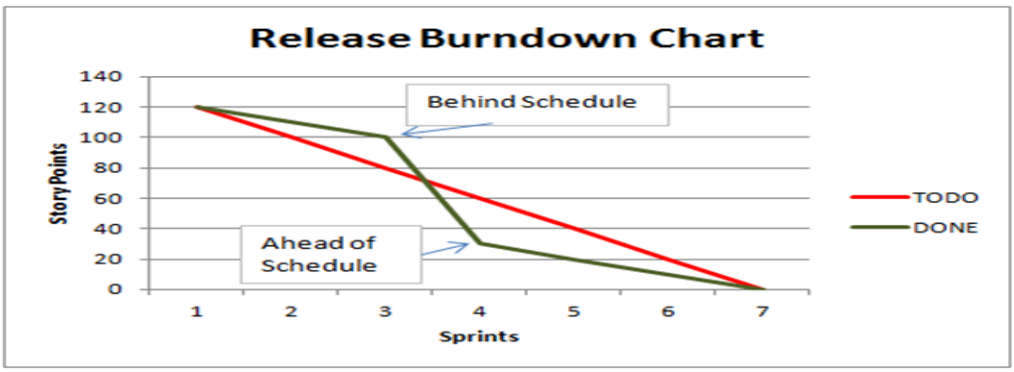
1- Sprint wise delivery
2- Working software is delivered frequently.
3- Change request are accepted immediately.
4- Customers, developers and testers constantly interacts with each other.

Disadvantage

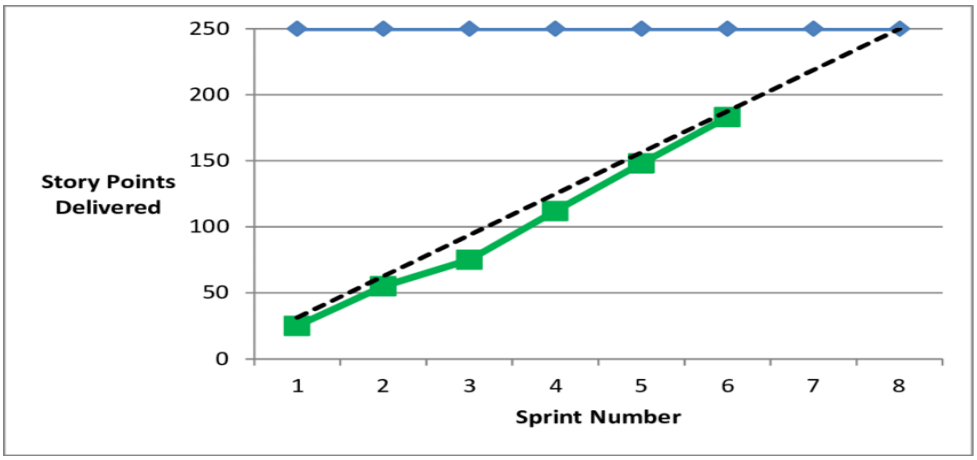
1- If requirement is not clear before the sprint or product owner is not clear about the requirement then the sprint can go out of scope.
2- Less documentation
3- Cost of agile development methodology is slightly more as compared to other development methodology.

Agile terms

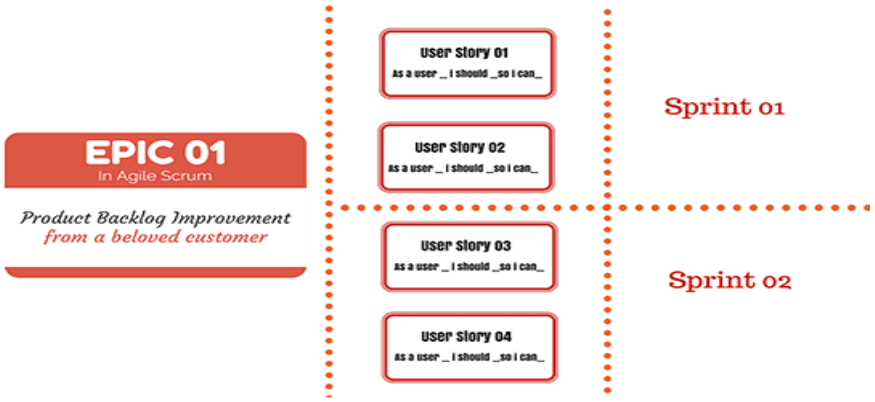
Burn Down Chart – How much user story remaining w.r.t to time.



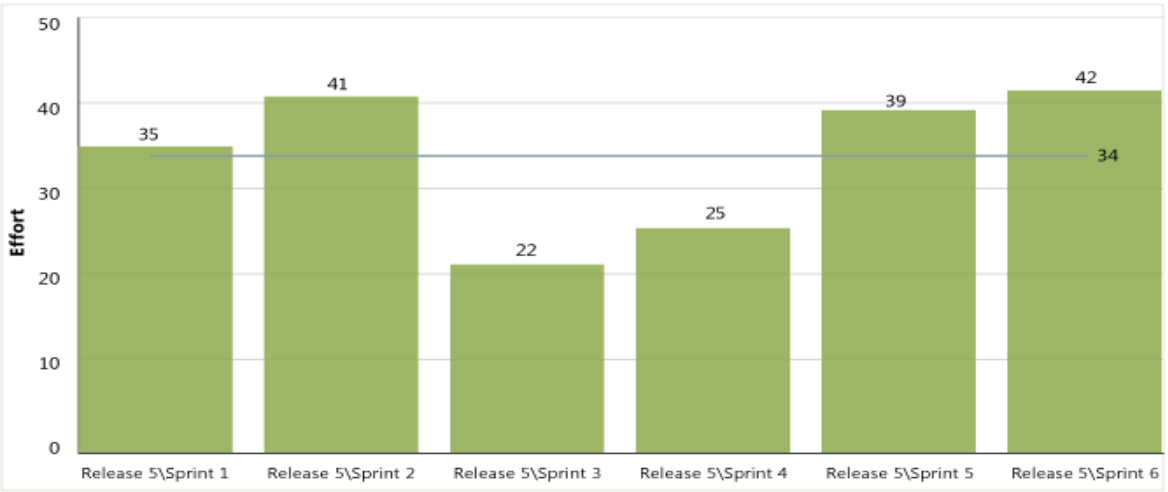
Burn up Chart – How much user story completed w.r.t time



Epic – It is User stories present in single sprint.



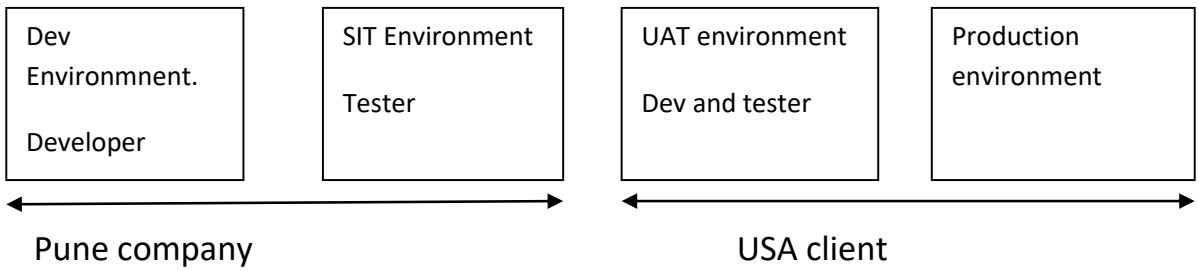
Velocity – Progress against the estimated work.



Zero Sprint - It generally refers to the first step or pre-prepartion step that comes just before the sprint 1 .It includes all activities such as Setting a development environment, preparing backlog, Project skeleton, etc.

Environments in Project.

How many environments are there in your project ideally?



Remote desktop (Access UAT)

Types of testing in different environment.

Dev Environment

1- Unit testing
2- Integration testing

Unit testing

Testing which will be performed on individual module is called as unit testing.
Developer will perform Unit testing.
Test application os checked in +ve scenario/way.

Integration testing

The testing perform on main module by combining various sub-module is called as integration testing.
Integration testing is perform by developer.

There are 3 ways of integration testing.

Top down approach

In top down approach testing is done by integrating or joining two or more modules by moving from top to button. In this first high level modules are tested , and then low level modules are tested.
In this type of testing , Stubs are used as temporary module if a module is not ready for integration testing.
In this case we are using dummy module.
Stub is dummy program created by developer.

Bottom up approach

If we have sub-module but do not have main module , then in that case we use bottom up approach.
To Check sub module, developer create dummy main module.
Developer first create program called driver.
These driver program are in xml language.

Bidirectional approach /Sandwich approach

Combination of both bottom up and top down approach.

SIT Environment (System integration testing)

Smoke testing

- It is a software testing process that determines whether the developed software build is stable or not.
- When developer sends a new build / initial build to the tester then smoke testing is performed.
- It is called as zero level testing or Build Acceptance testing.

- In smoke testing test cases are not written.
- In smoke testing we are checking whether the build is stable or not for testing.

Smoke testing checkpoints/Validation

1- Validating the core functionality
2- Validating the GUI/UI functionality
3- Validating the pages
4- Validating the links present in build
5- Validating the tabs.

- In this testing we are going to verify all the basic functionalities are working or not based on this we will accept or reject the build.
- Accordingly our test lead will send mail to development lead saying that we are accepting the build for Major testing.
- If build is not meeting the acceptance criteria at that time test lead will email to development lead saying that we are rejecting the build in that specifying that what all are the features not working.

Sanity Testing

- Sanity testing is a subset of regression testing.
- After receiving the build from developer sanity testing is performed to ensure that the code changes are working as expected.
- It is working as expected then we will be performing further functional testing.
- If the defects are found during the sanity testing, then build is rejected that is helpful in saving time for execution of regression testing.

Smoke testing	Sanity testing
It is a broad approach to testing where all parts of the application are tested.	ITS is a narrow approach to testing where specific part of the application is tested.
Smoke testing is the first testing performed on the initial build	Sanity testing is performed when build is comparatively stable. (When we are getting defect in smoke testing and after fixing it we are performing Sanity testing for that particular defect)
It is used to test end to end function of the application	It is used to test only modified or defect fixed functions.
Smoke testing is performed by developer or tester.	Sanity testing is performed by tester
Smoke testing is Build acceptance testing	Sanity is subset of regression testing
In this we are going to check whether the build is ready for testing or not	In sanity we are going to focus on particular part of our application

Retesting

When we find any defect then that defect is fixed by developer, Then tester is going to check whether the defect is fixed or not that means he will retest it. ie. it is called as retesting.

Regression testing

- When the defect is fixed we will be checking whether that change has impacted our other functionalities or not that is nothing but we are performing regression testing.
- Also we are going to perform regression testing whenever any CR comes from the customer.

- When we are moving from one environment to another we are performing regression testing.

System and Functionality testing

It is nothing but BBT

In this testing we are going to check whole functionality of the application.

System and Functional testing includes.

1- Usability testing
2- Functional testing
3- Security testing
4- Performance testing

Usability testing

Testing the user friendliness of the application is called usability testing.

Factors that decide the user friendliness of an application

1- It should be easily accessible.
2- Application should be in easy language.
3- It should be easy to understand.
4- Proper help text message should be displayed.
5- If user is doing any mistake or entering wrong credentials it should show error message.
6- Appearance of the application
7- Navigation should be very simple.

Functional testing

It depends on Validating/checking the internal functionality of the application /module/sub module as per the requirement.
It includes Six types of coverage's/Testing

1- Behavioural testing/coverage's
2- Input domain testing/Coverage
3- Error handling testing/coverage
4- Backend testing/Coverage
5- Service level testing/Coverage
6- Calculation based testing/Coverage

Behavioural testing/Coverage

In this we are validating or checking the behaviour of the application

Object	Behaviour of the object
Text box	Focus or Unfocus
Radio button	On or off
List box/dropdown	Enabled or disabled
Checkbox	Clickable or Non clickable

Input domain testing/coverage

Validating/checking the data type and size/length of data which we are passing into object.

BVA (Boundary value analysis)

We are checking size/length of data passing into the object.

In BVA We are going to check how our system is functioning or working when we insert a boundary value

Formula BVA – Min, Max, Min-1,Min+1,Max-1,Max+1

Eg. – Password textbox contain 4 to 8 char value

Object	Passing Criteria	Fail criteria
Text box 4 char no	pass	

Text box 8 char no	pass	
Text box 3 char no		fail
Text box 5 char no	pass	
Text box 7 char no	pass	
Text box 9 char no		fail

ECP (Equivalence class partition)

In ECP we are going to define the data type which we are passing into object.

Eg

	Passing criteria	Fail criteria
Mobile number	Int (0-9)	A-Z, a-z, Special char
4to8 char (that should be in alphabets)	A-Z,a-z	0-9, Special char

Error handling testing/Coverage

In this testing we will be validating/checking by passing invalid test data that system is showing warning or Error message.

Eg. Payment tab – debit card

Debit card- By passing invalid card number – error message- Please provide valid card number.

By passing blank value in debit card – These field should not be blank please provide debit card number.

Backend testing/Coverage

In this we will validating or checking that whatever operations we are performing on front end are stored in backend or database.

Eg. SIP,

Service level testing/Coverage

In this we are going to check the sequence and functions of module on the basis of functional flow diagram.

We check working of system as per functional flow diagram.

Calculation based testing / Coverage

We check Arithmetic operations , it includes addition, Subtraction, division, multiplication.

Non-Functional testing

Validating or checking the external functionality of the application or how your application is interacting with other application.

Types of non-functional testing

1- Recovery testing
2- Compatibility testing
3- Configuration testing
4- Intersystem testing
5- Installation testing
6- Parallel testing
7- Sanitation testing/Garbage testing
8- Globalisation testing

Recovery testing

It is a testing which verifies softwares ability to recover from failures like software / hardware crashes , network failures, etc
Process of checking system is able to recover from abnormal situation to normal situation.
Eg. Restart the system when browser has multiple sessions open and check whether the browser is able to recover or not. Eg download.

Compatibility testing.

It is a testing to check whether our software is capable of running on different OS, Applications, and network environments

Types of compatibility testing
Forward compatibility testing
Backward compatibility testing

Forward compatibility testing
Build is correct but OS or browser is not working properly then it is a forward compatibility.

Backward compatibility testing
I have worked in backward compatibility testing.
In backward compatibility testing i have worked in browser compatibility testing
Browser compatibility testing is of two types

1- Cross browser compatibility testing
   We will be validating our build is working on various different browser or not.
   Eg – Chrome, IE, edge, Firefox

2- **Version comparison compatibility testing**
   IN this we will be validating the different version ofn same browser is supporting are build or not.
   Eg. Chrome browser with latest versions.
   98.0 , 97.04672, 96.04664

Configuration testing

It is the process of validating or checking whether our application is working on different hardware or not.
It is nothing but hardware compatibility testing.
Eg, OS, Android, Ios devices (ipad iphone), printer , scanner.

Inter-system testing

It is the process of checking or validating whether our application shares data with other application or not.
Eg – paytm --------- bookmyshow

Installation testing

It is been perform to check if the software has been correctly installed or not and product is working as per expectation.

Parallel testing

Parallel testing is testing multiple applications or components of the application simultaneously, to reduce the time.

Sanitation testing/ Garbage testing

Validating or checking the extra features added by developer but not present in SRS document.
Eg - +91 [redacted]

Globalisation testing

It is a process of checking whether our application supports different languages or not.
It includes
Localisation testing – In this we are going to check whether our application supports local language or not
Eg, Marathi, Hindi

Internationalisation testing - In this we are going to check whether our application supports international languages or not.
Eg. Japanese, French, Spanish etc

Globalization

In this we are going to check whether our application supports English languages or not.

Security testing

Security testing is a process of checking privacy related to user application.

Types

Authorisation

Access control

Encryption and decryption.

Authorisation

Process of checking whether person is authorised or not .

Authorised person is registered person.

Access control

Process of checking whether authorised person has permission to access specific operations.

Encryption and decryption

Encryption is the process of converting normal message into meaningless message.(Unreadable format)

Whereas decryption is the process of converting meaningless message into its original format.

Performance testing

Performance testing is termed as a type of software testing to ensure that software application will perform under their expected workload.

Attributes of performance testing

1- Speed

It shows whether the software responses quickly in all conditions.

2- **Scalability**

How much load the software can handle when multiple user performing operation with the software for a particular time period.

3- **Stability**

It helps to know that the software is stable under the fluctuating loads.

4- **Reliability (consistent)**

It checks whether the software can perform a fault free task for a given period of time in a specified environment.

UAT (User acceptance testing)

1- UAT is also called as customer acceptance testing
2- In UAT we check the real time scenarios of the application.
3- UAT is a type of testing which is performed by client or end user to verify/accept the software system before moving the application to the production environment.
4- After completion of all the testing in SIT we are going ton perform UAT testing.
5- In UAT both Dev and testers are involved.
6- UAT is a type of testing which is done by the customers before accepting the final product.

Who perform UAT?

1- Client (Alpha testing)
2- End users (Beta testing)

Alpha testing	Beta testing
In alpha testing client is involved	In beta testing end users are involved
In alpha testing both dev and testers are present	In beta testing dev and testers are not available immediately
Alpha testing ensures the quality of the product before forwarding it further.	Beta testing also concentrates on the quality of the product but collects users input on the product and ensures that the product is ready for real time users .
Alpha testing requires a testing environment.	Beta testing doesn't requires a testing environment
Alpha testing may require long execution cycle	Beta testing requires only few weeks of execution.
In alpha testing critical issues can be immediately identified and fixed.	Most of the issues or feedback collected from beta testing will be implemented in the future version of the product.

Need of User acceptance testing

1- Developers have included features on their own understanding.
2- Requirement changes not communicated effectively to the developer.

Prerequisites of UAT

1- Business requirement must be available.
2- Application should be fully developed
3- Unit testing, integration testing and system testing should be completed.

4- Only cosmetic error is acceptable before UAT.
   (Cosmetic error – Spelling mistake, tab sequence, Background colour of specific fields, etc)
5- Regression testing should be completed with no major defects.
6- All the reported defects should be fixes and tested before UAT.
7- UAT environment must be ready.
8- Mail from testing team should be there that system is ready for UAT execution.

Testing Terminologies

Monkey testing

1- Monkey testing is a testing to test the application with random i/p to check whether application is crashing or not.
2- It is a speed testing
3- When there are multiple test cases and less time to deliver a product then we are performing monkey testing.
4- In monkey testing we are conducting testing on high priority test cases and core functionality of the application.

Eg – In my project – payment tab – UPI module added – user story – 100 test cases and to execute it in 2hrs.

PM will say execute the test cases in 2hrs – tester will be saying i will perform monkey testing.

Exploratory testing

1- When we are not having the knowledge about the application but we have the test data at that time we are going to perform exploratory testing.
2- Step by step we will be passing multiple test data on the application to get aware about the functionality.
3- In exploratory testing we write test cases when we are aware about the application.

Ad-hoc testing

1- We have knowledge about the application, we have test cases , but not the test data at that time we will be performing Ad-hoc testing.
2- As per our previous experience about the module or functionality of the application.

Production issue

If we found a issue/defect in production it is called as production issue.

If we have missed some functionality while doing testing, production issue may occur which is called as Hot-fix.

What should we do if we found a defect in production environment?

1- Reproduce the defect-
2- Try to receive as much information as possible.
3- Find the causes
4- Indicate the time when the bug should be fixed.
5- Check if the bug has been fixed.
6- We should analyze why the issue has happened.

Priority and severity

Severity

The impact of the bug on the application is known as severity or the bug which breaks our applications functionality is called as severity.

It can be Critical, Major or Minor for the bug.

1- Critical – It is critical , that means the main functionality is not working, and the test engineer cannot continue testing.
2- Major – If it is major , that means the supporting components or modules are not working fine. But tst engineer can continue the testing.
Eg. In CC section we cannot add multiple email ids.

3- Minor – If the severity of bug is minor, which means that all the UI problems are not working fine , but testing can be done. Eg. Terms and conditions.

Priority

Priority is important for fixing the bug or which bug to be fixed first or how soon the bug should be fixed.
It can be High , medium , low
High – If a major impact on customer application, and it has to be fixed first. Eg payment tab.

Medium – In this , the problem should be fixed before the release of the current sprint or current version. Eg Search tab

Low – The flow should be fixed if there is time, but it can be deffered with next release also. Eg. Alignments of tab

Who decides the severity and priority of a defect?

Severity type of defect is decided by tester based on the product functionality.
Priority - Priority is decided by Product owner based on customer requirement.

1- **High severity and High priority**
   Eg . payment tab is not working

2- **High priority and Low severity.**
   Eg bank logo is not correctly present

3- **Low priority and high Severity**
   Eg. Print button and download button.

4- **Low priority and low severity**

Eg. Help and support function tab not working

Difference between error, bug, defect

1- **Error** – If there is something wrong in the code then it is called as error.
2- **Defect** – while doing the testing when we found the error then it is called as defect.
3- **Bug** – when developer will accept the defect it is called as bug.

The End

MANUAL TESTING

INDEX: PAGE No:

MANUAL TESTING

What is MANUAL TESTING?

MANUAL TESTING is a process, in which all the phases of STLC (SOFTWARE TESTING LIFE CYCLE) like Test planning, Test development, Test execution, Result analysis, Bug tracking and Reporting are accomplished successfully and manually with Human efforts.

Why did U choose Testing?

- Scope of getting jobs is very very high.
- No need to depend upon any Technologies.
- Testing there for ever.
- One can be consistent throughout their life.

Who can do Testing?

Any graduate who is creative can do.

What exactly we want to get a job?

Stuff+communications+confidence+dynamism.

Why the Test engineers exclusively required in the software companies?

- One cannot perform two tasks efficiently at a time.
- Sentimental attachment.

Project: Project is something that is developed based on particular customer's requirements and for their usage only.

Product: Product is something that is developed based on the company specifications and used by multiple customers.

Note: The product based company will first have general survey in the market. Gather's clear requirements from different customers, and based on common requirements of so many customer`s. They will decide the specifications (Requirements).

Quality:

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

Note: Quality is not defined in the product. It is defined in the customer`s mind.

Latest Definition of Quality:

Quality is defined as not only the justification of all the requirements but also the presence of the value (User friendliness).

Defect: Defect is defined as a deviation from the Requirements.

Testing: Testing is a process in which defects are identified, isolated, subjected for rectification and ensure that the product is defect free, in order to produce the quality product and hence the customer satisfaction. (or) Verification & Validation of software is called Testing.

Bidding: The project is defined as a request for proposal, estimation and signing off.

Kick off meeting: It is an initial meeting conducted in the software company, soon after the project is signed off, in order to discuss the overview of the project and also to select a project manager.

Usually Project managers, Technical managers, Quality managers, High level management, Test leads, Development leads and sometimes customer representatives will be involved in this meeting.

Note: Apart from this meeting any kind of startup meeting during the process can be considered as 'Kick off Meeting.

Project Initiation Note (PIN): It is a mail prepared by the project manager and sent to CEO of the software company as well as for all the core team members in order to intimate them, that they are about to start the actual project activities.

Software Quality:

Technical:

- Meeting Customer Requirements
- Meeting Customer Expectations (User friendly, Performance, Privacy)

Non-Technical:

- Cost of Product
- Time to Market

Software Quality Assurance: To monitor and measure the strength of development process, Organization follows SQA concepts.

Software Project: Software related problems solved by software engineers through a software engineering process.

Software Development Life Cycle (SDLC):

There are six phases in software development life cycle

1. **Initial (or) Requirements phase**
2. **Analysis phase**
3. **Design phase**
4. **Coding phase**
5. **Testing phase**
6. **Delivery and Maintenance phase**

I. Initial (or) Requirement phase:

Tasks: Interacting with customer and gathering the requirements.
Roles: Business analyst –BA, Engagement manager - EM
Process:

- First of all business analyst will take an appointment from the customer, collect the template from the company and meet the customer on appointed date, gather the requirements with the support of that template and comes back to the company with the requirement document.
- The engagement manager go through the requirements, if he find any extra requirements then he will deal with excess cost of the project.
- If at all he finds any confused requirements, then he will ask the concern team to build a prototype, demonstrate that prototype to the customer, gather's the clear requirements and finally hand over the required documents to the BA

Proof: The proof document of Initial phase is Requirement's Document (RD).

These documents are called different Names in different companies:

FRS: Functional Requirement Specifications
CRS: Customer (or) Client Requirement Specifications
URS: User Requirement Specifications
BRS: Business Requirement Specifications
BDD: Business Design Document
BD: Business Document

Some companies will maintain two documents. One is for overall business flow information and second is detailed functional information, but some companies will maintain both this information's in a single document.

Template: Template is pre-defined format, which is used for preparing a document very easily and perfectly.

Prototype: Prototype is a roughly and rapidly developed model which is used for demonstrating to the client in order to gather clear requirements and also to build the confidence of a customer.
Ex: Power Point slide show.

II. ANALYSIS PHASE:

Tasks:
- Feasibility study
- Tentative planning
- Technology selection and Environment confirmation
- Requirement analysis

Roles: System Analyst – SA, Project Manager – PM, Technical Manager - TM
Process:

a. **Feasibility study**:
It is detailed study conducted on the requirement documents, in order to confirm whether the given requirements are possible within the given budget, time and available resources or not.

b. **Tentative planning:**
In this section resource planning and time planning will be temporarily done.

c. **Technology selection & Environment confirmation:**
The list of all technologies required for accomplishing the project successfully will be analyzed, the environment suitable for that project will be selected and mentioned here in this section.

d. **Requirement analysis:**
The list of all the requirements that are required by the company to accomplish this project successfully will be analyzed and listed out clearly in this section.

Note: Requirements may be Human Resources, Software's, and Hardware's.
Proof: The proof document of the Analysis phase is System Requirements Specifications (SRS).

III. DESIGN PHASE:

Tasks:
- High level designing
- Low level designing

Roles:
- High level design is done by chief Architect(CA)
- Low level design is done by Technical Lead(TL)

Process:
- The chief architect will divide the whole project in to modules by drawing some diagrams using unified modeling language (UML).
- The Team lead will divide the modules into sub modules by drawing some diagrams using the same UML.
- In this phase they will also design GUI part of the application, as well as PSEUDO CODE also developed

Proof: The proof document of this phase is Technical Design Document (TDD).
LLD: Ex: DFD-Data Flow Diagram, E-R Diagram, Class Diagram, Object Diagram.

PSEUDO CODE: PSEUDO Code is a set of English instructions, which will make the developer's more comfortable, while developing the actual source code.

IV. CODING PHASE (WHITE BOX TESTING):

Tasks: Programming (or) Coding
Roles: Programmers (or) Developers
Process: The developers will develop the actual source code by following the coding standards and also with the support of Technical Design Document.

Example's for Coding standards:

- Proper Indentation (left margin)
- Color codling's
- Proper Commenting

Proof: Proof document of the Coding phase is Source Code Document (SCD).

V TESTING PHASE (BLACK BOX TESTING):

Tasks: Testing
Roles: Test Engineer's.
Process:

- The Testing department will receive the requirement document and the test engineers will start understanding the requirements.
- While understanding the requirements, if they get any doubts they will list out all the doubts in Requirement clarification Note and sent it to the author of the requirement document and wait for the clarification.
- Once the clarification is given and after understanding all the requirements clearly they will take the test case template and write the Test cases.
- Once the first build is released they will execute the test cases.
- If at all any defects are found they will list out all the defects in the defects profile and send it to the development department and will wait for the next build.
- Once the next build is released they will re execute the required test cases.
- If at all any more defects are found they will update the defect profile. Send it to development department and will wait for the next build.
- This Process continuous till the product is defect free.

Proof: The proof of Testing phase is Quality product.

BUILD: A finally intigrated all modules set .EXE form is called Build.

TEST CASES: Implementing the creative Ideas of the Test Engineer on the application for testing, with the help of requirement document is known as TEST CASES.

VI. DELIVERY AND MAINTENANCE PHASE:

Delivery:
Tasks: Hand over the Application to the client
Roles: Deployment engineers (or) Installation engineers.
Process: The Deployment engineers will go to the customers place, install the application in to the customers environment and handover the original software to the client.
Proof: The final official agreement made between the customer and company is proof document for Delivery.

Maintenance:

Once the application is delivered. The customer will start using it, while using if at all they face any problems then that particular problem will be created as tasks. Based on the tasks corresponding roles will be appointed. They will define the process and solves the problem .This process is known as Normal Maintenance. But some customers may request for continuous Maintenance, in that case a team of members will be continuously working in the client site in order to take care of their Software.

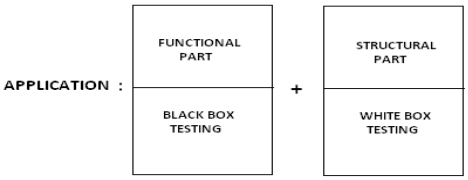
Where exactly testing comes in to the picture?
Which sort of testing we are expecting?
How many sort's of testing are there?

There are two sorts of Testing.

1. **Unconventional Testing**
2. **Conventional Testing**

Unconventional testing: It is sort of Testing in which the Quality Assurance people will check each and every out come document is according to the company standards or not right from the Initial phase to the end.

Conventional testing: It is sort of Testing in which one will check the developed applications or its related parts are working according to the exceptions or not, from the Coding phase to the end. Usually Quality Control people will do Conventional testing.



Testing methodology (Testing Techniques):

Basically there are 2 methods of Testing:

1. **Black Box Testing**
2. **White Box Testing**

Note: One more derived method is Grey Box Testing

BLACK BOX TESTING:

- It is method of testing in which one will perform testing only on the function part of the application without having the knowledge of structural part.
- Usually the Black Box Test engineers will perform.

WHITE BOX (or) GLASS BOX (or) CLEAR BOX TESTING:

- It is a method of testing in which one will perform testing on the structural part of the application.
- Usually the White Box Tester's or Developer's will perform.

GREY BOX TESTING:

It is method of testing in which one will perform testing on both the functional part as well as structural part of on application.

- Usually the Test engineer's who has the knowledge of structural part will perform.

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 acceptance level testing**

1. UNIT LEVEL TESTING:

Unit: Unit is a smallest part of an application (Program).

In this stage the white box testers will test each and every program and combinations of programs in order to confirm whether they are working according to the expectations or not. They test the structural part of a module.

2. MODULE LEVEL TESTING:

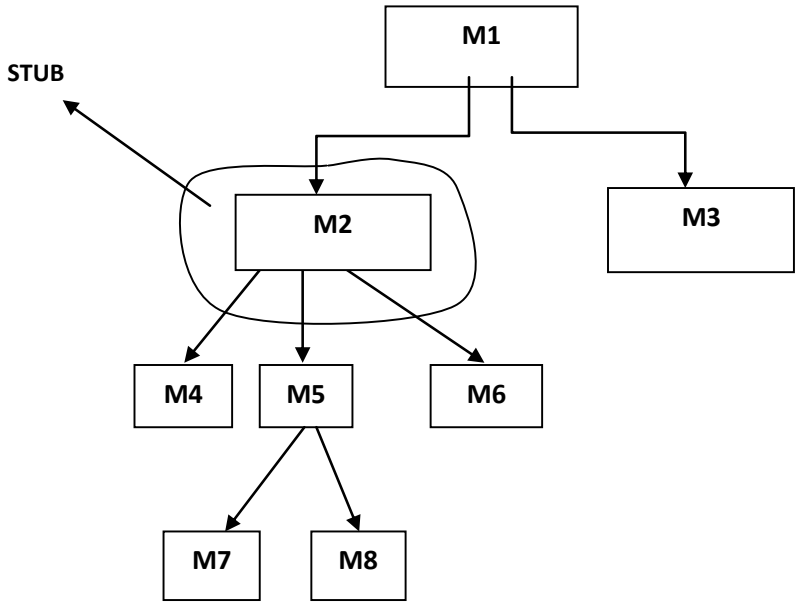
Module: Module is defined as a group of related features to perform a major task in an application.

In this stage the Black Box test engineer's will test the functional part of a module.

3. INTEGRATION LEVEL TESTING:

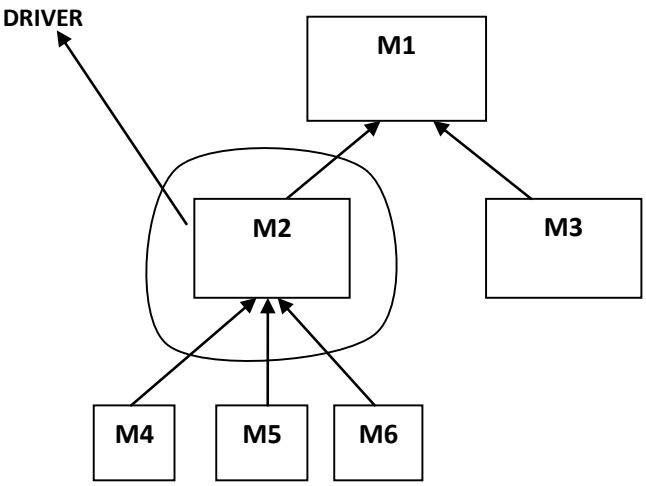
In this stage the developers will develop interfaces (Linking Prg's), in order to integrate the modules. The White Box testers will test whether the interfaces are working fine or not. Developers will integrate the modules by following any one of the following approach:

TOP-DOWN APPROACH: In this approach parent modules will be develop first and then related child modules will be integrated.



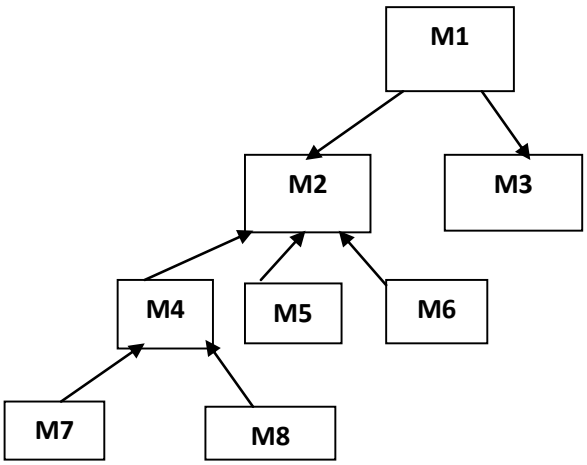
STUB: While integrating the modules in the Top-Down approach, if at all any mandatory module is missing then that module is replaced with a temporary program known as STUB.

BOTTOM-UP APPROACH: In this approach child modules will be developed first and then integrated back to the corresponding parent modules.

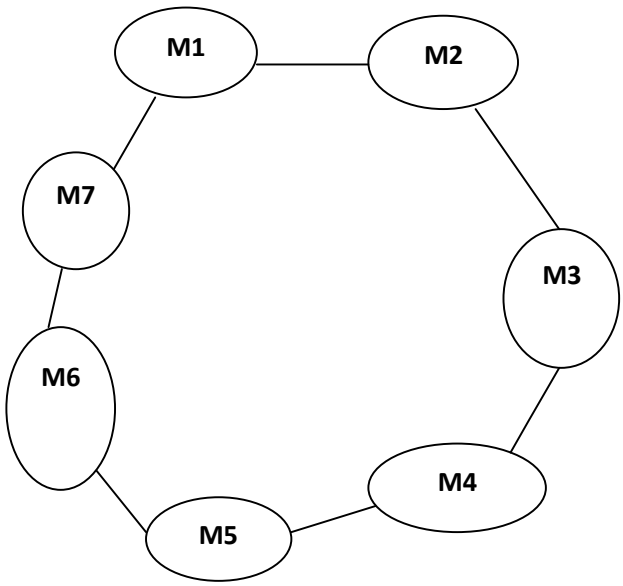


DRIVER: While integrating the modules in Bottom-Up approach, if at all any mandatory module is missing then that module is replaced with a temporary program known as DRIVER.

SANDWICH (OR) HYBRID APPROACH: This is a mixture of both Top-Down and Bottom-Up approach.



BIG BANG APPROACH: In this approach one will wait till the modules are developed and will integrate them at a time finally.



4. SYSTEM LEVEL TESTING:

In this level the Black Box test engineers will conduct so many types of testing like load testing, performance testing, stress testing, compatibility testing, system integration testing etc.

These type of Testings are also conducted:

1. Usability Testing
2. Functionality Testing

CORE LEVEL

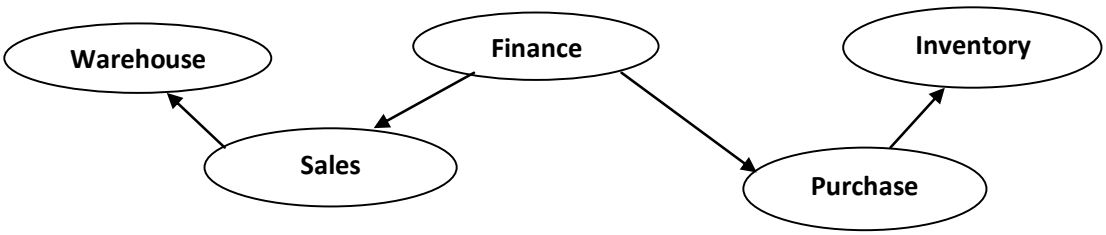
3. Performance Testing
4. Security Testing

ADVANCE LEVEL

During Usability Testing, testing team validates **User Friendliness** of screens.
During Functionality Testing, testing team validates **Correctness of Customer Requirements**.
During Performance Testing, testing team estimates **Speed of Processing**.
During Security Testing, testing team validates **Privacy to User Operations**.

SYSTEM INTEGRATION TESTING: It is a type of testing in which one will perform some actions at one module and check for the reflections in all the related areas.

Ex:



5. USER ACCEPTANCE TESTING:

In this stage the Black Box test engineers will test once again the user desired areas in the presence of the user in order to make him to accept the application.

ENVIRONMENT: Environment is defined as group of hardware components with some basic software's which can hold the business logic, presentation logic and database logic.

(Or)

Environment is a combination of Presentation layer, Business layer, and Database layer which can hold presentation logic, business logic and database logic.

TYPES OF ENVIRONMENTS:

There are 4 types of environments:

1. **STAND-ALONE ENVIRONMENT (OR) ONE-TIER ARCHITECTURE.**
2. **CLIENT-SERVER ENVIRONMENT (OR) TWO-TIER ARCHITECTURE.**
3. **WEB ENVIRONMENT (OR) THREE-TIER ARCHITECTURE.**
4. **DISTRIBUTED ENVIRONMENT (OR) N-TIER ARCHITECTURE.**

1. STAND-ALONE ARCHITECTURE:

In this environment all the three layers that is presentation layer, business layer, database layer will be available in the single tier. When the application needs to be used by a single user at a time then one can suggest this environment.

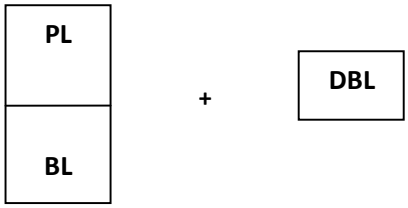
Ex: Personal Computer.

▲ **2. CLIENT-SERVER ENVIRONMENT:**

In this environment two tiers will be there. One is for clients, other is for servers. Presentation layer and business layer will be available in each and every client; database layer will be available in the server.

Whenever the application need to be used by multiple users sharing the common data in a single premises and wants to access the application very fastly and there is no problem with security. Then one can suggest client-server environment.

Ex: LAN.



3. WEB ENVIRONMENT:

This environment contains three tiers. One is for clients, middle one is for application server and the last one is for database servers.

Presentation layer will be available in client, Business layer will be available in the application server and Database layer will be available in the database servers.

Whenever the application needs to be used all over the world by limited number of people then this environment can be suggested.

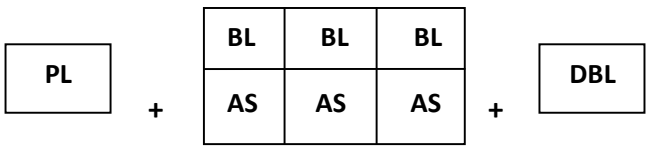
Ex: WAN.

4. DISTRIBUTED ENVIRONMENT:

This environment is similar to web environment but number of application servers are introduced in individual tiers. In order to distribute the business logic, so that load will be distributed and performance will be increased.

Whenever the application needs to be used all over the world by huge number of people then this environment can be suggested.

Ex: yahoo.co.in, yahoo.co.uk, yahoo.co.us….etc.



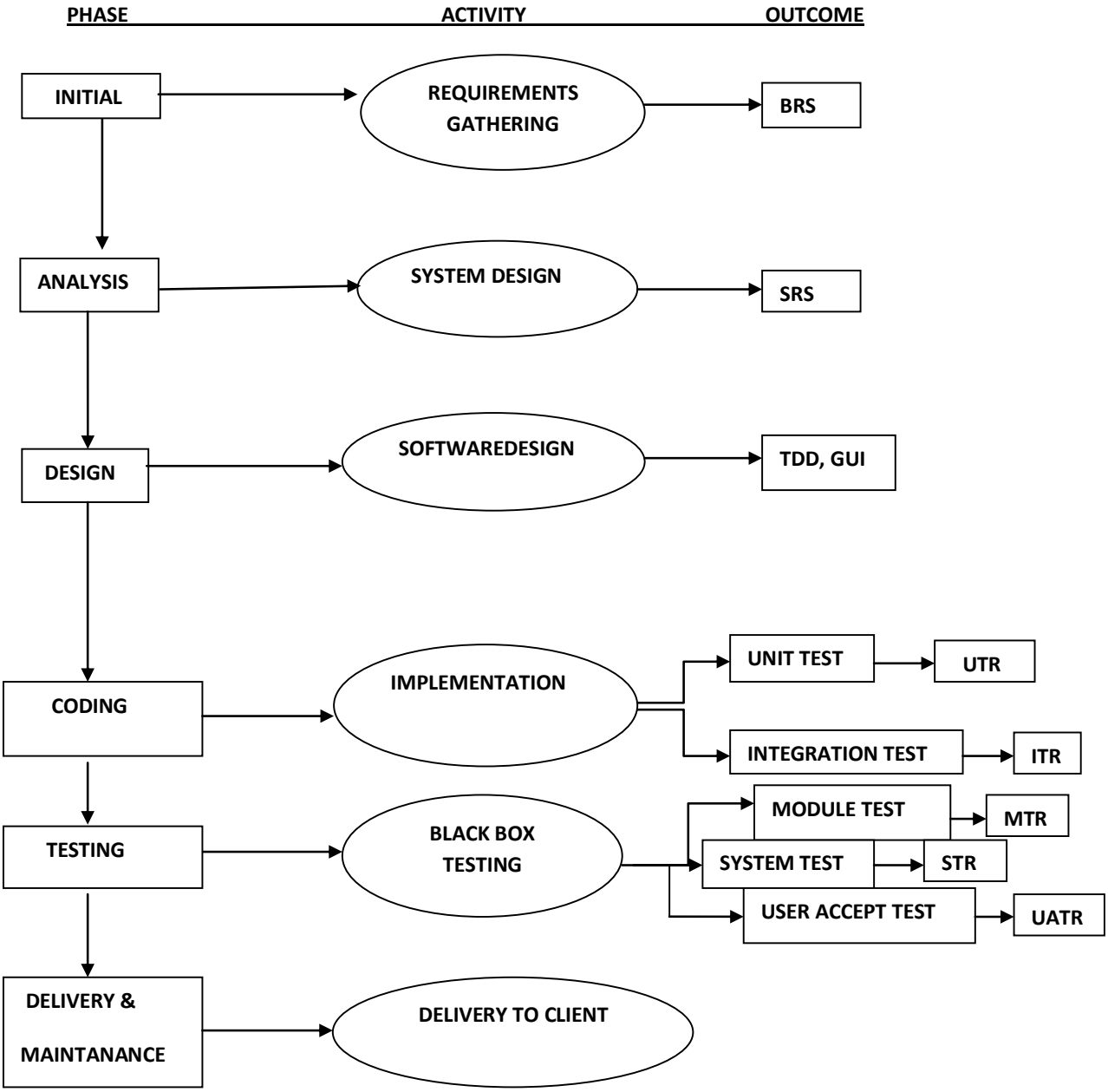
AS – APPLICATION SERVER
BL – BUSSINESS LOGIC

DATABASE: It is a base (or) a place where on can store and retrieve the data

SOFTWARE PROCESS DEVELOPMENT MODELS:

1. WATER FALL MODEL
2. PROTOTYPE MODEL
3. EVOLUTIONARY MODEL
4. SPIRAL MODEL
5. FISH MODEL
6. V-MODEL

1. WATERFALL MODEL:



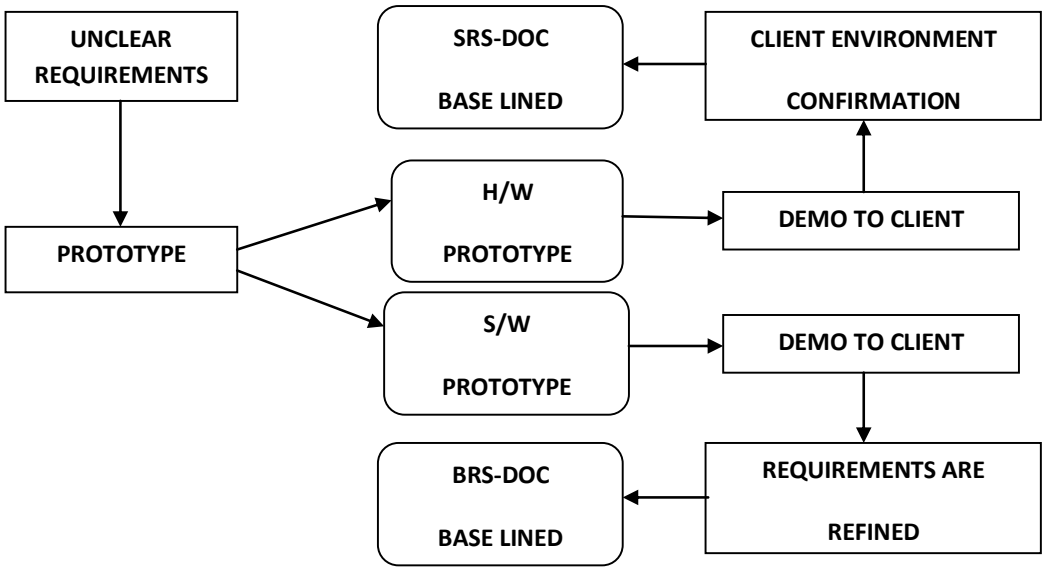
ADVANTAGES:

1. It is a simple model.
2. Project monitoring and maintenance is very easy.

DISADVANTAGES:

Can't accept the new requirements in the middle of the process.

2. PROTOTYPE MODEL:



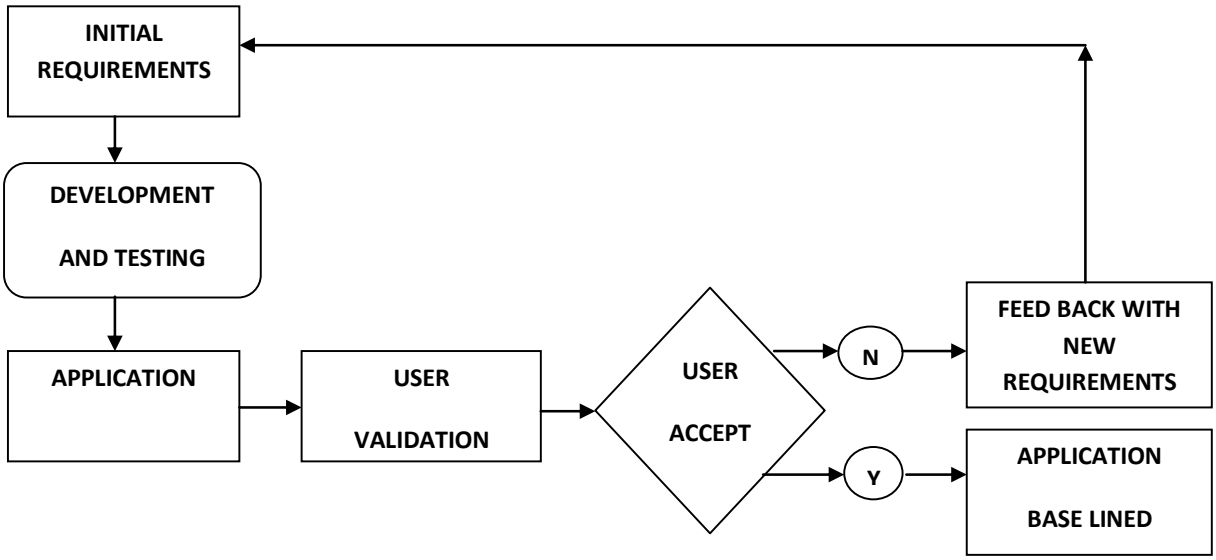
ADVANTAGES:

Whenever the customer's are not clear with their requirements then this is the best suitable model.

DISADVANTAGES:

a. It is not a full fledged process development model.
b. Prototype need to be build on companies cost.
c. Slightly time consuming model.
d. User may limit his requirements by sticking to the PROTOTYPE.

3. ENVIRONMENT MODEL:



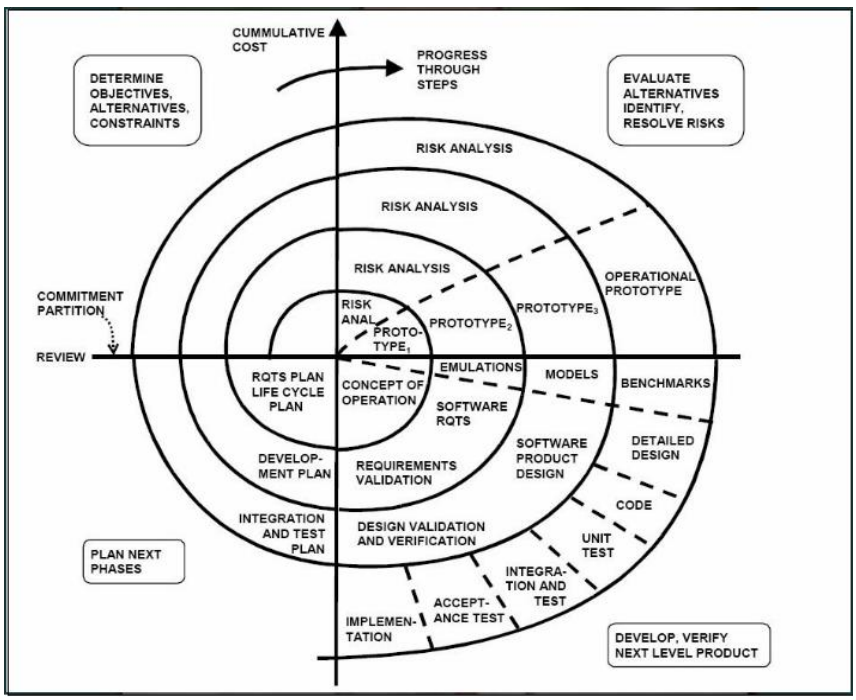
ADVANTAGES:

Whenever the customers are evolving the requirements then this is the best suitable model. Adding the new requirements after some period of time.

DISADVANTAGES:

1. Project monitoring and maintenance is difficult.
2. Can't define the deadlines properly.

4. SPIRAL MODEL:



Ex: Risk based scientific projects, Satellite projects.

ADVANTAGES:

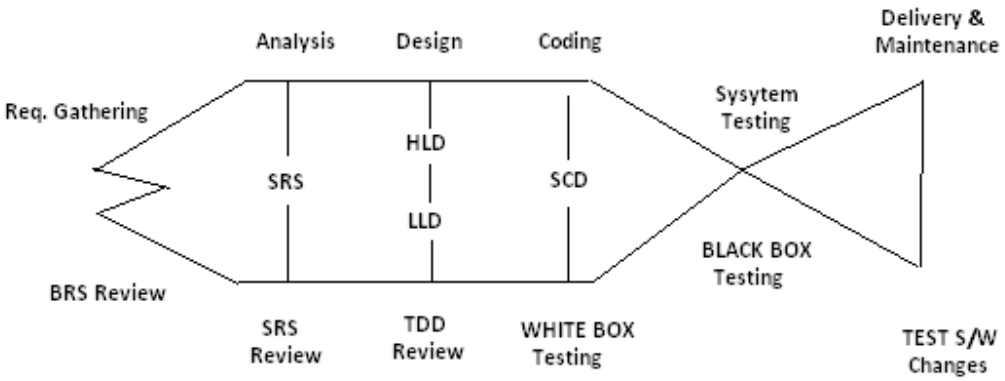
Whenever the project is highly risk based this is the best suitable model.

DISADVANTAGES:

1. Time consuming model.
2. Costly model.
3. Risk route cause analysis is not an easy task.

NOTE: Cycles depend upon Risk involved in the project and size of the project, Every cycle has 4 phases, except the last phase.

5. FISH MODEL:



ADVANTAGES:

As both verification and validation are implemented the outcome will be a quality product.

DISADVANTAGES:

1. Time consuming model.
2. Costly model.

VERIFICATION:

Verification is a process of checking each and every role in the organization in order to confirm weather they are working according to the company's process guidelines or not.

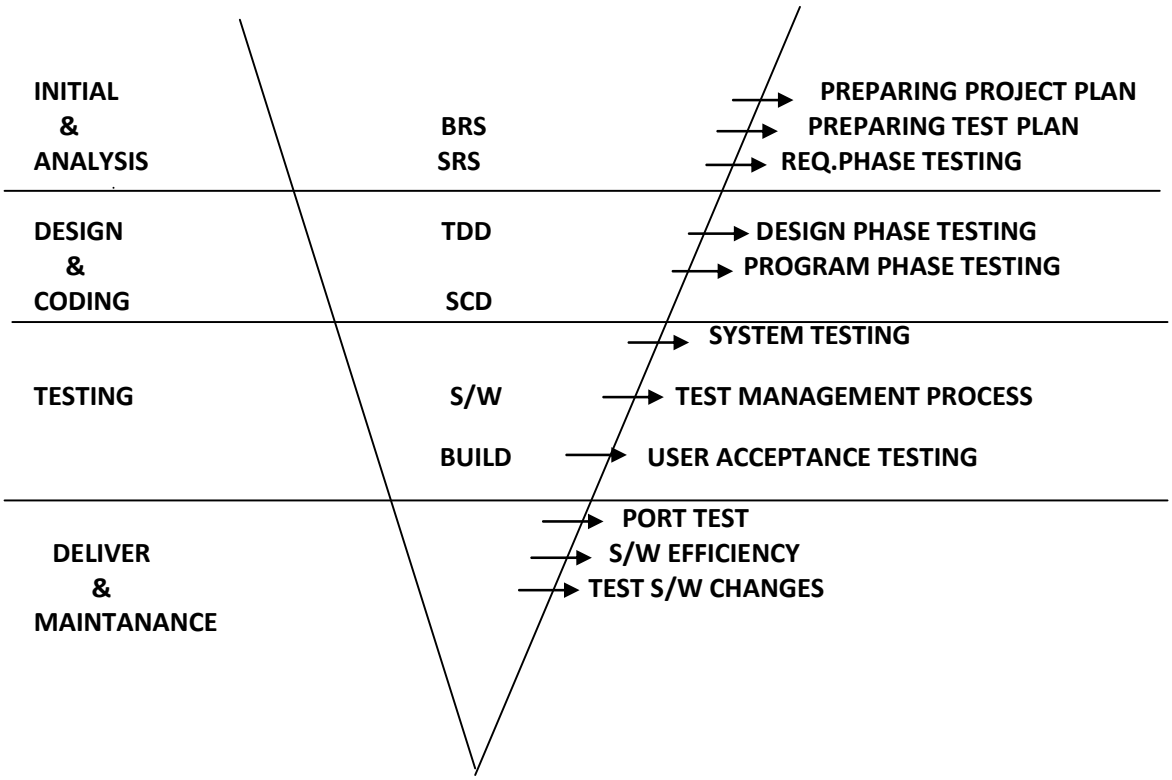
VALIDATION:

Validation is a process of checking, conducted on the developed product or its related parts in order to confirm weather they are working according to the expectations or not.

VERIFICATION: Quality Assurance people (Reviews, Inspections, Audits, Walk through).

VALIDATION: Quality Control people (Testing).

6. V-MODEL:



"V" Represents Validation and Verification.

DRE=0-1(Range).
DRE=A/A+B.
DRE = Defect Removal Efficiency.
A = Defects found by the Testing Team. .
B = Defects raised by the Customer.
DRE=80/80+20=80/100=4/5=0.8 (Good Software).

GOOD SOFTWARE :0.7-1
POOR SOFTWARE :Below 0.7

ADVANTAGES:

As verification, validation, test management process is maintained. The outcome will be a quality product.

DISADVANTAGES:

1. Time consuming model.
2. Costly model.

AGILE MODEL: Before development of the application, where testers write the test cases and gives to the development team, so that it can be easy for developers to develop defect free Programs.

TERMINOLOGY:

IF A DEVELOPER FINDS A MISTAKE IN CODE, WHILE DEVELOPING OF AN APPLICATION IT IS CALLED AN **ERROR**.
IF A TESTER FINDS A MISTAKE IN A BUILD, WHILE TESTING IT IS CALLED A **DEFECT** (or) **ISSUE.**
IF A DEFECT IS ACCEPTED BY DEVELOPER TO SOLVE IT IS CALLED A **BUG**.
IF A CUSTOMER FINDS ANY MISTAKES, WHILE USING THE APPLICATION IT IS CALLED A **MISTAKE** (or) **FAULT** (or) **FAILURE**.

A mistake in code is called **ERROR**. Due to errors in coding, test engineers are getting mismatches in application called **DEFECTS**. If defect accepted by development to solve called **BUG.**

TYPES OF TESTINGS

1. BUILD ACCEPTANCE TEST/BUILD VERIFICATION TEST/SANITY TESTING:

It is type of testing In which one will perform overall testing on the released build, in order to confirm whether it is proper for conducting detailed testing or not.

Usually during this type of testing they check the following:

- Whether the build is properly installed or not
- Whether one can navigate to all the pages of application or not
- Whether all the important functionality are available or not
- Whether all the required connections are properly established or not

Some companies even called this as SMOKE TESTING, but some companies will say that before releasing the build to the testing department, the developers will check whether the build is proper or not that is known as SMOKE TESTING, and once the build is released what ever the test engineers is checking is known as BAT or BVT or SAINITY TESTING (BAT: Build Acceptance Test, BVT: Build Verification Test).

2. REGRESSION TESTING:

It is type of testing in which one will perform testing on the already tested functionality again and again. Usually we do this in 2 scenarios:

- When ever the tester's identify the defects raise to the developers, next build is released then the test engineers will check defect functionality as well as the related functionality once again.
- When ever some new features are added, the next build is released to testing department team. Then the test engineers will check all the related features of those new features once again this is known as Regression Testing

Note: Testing new features for the first time is known as New testing it not the Regression testing.
Note: Regression testing starts from 2nd build and continuous up to last build.

3. RETESTING:

It is type of testing in which one will perform testing on the same funcatnality again and again with deferent sets of values, in order to confirm whether it is working fine or not.
Note: Retesting starts from 1st build and continuous up to last build.
Note: During Regression testing also Retesting will be conducted.

4. ALPHA TESTING:

It is type of user acceptance testing conducted in the software company by the test engineers just before delivering the application to the client.

5. BETA TESTING:

It is also a type of user acceptance testing conducted in the client's place either by the end users or third party experts, just before actual implementation of the application.

6. STATIC TESTING (Look and Feel Testing):

It is a type of testing in which one will perform testing on the application or its related factors without doing any actions.

EX: GUI Testing, Document Testing, Code Reviews etc…,

7. DYNAMIC TESTING:

It is a type of testing in which one will perform testing on the application or its related factors by doing some actions.

Ex: Functional Testing.

8. INSTALLATION TESTING:

It is a type of testing in which one will install the application in to the environment, following the guide lines provided in the deployment document(Installation Document), in order to confirm whether these guide lines are really suitable for installing the application into the environment or not.

9. PORT TESTING:

It is a type of testing in which one will install the application in to the original client's environment and check weather it is compatible with that environment or not.

10. USABILITY TESTING:

It is a type of testing in which one will test the user friendliness of the application.

11. COMPATABILITY TESTING:

It is a type of testing in which one will install the application into multiple environments, prepared with different configurations, in order to check whether the application is suitable with those environments or not.

Usually these types of testing will focused in product based companies.

12. MONKEY TESTING:

It is a type of testing in which one will perform abnormal actions on the application. Intentionally, in order to check the stability of the application.

13. EXPLORATORY TESTING:

EXPLORING: Having basic knowledge of about some concept, doing some thing and knowing more about the same concept is known as Exploring.

It is a type of testing in which the domain experts will perform testing on the application with out having the knowledge of requirements, just by parallel exploring the functionality.

14. END TO END TESTING:

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

EX: Login---> Balance Enquiry ---> Withdraw ----> Balance Enquiry ---> Logout.

15. SECURITY TESTING:

It is a type of testing in which one will check whether the application is properly protected or not.

To do the same the BLACK BOX TEST Engineers will perform the following types of Testing:

1. **AUTHENTICATION TESTING:** In this type of testing one will enter different combination of user names and passwords and check whether only the authorized people are able to access application or not.
2. **DIRECT URL TESTING:** In this type of testing one will directly enter the URL's of secured pages, in order to check whether the secured pages are directly access or not with out login to the application.
3. **FIRE WALL LEAKEGE TESTING(or) USER PRIVILLAGES TESTING:** It is a type of testing in which one will enter in to the application as one level of user and will try to access beyond the user limits, in order to check whether the fire walls are working properly or not.

16. MUTATION TESTING:

It is a type of testing in which one will perform testing on the application are its related factors by doing some changes to them.

17. SOAK TESTING/REALIABILITY TESTING:

It is a type of testing in which one will use the application continuously for a long period of time, in order to check the stability of the application.

18. ADHOC TESTING:

It is a type of testing in which one will perform testing in their own style after understanding the requirements clearly.

Note: Usually in the final stages of the project, This type of Testing can be encouraged.

19. INPUT DOMAIN TESTING:

It is a part of Functionality Testing. Test engineers are maintaining special structures to define size and type of every input object.

20. INTER SYSTEM TESTING:

It is also known as end to end testing. During this test, testing team validates whether our application build co-existence with other existing software's are not?

21. PARALLEL TESTING:

It is also known as comparative testing and applicable to software products only. During this test, testing team compare our application build with competitors products in the market.

22. PERFORMANCE TESTING:

It is an advanced testing technique and expensive to apply because testing team have to create huge environment to conduct this testing. During this test, testing team validates Speed of Processing. During this performance testing, testing team conduct load testing and stress testing.

23. LOAD TESTING:

The execution of our application under customer expected configuration and customer expected load to estimate performance is called Load Testing.

24. STRESS TESTING:

The execution of our application under customer expected configuration and un interval load's to estimate performance is called stress testing.

25. STORAGE TESTING:

The execution of application under huge amounts of resources to estimate storage limitations is called storage Testing.

26. DATA VOLUME TESTING:

The execution of our application under customer expected configuration to estimate peak limits of data is called data volume testing.

27. BIG BANG TESTING/INFORMAL TESTING/SINGLE STAGE TESTING:

A testing team conducts single stage testing, after completion of entire system development instead of multiple stages.

28. INCREMENTAL TESTING/FORMAL TESTING:

A multiple stages of testing process from unit level to system level is called incremental testing. It is also known as formal testing.

SOFTWARE TESTING LIFE CYCLE (STLC):

STLC contains 6 phases:

1. **Test Planning.**
2. **Test Development.**
3. **Test Execution.**
4. **Result Analysis.**
5. **Bug Tracking.**
6. **Report.**

1. Test planning:

Plan: Plan is strategic document which describes how to perform a task in an effective, efficient and optimized way.

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

Optimization: It is process of utilizing the available resources to their level best and getting maximum possible out put.

Note: Test plan is prepared by the Test Lead.

CONTENTS OF TEST PLAN (or) INDEX OF TEST PLAN:

1.0 Introduction
- **1.1 Objective.**
- **1.2 Reference documents.**

2.0 Test coverage
- **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 technology**
- **3.4 Configuration management**
- **3.5 Test matrices**
- **3.6 Terminology**
- **3.7 Automation plan**
- **3.8 List of automated tools**

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 and Training
10.0 Risk and Contingences.
11.0 Assumptions.
12.0 Approval information.

1.0 INTRODUCTION:

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

1.2 Reference documents: The list of all the documents that are referred while preparing the test plan will be listed out here in this section.

2.0 TEST COVERAGE:

2.1 Features to be tested: The list of all the features with in the scope are to be tested will be listed out here in this section.

2.2 Features not to be tested: The list of all the features that are not for planned for the testing will be listed out here in this section.

EX:

a. Out of scope features.
b. Low risk features.
c. Features that are planed to in corporate in future.
d. Features that are skipped based on time constrains.

3.0 TEST STRATEGY:

It is an organizational level term which describes how to perform testing on all the projects in that organization.

Test plan: It is project level term which describes how to perform testing in a particular project in a detailed manner.

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 conducted in that company will be listed out here in this section.

Ex: Build Acceptance Testing, Retesting etc.

3.3 Test design technique: The list of all the techniques that are used in that company, while developing the test cases will be listed out in this section

EX: BVA- Boundary Value Analysis, ECP- Equalance Class Partition.

3.4 Configuration management: To be discussed

Note: Metrics: Measurements.

3.5 Test metrics: The list of metrics needs to be maintained in that company during the testing process will be maintained in this section.

Ex: Test case metrics, Defect metrics etc.

3.6 Terminology: The list of all the terms along with their meaning's that are used in that company will be maintained here in this section.

Ex: GUI/UI/COSMETIC etc.

3.7 Automation plan: The list of all the areas that are planed for automation will be listed out here in this section.

3.8 Automated tools: The list of automated tools that are used in that company will be listed out in this section.

Ex: LOAD RUNNER, WIN RUNNER, QTP etc.

4.0 BASE CRIATERIA:

4.1 Acceptance criteria: When to stop testing will be clearly specified in this section.

4.2 Suspension criteria: When to suspend the build will be clearly maintain in this section.

5.0 TEST DELEVERABELS:

The lists of all the documents that are to be prepared during testing process will be maintained here in this section.

Ex: Test case document, Defect profile document etc.

6.0 TEST ENVIRONMENT:

The clear details of the environment that is about to be used for testing the application will be clearly maintained in this section.

Ex: Depending upon the Project the Environment will be selected.

1. STAND-ALONE ENVIRONMENT (OR) ONE-TIER ARCHITECTURE.
2. CLIENT-SERVER ENVIRONMENT (OR) TWO-TIER ARCHITECTURE.
3. WEB ENVIRONMENT (OR) THREE-TIER ARCHITECTURE.
4. DISTRIBUTED ENVIRONMENT (OR) N-TIER ARCHITECTURE.

7.0 RESOURCE PLANING:

Who has to do what will be clearly planned and maintained in this section.

8.0 SCHEDULING:

The starting dates and ending dates of each and every list will be clearly planed and maintained here in this section.

9.0 STAFFING AND TRAINING (KTS-KNOWLEDGE TRANSFER SESSION'S):

To accomplish this project successfully any staff or training is required then that information will be clearly maintained in this section.

10.0 RISK & CONTINGENCIES:

The list of all the potential risks and corresponding plans will be maintained in this section.

RISKS:

a. Employees may leave the organization in the middle of the project.
b. Unable to deliver the project with in the dead lines.
c. Customers imposed dead lines.
d. Unable to test all the features with in the time.

CONTINGENCIES (SOLUTION'S):

a. Employees need to maintain on the bench.
b. Proper plan ensurance.
c. What to be skipped should be planed in case of customers imposed dead lines.
d. Priority based execution.

11.0 ASSUMPTION:

The list of all the assumption's need to be made by the testing people will be maintained here in this section.

Ex: Test Data.

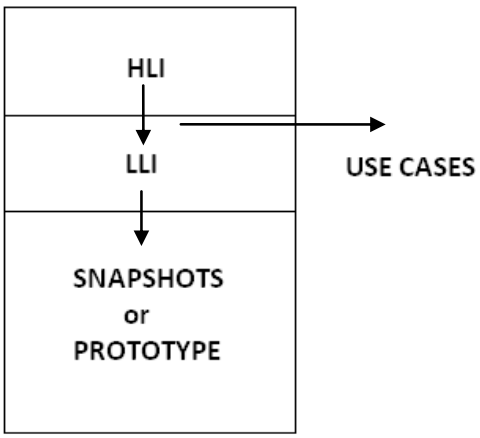
12.0 APPROVAL INFORMATION:

Who has approved this document, when it is approved will be maintained in this section.

Ex: Test Manager.

2. Test Development:

Requirement Document:

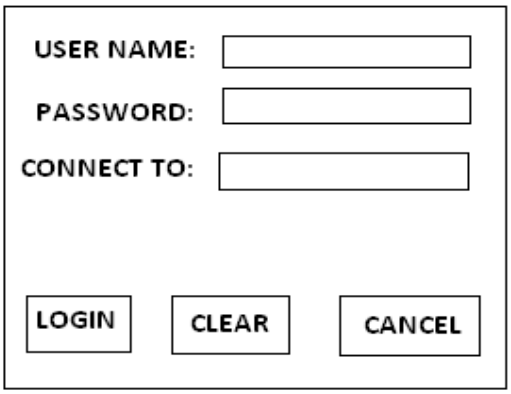


HLI-HIGH LEVEL INFORMATION, LLI-LOW LEVEL INFORMATION.

USE CASE: It describes the functionality of certain feature of an application in terms of actors, actions and responses.

Note: USECASE was prepared by Business Analyst (BA).

SNAPSHOT:



Functional Requirements:

1. Login screen should contain USER NAME, PASSWORD, CONNECT TO Fields, LOGIN, CLEAR, CANCEL Buttons.
2. Connect to field is not a mandatory field, but it should allow the user to select a database option, if required.
3. Upon entering the user name, password and clicking on login button, the corresponding page must be displayed.
4. Upon entering the information into any of the fields and clicking on clear button, all the fields must be cleared and the cursor must be available in the user name field.
5. Upon clicking on cancel button, login screen must be closed.

Special Requirements:

1. Upon invoking the application, login and clear buttons must be disable.
2. Cancel button must be always enabled.
3. Upon entering some information into any of the fields, the clear button must be enabled.
4. Upon entering some information into username and password login button must be enabled.
5. Tabbing order must be username, password, connect to, login, clear, and cancel.

USE CASE TEMPLATE:

NAME OF THE USE CASE :
DESCRIPTION OF THE USE CASE :
ACTORS INVOLVED :
SPECIAL REQUIREMENTS :
PRE CONDITIONS :
POST CONDITIONS :
FLOW OF EVENTS :

USE CASE DOCUMENT:

NAME OF THE USE CASE : LOGIN USE CASE
DSCRIPTION 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:

1. **Implicit Requirements**
2. **Explicit Requirements**

1. **Implicit Requirements:** The requirements that are analyzed by the Bossiness Analyst and his team, which will add some value to the application, and will not affect any of the customer's requirement.
2. **Explicit Requirements:** The requirements that are explicitly given by the customer are known as Explicit Requirements.

Explicit Requirements:

1. Upon invoking the application, login and clear buttons must be disable.
2. Cancel button must be always enabled.
3. Upon entering some information into any of the fields, the clear button must be enabled.
4. Upon entering some information into both the username and password fields' login button must be enabled.
5. Tabbing order must be username, password, connect to, login, clear, and cancel.

Implicit Requirements:

1. Upon invoking the application the cursor must be available in the user name field.
2. Upon entering invalid user name, valid password, and clicking on login button the following error msg must be displayed "INVALID USER NAME Please try again".
3. Upon entering valid user name, invalid password and clicking on login button the following error msg must be displayed "INVALID PASSWORD Please try again".
4. Upon entering invalid user name, invalid password and clicking on login button the following error msg must be displayed "INVALID USER NAME & INVALID PASSWORD Please try again".

GENERIC REQUIREMENTS: Universal Requirements.
SPECIFIC REQUIREMENTS: Customer Requirements.

Pre Conditions : Before Starting any Task.
Post Conditions : After Completion of any Task.

Pre Conditions : Login screen must be available.
Post Conditions : Either Home page or Admin page for valid users and error msg's for invalid users.

Flow of Events:

Main Flow:

ACTION	REPONSE
*Actor invoke the application	* Application displays the login screen with the following fields: User name, password, connect to, login, clear, and cancel buttons.
*Actor enters valid user name, valid password and clicks on login button.	*Authentication's, application displays either home page or admin page depending upon the actor enter.
*Actor enters valid user name, valid password, selects a database option and clicks on login button.	*Authenticates, application displays either home page or admin page depending upon the actor enter with the mentioned database connection.
*Actor enters invalid user name, valid password and clicks on login button.	*Go to alternative flow table 1.
*Actor enters valid user name, invalid password and clicks on login button.	*Go to alternative flow table 2.
*Actor enters invalid user name, invalid password and click on login button.	*Go to alternative flow table 3.
*Actor enters some information into any of the fields and click 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
*Actor enters invalid user name, valid password and clicks on login button.	*Authenticates, application displays the following error msg: "INVALID USER NAME Please try again".

Alternative flow table 2 (INVALID PASSWORD):

ACTION	RESPONSE
*Actor enters valid user name, invalid password and clicks on login button.	*Authenticates, application displays the following error msg: "INVALID PASSWORD Please try again".

Alternative flow table 3 (INVALID USER NAME & PASSWORD):

ACTION	RESPONSE
*Actor enters invalid user name, invalid password and clicks on login button.	*Authenticates, application displays the following error msg: "INVALID USER NAME & PASSWORD Please try again".

Alternative flow table 4 (CLEAR CLICK):

ACTION	RESPONSE
*Actor enters some information in any of the fields and clicks on clear button.	*Application clears the fields and makes the cursor available in the user name.

Alternative flow table 5 (CANCEL CLICK):

ACTION	RESPONSE
*Actor clicks on cancel button.	*Login screen is closed".

THE GUIDE LINES TO BE FOLLOWED BY A TEST ENGINEER, SOON AFTER THE USE CASE DOCUMENT IS RECEIVED:

1. Identify the module to which the use case belongs to.
A: Security module.

2. Identify the functionality of the use case with the request of total functionality.
A: Authentication.

3. Identify the actors involved in the use case.
A: Normal user/Admin user.

4. Identify the inputs required for testing.
A: Valid and invalid user names and passwords.

5. Identify whether the use case is linked with other use case or not.
A: It is linked with Home page and Admin page use cases.

6. Identify the pre conditions.
A: LOGIN Screen must be available.

7. Identify the post conditions.
A: Either Home page/Admin page for valid users, and error msgs for invalid users.

8. Identify the functional points and prepare the functional point document.

UNDERSTAND:

9. Understand the main flow of the application.
10. Understand the alternative flow of the application.
11. Understand the special requirements.

DOCUMENT:

12. Document the test cases for main flow.
13. Document the test cases for alternative flow.
14. Document the test cases for the special requirements.
15. Prepare the cross reference metrics or traceability metrics.

Functional Point:

The point at which the user can perform some actions in the application can be considered as Functional Point.

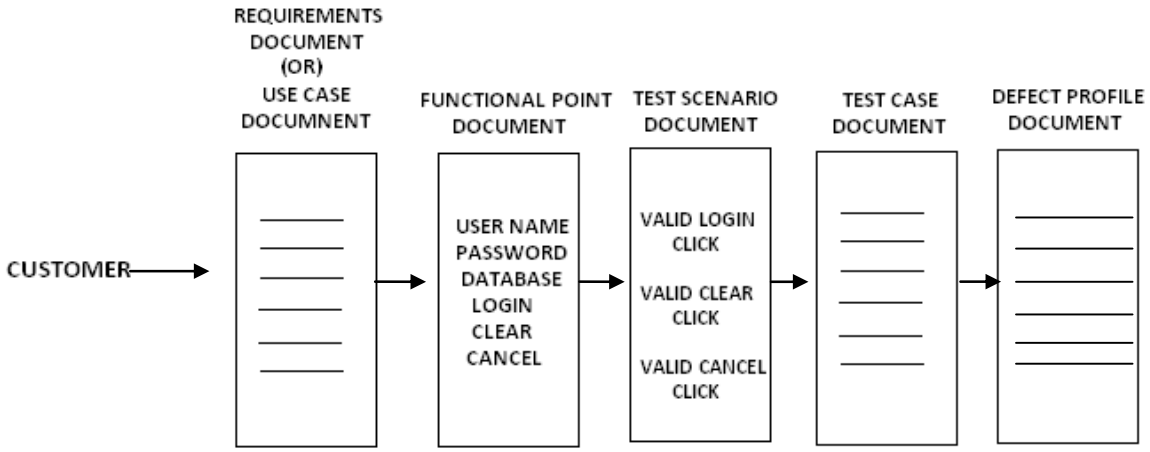
Test Scenario:

The situation where we can do testing.

There are 3 types of flow:

1. Main flow : Main page/Home Page.
2. Alternative flow : Error msgs page.
3. Exceptional flow : Server problems/Network problems.

Testing process related Documents:



TRACEABILITY MATRIX:

It is a document which contains a table of linking information used for tracing back for the reference. In any kind of confusion or questionable situation.

Note: Matrix: Combining different points in a document.

TM:

UCD ID	FPD ID	TSD ID	TCD ID	DPD ID
8.1	3	4	26	1
23.2	21	8	86	2
5.4	34	6	44	3

REQUIREMENT TRACEABILITY MATRIX:

RTM:

TEST CASE ID	REQUIREMENT ID
1	1.0
2	1.0
3	1.1
4	1.2
5	1.2
6	2.0

Note: The Test cases which are prepared related to which requirement is mentioned here in this table. If we mention Req. Id in the test case document, it same as the RTM, so we need not to maintain RTM separately for the application.

DEFECT TRACEABILITY MATRIX:

DTM:

DEFECT ID	TEST CASE ID
1	23
2	34
3	56
4	44

Note: The defects which are related to which test case is mentioned here in this table.

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

NON FUNCTIONALITY TEST CASES:

Ex:
a. Compatibility Testing.
b. Performance Testing.
c. Usability Testing.
d. Installation Testing.

FUNCTIONAL TEST CASES are further divided into 2 ways:

1. POSITIVE TEST CASES
2. NEGATIVE TEST CASES

FUNCTIONALITY TESTING: It is a type of testing in which one will perform testing the functionality of an application, functionality means behavior of the application.

Guidelines for writing the GUI Test Cases:

1. Check for the availability of all the objects.
2. Check for the consistency of the objects.
3. Check for the alignment of the objects, in case of customer requirement only.
4. Check for the spellings and grammar.

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

Guidelines for writing the Positive Test Cases:

1. A Test engineer should have positive mind set.
2. He should consider the positive flow of the application.
3. He should use only valid inputs from the point of the functionality.

Guidelines for writing the Negative Test Cases:

1. A Test engineer should have negative mind set.
2. He should consider the negative flow of the application.
3. He should use at least one invalid input for each set of data.

Guidelines for writing Test Cases:

1. Feel like Boss.
2. Action should be possible and expected value should be related to the actions based on the requirements.
3. Test Case should be clear and understandable.
4. It should be Simple, Easy and Powerful.

TEST CASE TEMPLATE:

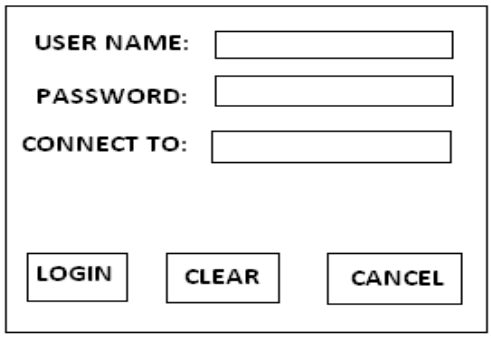
PROJECT NAME: MODULE: AUTHOR:										
REQ. ID	**TEST CASE ID**	**CATEGORY**	**PREREQUISITE**	**DESCRIPTION/ TEST STEPS**	**EXPECTED VALUE**	**ACTUAL VALUE**	**TEST DATA**	**RESULT (PASS/FAIL) BLOCKED**	**BUILD No.**	**PRIORITY**

TEST CASE DOCUMENT:

PROJECT NAME: MODULE: AUTHOR:										
REQ. ID	**TEST CASE ID**	**CATEGORY**	**PREREQUISITE**	**DESCRIPTION/ TEST STEPS**	**EXPECTED VALUE**	**ACTUAL VALUE**	**TEST DATA**	**RESULT (PASS/FAIL) BLOCKED**	**BUILD No.**	**PRIORITY**
	1	POSITIVE	NA	INVOKE THE APPLICATION	LOGIN SCREEN MUST BE DISPLAYED	LOGIN SCREEN DISPLAYED		PASS	1	
	2	GUI	NA	CHECK FOR THE AVAILABILITY OF ALL THE OBJECTS IN THE LOGIN SCREEN AS PER THE LOT	ALL THE OBJECTS MUST BE DISPLAYED AS PER THE **LOT**	ALL THE OBJECTS ARE AVAILABLE AS PER THE LOT	**LOT**	PASS	1	
	3	GUI	NA	CHECK FOR THE CONSISTANCY OF ALL THE OBJECTS IN THE LOGIN SCREEN	ALL THE OBJECTS MUST BE CONSISTANT WITH EACH OTHER	ALL THE OBJECTS ARE CONSISTANT WITH EACH OTHER		PASS	1	
	4	GUI	NA	CHECK FOR THE SPELLINGS IN THE LOGIN SCREEN	ALL THE SPELLINGS MUST BE CORRECT	ALL THE SPELLINGS ARE CORRECT		PASS	1	
	5	GUI	NA	CHECK FOR THE INTIAL POSITION OF THE CURSOR	THE CURSOR MUST BE AVAILABLE AT THE USERNAME FIELD	THE CURSOR IS NOT POSITIONED IN THE USERNAME FIELD		FAIL	1	
	6	GUI	NA	CHECK FOR THE ENABLED PROPERTY OF THE LOGIN, CLEAR & CANCEL BUTTONS	INITIALLY LOGIN & CLEAR BUTTONS MUST BE DISABLED AND CANCEL MUST BE ALWAYS ENABLED	LOGIN BUTTON IS DISABLED AND CLEAR & CANCEL BUTTONS ARE ENABLED		FAIL	1	
	7	POSITIVE	NA	ENTER SOME INFORMATION INTO ANY OF THE FIELDS & CHECK FOR THE ENABLED PROPERTY OF CLEAR BUTTON	CLEAR BUTTON MUST BE ENABLED	CLEAR BUTTON IS ENABLED		PASS	1	
	8	POSITIVE	NA	ENTER SOME INFORMATION INTO USERNAME & PASSWORD AND CHECK FOR THE ENABLED PROPERTY OF LOGIN BUTTON	LOGIN BUTTON MUST BE ENABLED	LOGIN BUTTON IS ENABLED		PASS	1	
	9	POSITIVE	NA	ENTER SOME INFORMATION INTO ANY OF THE FIELDS AND CLICK ON CLEAR BUTTON	ALL THE FIELDS MUST BE CLEARED AND THE CURSOR SHOULD BE DISPLAYED IN THE USERNAME FIELD	ALL THE FIELDS ARE CLEARED BUT CURSOR IS NOT PLACED IN THE USERNAME FIELD		FAIL	1	
	10	POSITIVE	NA	ENTER THE USERNAME, PASSWORD AS PER THE VIT AND CLICK ON LOGIN BUTTON	CORRESPONDING PAGE MUST BE DISPLAYED AS PER THE **VIT**	CORRESPONDING PAGES ARE NOT DISPLAYED AS PER THE VIT	**VIT**	FAIL	1	
	11	POSITIVE	NA	ENTER USERNAME, PASSWORD AS PER THE VIT & SELECT A DATABASE OPTION	CORRESPONDING PAGE MUST BE DISPLAYED AS PER THE **VIT** WITH THE MENTIONED DATABASE CONNECTION	CORRESPONDING PAGES ARE NOT DISPLAYED AS PER THE VIT, BUT THE MENTIONED DATABASE CONNECTION IS PROPERLY ESTABLISHED	**VIT**	FAIL	1	
	12	POSITIVE	NA	CLICK ON CANCEL BUTTON	LOGIN SCREEN MUST BE CLOSED	LOGIN SCREEN IS CLOSED		PASS	1	
	13	POSITIVE	LOGIN SCREEN MUST BE INVOKED	CHECCK FOR THE TABBING ORDER	TABBING ORDER MUST BE AS FOLLOWS: USERNAME, PASSWORD, CONNECT TO, LOGIN, CLEAR & CANCEL	TABBING ORDER IS AS FOLLOWS: USERNAME, PASSWORD, CONNECT TO, LOGIN, CLEAR & CANCEL		PASS	1	
	14	NEGATIVE	NA	ENTER THE USERNAME & PASSWORD AS PER IVIT AND CLICK ON LOGIN BUTTON	CORRESPONDING ERROR MSG SHOULD DISPLAYED AS PER **IVIT**	CORRESPONDING MSG`S ARE NOT DISPLAYED AS PER IVIT	**IVIT**	FAIL	1	
	15	NEGATIVE	NA	ENTER SOME INFORMATION ONLY INTO THE USERNAME FIELD AND CHECK FOR THE ENABLED PROPERTY OF LOGIN BUTTON	LOGIN BUTTON MUST BE DISABLED	LOGIN BUTTON IS ENABLED		FAIL	1	
	16	NEGATIVE	NA	ENTER SOME INFORMATION ONLY INTO THE PASSWORD FIELD AND CHECK FOR THE ENABLED PROPETY OF LOGIN BUTTON	LOGIN BUTTON MUST BE DISABLED	LOGIN BUTTON IS DISABLED		PASS	1	

Note: The underlined words in the Test Data column are the Hyperlinks to go for the Respective Data Table.

LOGIN SCREEN:



LOGIN OBJECTS TABLE (LOT):

S.No.	OBJECT NAME	OBJECT TYPE
1	USERNAME	TEXT BOX
2	PASSWORD	TEXT BOX
3	CONNECT TO	COMBO BOX
4	LOGIN	BUTTON
5	CLEAR	BUTTON
6	CANCEL	BUTTON

VALID INPUTS TABLE (VIT):

S.No.	USERNAME	PASSWORD	EXPECTED PAGE	ACTUAL PAGE	RESULT
1	SURESH	QTP	ADMIN	HOME	FAIL
2	ADMIN	ADMIN	ADMIN	ADMIN	PASS
3	CHIRU	SRIDEVI	HOME	CHIRU HOME PAGE	PASS
4	NAG	AMALA	HOME	NAG HOME PAGE	PASS
5	NTR	BALAKRISHNA	HOME	NTR HOME PAGE	PASS
6	VENKY	ILLU	HOME	VENKY HOME PAGE	PASS

INVALID INPUTS TABLE (IVIT):

S.No.	USERNAME	PASSWORD	EXPECTED MESSAGE	ACTUAL VALUE	RESULT
1	SURI	QTP	INVALID USER NAME PLEASE TRY AGAIN	INVALID USER NAME PLEASE TRY AGAIN	PASS
2	CHIRUTHA	SRIDEVI	INVALID USER NAME PLEASE TRY AGAIN	INVALID USER NAME PLEASE TRY AGAIN	PASS
3	VENKI	SAVITHRI	INVALID PASS WORD PLEASE TRY AGAIN	INVALID PASS WORD PLEASE TRY AGAIN	PASS
4	NTR	BALU	INVALID PASS WORD PLEASE TRY AGAIN	INVALID PASS WORD PLEASE TRY AGAIN	PASS
5	SRI	JAVA	INVALID USERNAME & PASS WORD PLEASE TRY AGAIN	SRI HOME PAGE	FAIL
6	RAJA	RANI	INVALID USERNAME & PASS WORD PLEASE TRY AGAIN	INVALID USER NAME PLEASE TRY AGAIN	FAIL

TEST DESIGN TECHNIQUES (or) INPUT DESIGN TECHNIQUES :

- **Boundary Value Analysis BVA(Range / Size):**

Min	PASS
Min-1	FAIL
Min+1	PASS
Max	PASS
Max-1	PASS
Max+1	FAIL

- **Equivalence Class Partitions ECP (Type):**

VALID	INVALID
Pass	Fail

Ex:

A login process allows user ID and Password to validate users. User ID allows Alpha Numerics in lower case from 4 to 16 characters long. Password allows alphabets in lower case 4 to 8 characters long. Prepare BVA and ECP for user ID and password.

USER ID

BVA

4 --- PASS
3 --- FAIL
5 --- PASS
16 -- PASS
15 -- PASS
17 -- FAIL

ECP

VALID	INVALID
a to z 0 to 9	A to Z Special characters Blank space

PASSWORD

BVA

4 -- PASS
3 -- FAIL
5 -- PASS
8 -- PASS
7 -- PASS
9 -- FAIL

ECP

VALID	INVALID
a to z	A to Z 0 to 9 Special characters Blank space

TEST DESIGN TECHNIQUES: They are used for making the Test Engineers to write the Test Cases very easily and comfortably, even in the complex situations. Mainly 2 famous techniques used by most of the companies are:

1. BOUNDARY VALUE ANALYSIS (BVA)
2. EQUALANCE CLASS PARTITION (ECP)

1. BOUNDARY VALUE ANALYSIS (BVA): Whenever there is a range kind of input to be tested, it is suggested to test the boundaries of the range instead of whole range, usually one will concentrate on the following values:

LB-1	FAIL
LB	PASS
LB+1	PASS
MV	PASS
UB-1	PASS
UB	PASS
UB+1	FAIL

LB-LOWER BOUNDARY
MV-MEDIUM VALUE
UB-UPPER BOUNDARY

2. EQUALANCE CLASS PARTITION (ECP): Whenever there are more no of requirements for particular feature, for huge range of data need to be tested than it is suggested to, first divide the inputs into equal classes and then write the Test Cases.

Ex: Write the Test Cases for testing a Text Box, whose requirements are as follows:
1. It should accept Min 4 Characters and Max 20 Characters.
2. It should accept only @ and _ Symbols only.
3. It should accept only Small Alphabets.

BVA:

LB-1	3
LB	4
LB+1	5
MV	12
UB-1	19
UB	20
UB+1	21

ECP:

VALID	INVALID
4	3
5	21
12	A-Z
19	Except @ and _ all the remaining Special characters
20	0-9
a-z	Spaces
@,_	Decimal Points

VIT:

S.No.	INPUTS
1	abcd
2	ab@cd
3	abcdabcda__z
4	abcdabcdabcdabcd
5	abcdabcdabzdabcda_@z

IVIT:

S.No.	INPUTS
1	abc
2	ABCD
3	ABCD@__@abcd
4	<>@+-*/,.\abcdzyxw
5	12345
6	5.4
7	abcd ABCD z@/*
8	abcdabcdABCDABCD@<>+-ab
9	ABCD123, abcd123
10	abcdabcdABCD12345@<>ab @abcd

TEST CASE DOCUMENT:

TEST CASE ID	TEST CASE TYPE	DESCRIPTION	EXPECTED VALUE	TEST DATA
1	Positive	Enter the values into the Text Box as per the **VIT**	It should accept	**VIT**
2	Negative	Enter the values into the Text Box as per the **IVIT**	It should not accept	**IVIT**

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 actual value with the expected value. If both are matching, then he will decide the result as PASS otherwise FAIL.

Note: If at all the Test case is not executed in any reason, then the test engineer will specify BLOCKED in the result column.

Ex: If application has 5 pages, in each page a next button will be available, when we click on next button, we will enter into next page, if next page button of 4th page is not working, we can't enter into 5th page and all the test cases related to that 5th page can't test. The result to all that test cases will be BLOCKED.

5. Bug Tracking:

It is a process of identifying, isolating and managing the defects.

DEFECT ID: The sequence of defect no's will be mentioned here in this section.

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

ISSUE DISCRIPTION: What exactly the defect is will be clearly described here in this section.

REPRODUCABLE DEFECTS: The list of all the steps followed by the test engineer, to identify the defects will be listed out here in this section.

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

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

VERSION: On which version the build was released will be mentioned here in this section.

Note: Depending upon the size of an application the version will be changed, but the build will be the same keep on changing. VERSION will be decided by the Software Configuration Management (SCM).

Ex: If the application version is 2.3.6, if the build 1 is released, the testing team will test the build, if they identified the defects and sent back for next build, if the new requirements from the customer are added to that application, then the version of an application changes from 2.3.6 to 2.3.7. Then the build 2 is released, the build number will be the same keeps on increasing.

DEFECT SEVERITY: Severity describes the seriousness of the application.

Severity is classified into 4 types:

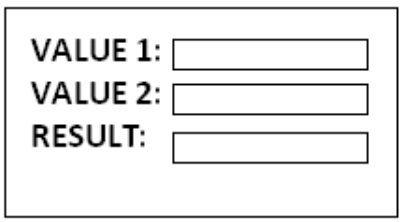
1.	**FATAL**	**SIVI 1**	**S1**	**1**
2.	**MAJOR**	**SIVI 2**	**S2**	**2**
3.	**MINOR**	**SIVI3**	**S3**	**3**
4.	**SUGGESTION**	**SIVI4**	**S4**	**4**

1. FATAL DEFECTS: If at all the problems are related to the navigational or unavailability if main functionality then such type of defects are treated as FATAL DEFECTS.

Ex:

PAGE1 → PAGE2 → PAGE3 → X.

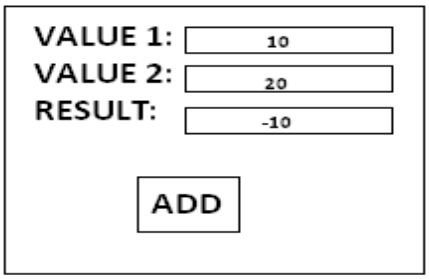
No navigation for next page.



ADD button is missing.

2. MAJOR DEFECTS: If at all the problems are related to working of the main functionality then such types of defects are treated as MAJOR DEFECTS.

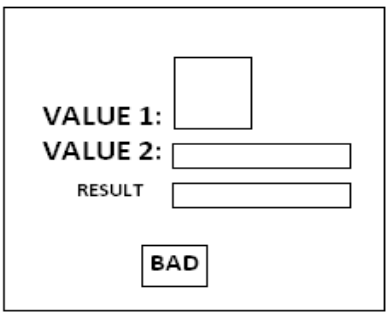
Ex:



Instead of 30 the result is -10.

3. MINOR DEFECTS: If at all the problems are related to look and feel of the application, then such type of defects are treated as MINOR DEFECTS.

Ex:



Instead of ADD button BAD button, and Text boxes size and Value names are not consistent with each other.

4. SUGGESTIONS: If at all the problems are related to value (User friendliness) of the application then such type of problems are treated as SUGGESTIONS.

Ex:

INVALID USERNAME PLEASE TRY AGAIN----POOR HELP
PLEASE ENTER ALPHANUMERIC ONLY----STRONG HELP

TOOL TIPS MUST BE HELPFUL.

DEFECT PRIORITY: It describes the sequence in which, the defects need to be rectify.

PRIORITY is classified into 4 types:

1.	**CRITICAL**	**PRI 1**	**P1**	**1**
2.	**HIGH**	**PRI 2**	**P2**	**2**
3.	**MEDIUM**	**PRI 3**	**P3**	**3**
4.	**LOW**	**PRI 4**	**P4**	**4**

Usually:

SEVERITY	PRIORITY
FATAL DEFECTS	CRITICAL
MAJOR DEFECTS	HIGH
MINOR DEFECTS	MEDIUM
SUGGESTIONS	LOW

Sometimes highest Severity defects will be given least Priority and sometimes least Severity defects will be given highest Priority.

CASE 1: LEAST SEVERITY-HIGHEST PRIORITY:

Whenever there is a customer visit, all the look and feel defects will be given highest priority.

Ex:

We are developing HDFC bank application, every page should have HDFC logo on top of it, but it shows HSBC, it is look and feel problem, even though we are not completed the main functionality of an application, customer will visit the company then we give all look and feel defects as high priority, even though the main functionality of an application is not completed.

CASE 2: HIGHEST SEVERITY-LEAST PRIORITY:

Whenever some part of the module or application is not released to the testing department as it is under construction the testers will usually raise it as a fatal defect but the development lead treats it as a least priority.

Ex:

Suppose 80% of an application developed and remaining 20% was not developed. The application is released to the testing department. The test engineer will raise this as a fatal defect in the 20% application, but developer lead will treated it as a least priority.

Note:

Sometimes highest Severity defects will be given least priority and sometimes least priority defects will be given highest priority, for this sake we use both SEVERITY and PRIORITY.

TYPES OF DEFECTS:

1. **User Interface Bugs: SUGGESTIONS**

Ex 1: Spelling Mistake → High Priority
Ex 2: Improper alignment → Low Priority

2. **Boundary Related Bugs: MINOR**

Ex 1: Does not allow valid type → High Priority
Ex 2: Allows invalid type also → Low Priority

3. **Error Handling Bugs: MINOR**

Ex 1: Does not providing error massage window → High Priority
Ex 2: Improper meaning of error massages → Low Priority

4. **Calculation Bugs: MAJOR**

Ex 1: Final output is wrong → Low Priority
Ex 2: Dependent results are wrong → High Priority

5. Race Condition Bugs: MAJOR
Ex 1: Dead Lock → High Priority
Ex 2: Improper order of services → Low Priority

6. Load Condition Bugs: MAJOR
Ex 1: Does not allow multiple users to operate → High Priority
Ex 2: Does not allow customer expected load → Low Priority

7. Hardware Bugs: MAJOR
Ex 1: Does not handle device → High Priority
Ex 2: Wrong output from device → Low Priority

8. ID Control Bugs: MINOR
Ex: Logo missing, wrong logo, version no mistake, copyright window missing, developers name missing, tester names missing.

9. Version Control Bugs: MINOR
Ex: Difference between two consecutive build versions.

10. Source Bugs: MINOR
Ex: Mistakes in help documents.

DEFECT RESOLUTION/STATUS: To set the status of the defect.
Ex: NEW, OPEN, DEFERRED…….etc.

FIX-BY/DATE/BUILD: The developer who has fixed the defect, on which date and build no will be mentioned here in this section.

DATE CLOSURE: The date on which the defect is rectified will be mentioned here in this section.

DEFECT AGE: The time gap between “Reported on” and “Resolved On”.

Note:

- Define NA in the Reproducible steps column if it is look and feel.
- Heart of the Defect profile is Issue description.
- Support column for Description is Reproducible steps.

DEFECT PROFILE TEMPLATE:

DEFECT ID	TEST CASE ID	ISSUE DISCRIPTION	REPRODUCABLE STEPS	DETECTION				DEFECT		DEFECT RESOLUTION (OR) STATUS	FIX			DATE CLOSURE
				BY	DATE	BUILD	VERSION	SEVERITY	PRIORITY		BY	DATE	BUILD	

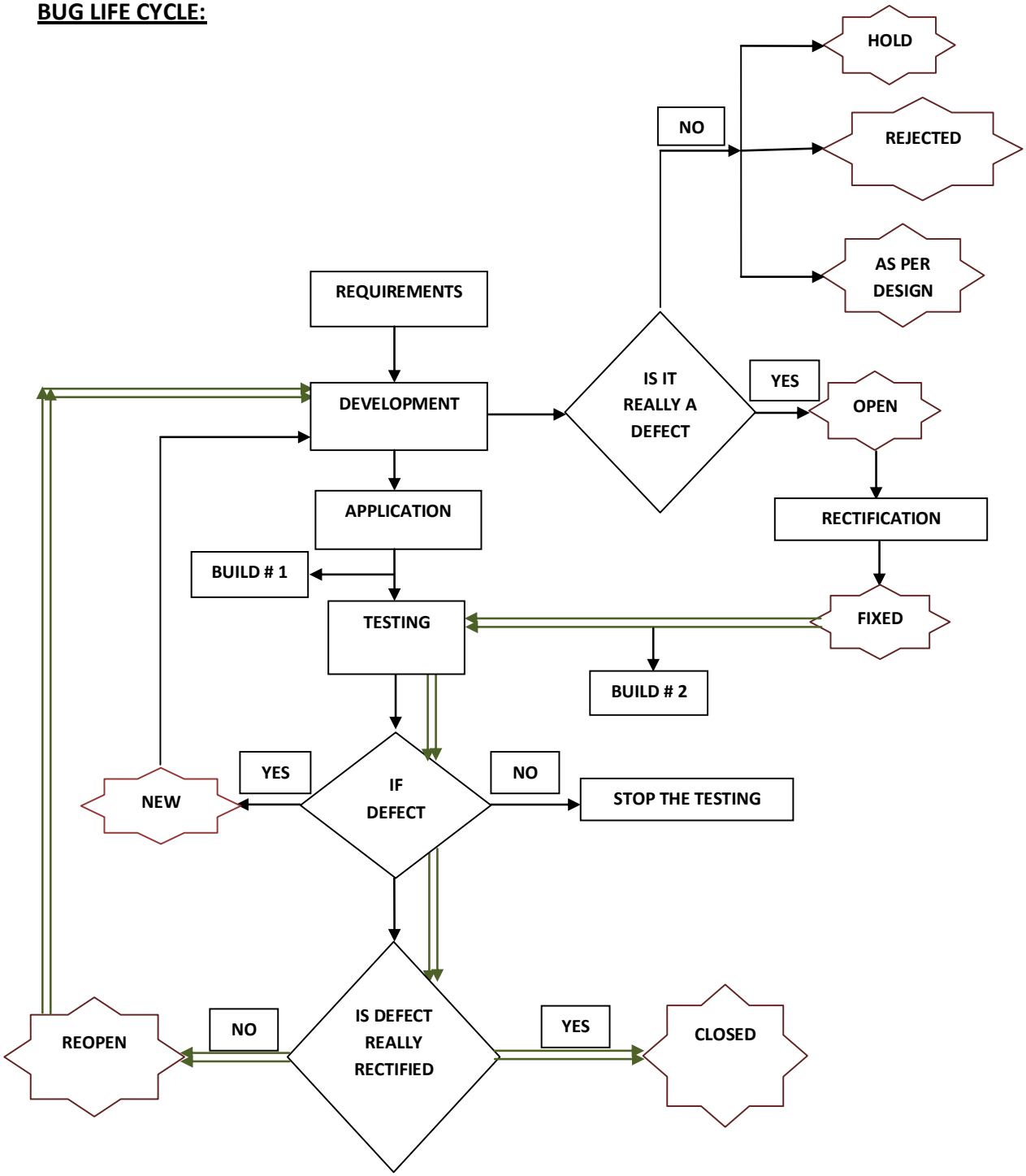
DEFECT PROFILE LOG:

DEFECT ID	TEST CASE ID	ISSUE DISCRIPTION	REPRODUCABLE STEPS	DETECTION				DEFECT		DEFECT RESOLUTION (OR) STATUS	FIX			DATE CLOSURE
				BY	DATE	BUILD	VERSION	SEVERITY	PRIORITY		BY	DATE	BUILD	
1	5	UPON INVOKING THE APPLIICATION, INITIALLY THE CURSOR IS NOT POSITIONED IN THE USER NAME FIELD	NA	SURI	16.12.09	1	2.3.6							
2	6	INITIALLY THE CLEAR BUTTON IS ENABLED, INSTEAD OF BEING DISABLED	NA	SURI	16.12.09	1	2.3.6							
3	9	UPON CLICKING ON CLEAR ALL THE FIELDS ARE CLEARED, BUT THE CURSOR IS NOT DISPLAYED IN THE USER NAME FIELD	1. ENTER SOME INFORMATION INTO ANY OF THE FIELDS. 2. CLICK ON CLEAR BUTTON. 3. OBSERVE THAT ALL THE FIELDS ARE CLEARED, BUT CURSOR IS NOT PLACED IN THE USER NAME FIELD.	SURI	16.12.09	1	2.3.6							
4	10	UPON ENTERING SURESH INTO USER NAME FIELD & QTP INTO PASSWORD FIELD THE PERSONAL HOME PAGE IS DISPLAYED, INSTEAD OF ADMIN PAGE	1. ENTER SURESH INTO USER NAME FIELD & QTP INTO PASSWORD FIELD. 2. CLICK ON LOGIN BUTTON. 3. OBSERVE THAT PERSONAL HOME PAGE IS DISPLAYED INSTEAD OF ADMIN PAGE.	SURI	16.12.09	1	2.3.6							
5	11	UPON ENTERING SURESH INTO USER NAME FIELD, QTP INTO PASSWORD FIELD. SELECT A DATABASE OPTION AND CLICK ON LOGIN BUTTON. DATABASE CONNECTION IS PROPERLY ESTABLISHED BUT PERSONAL HOME PAGE IS DISPLAYED INSTEAD OF ADMIN PAGE	1. ENTER SURESH INTO USER NAME FIELD & QTP INTO PASSWORD FIELD. 2. SELECT A DATABASE OPTION. 3. CLICK ON LOGIN BUTTON. 4. OBSERVE THAT DATABASE CONNECTION IS PROPERLY ESTABLISHED, BUT PERSONAL HOME PAGE IS DISPLAYED INSTEAD OF ADMIN PAGE.	SURI	16.12.09	1	2.3.6							
6	12	UPON ENTERING INVALID DATA INTO USER NAME & PASSWORD FIELDS. EXPECTED ERROR MESSAGES ARE NOT DISPLAYED. FOR DETAILS REFER TABLE 1.	1. ENTER USER NAME & PASSWORD AS PER THE TABLE 1. 2. CLICK ON LOGIN BUTTON. 3. OBSERVE THE ACTUAL VALUE AS PER THE TABLE 1.	SURI	16.12.09	1	2.3.6							
7	15	UPON ENTERING SOME INFORMATION ONLY INTO USER NAME FIELD, LOGIN BUTTON IS ENABLED INSTEAD OF BEING DISABLED	1. ENTER SOME INFORMATION ONLY INTO USER NAME FIELD. 2. OBSERVE THAT LOGIN BUTTON IS ENABLED INSTEAD OF BEING DISABLED.	SURI	16.12.09	1	2.3.6							

TABLE 1:

S.No.	USER NAME	PASSWORD	EXPECTED MESSAGE	ACTUAL VALUE
1	SRI	JAVA	INVALID USER NAME & PASSWORD PLEASE TRY AGAIN	PERSONAL HOME PAGE OF SRI
2	RAJA	RANI	INVALID USER NAME & PASSWORD PLEASE TRY AGAIN	INVALID USER NAME PLEASE TRY AGAIN

BUG LIFE CYCLE:



NEW: whenever the defect is newly identified by the tester, he will set the status as new.

OPEN: Whenever the developer accepts the defects then he will set the status as open.

DEFERRED: whenever the developer accept the defects and wants to rectify it in later then he will set the status as deferred.

FIXED: Once the defect is rectified then the developer will set the status as fixed, it is also called as rectified.

REOPEN & CLOSED: Once the next build is released the tester will check whether the defect is really rectified or not, if at all the defect is really rectified, then he will set the status as closed, otherwise reopen.

HOLD: Whenever the developer is confused to accept or reject, then he will set the status as hold.

Whenever the defect is in hold status there will be a meeting on that defect and if it is decided as a defect, then the developers will set the status as open otherwise the testers will close it.

REJECTED: Whenever the developers feel that, it is not at all a defect, then they will set the status as rejected.

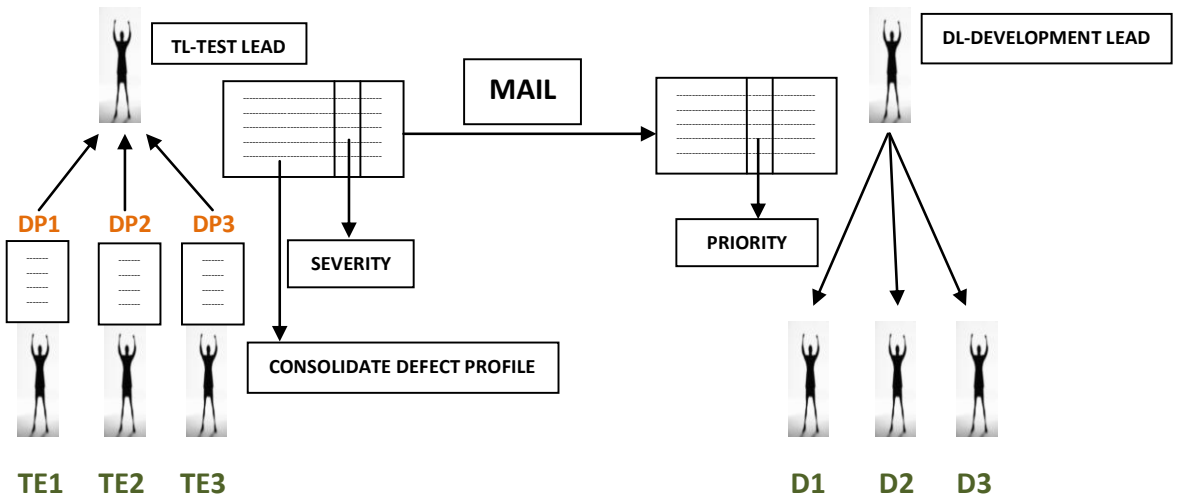
Whenever the defect is rejected, then the test engineers will once again check it, if at all they also feel it is not a defect then they will set the status as closed otherwise reopen.

AS PER DESIGN: This situation really occurs. Whenever developers feel the testers are not aware of latest requirements then they will set the status as as per design.

Whenever the defect is as per design status the testers will 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 reopen.

6. Reporting:

CLASSICAL BUG REPORTING PROCESS:

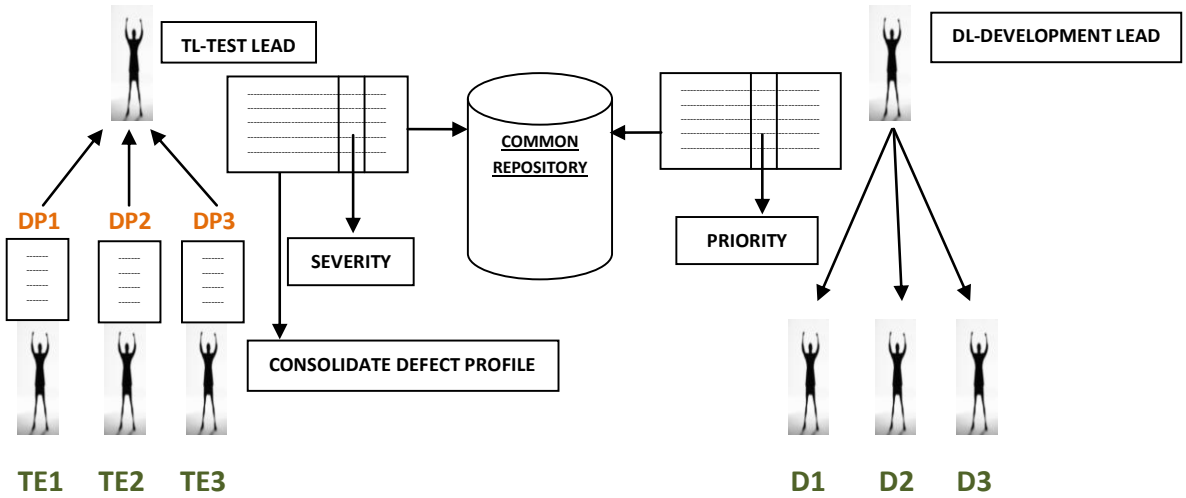


TE-TEST ENGINEER.
D-DEVELOPER.

DRAWBACKS:

1. Time consuming
2. No transparency
3. Redundancy
4. No security (Hackers may hack the Mails)

COMMON REPOSITORY ORIENTED BUG REPORTING PROCESS:



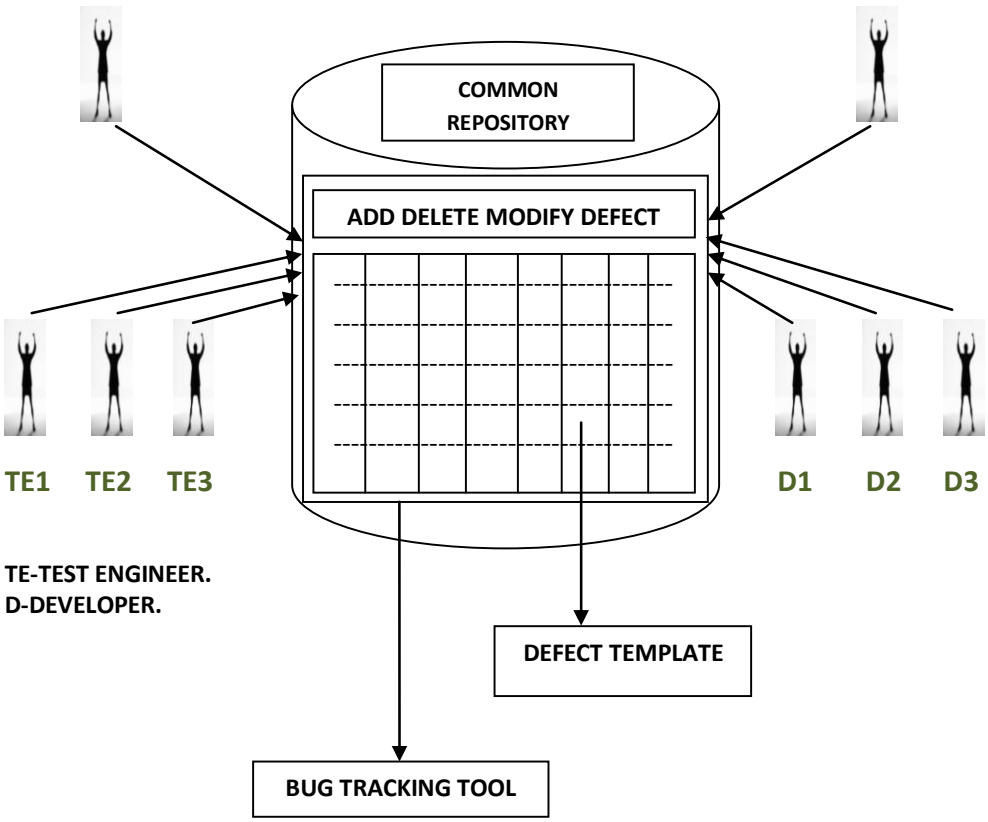
TE-TEST ENGINEER.
D-DEVELOPER.

COMMON REPOSITORY: It is a server which can allow only authorized people to upload and download.

DRAWBACKS:

1. Time consuming
2. No transparency
3. Redundancy (Repeating)

BUG TRACKING TOOL ORIENTED BUG REPORTING PROCESS:



BUG TRACKING TOOL: It is a software application which can be access only by the authorized people and provides all the facilities for bug tracking and reporting.

Ex: BUGZILLA, PR TRACKER, ISSUE TRACKER…..etc.

PR-PERFORMANCE REPORTER.

Ex:
No Transparency: Test engineer can't see what was happening in the development deportment and developer can't look what was the process is going in the testing deportment.
Redundancy: There is a chance that some defect will be found by all the test engineers.

Ex: Suppose all the test engineers found the defect of the login screen login button, and then they raise the same as a defect.

BUG TRACKING TOOL ORIENTED BUG REPORTING PROCESS:

The test engineer enter into bug tracking tool, he add defect to the template with add defect feature and writes the defect in corresponding columns, the test lead parallely observes it by bug tracking tool, and he assign Severity.

The development lead also enters into the bug tracking tool. He assigns the priority and assigns the task to the developer. The developer enters into the tool and understands the defect and rectifies it.

Tool: Something that is used to complete the work easily and perfectly.

Note: Some companies' use their own Bug Tracking Tool, this tool is developed by their own language, this tool is called 'INHOUSE TOOLS'.

TEST CLOSURE ACTIVITY: This is the final activity in the testing process done by the test lead, where he will prepare the test summary report, which contains the information like:

Number of cycles of execution,
Number of test cases executed in each cycle,
Number of defects found in each cycle,
Defect ratio and……etc.

TERMINOLOGY:

DEFECT PRODUCT: If at all the product is not satisfying some of the requirements, but still it is useful, than such type of products are known as Defect Products.

DEFECTIVE PRODUCT: If at all the product is not satisfying some of the requirements, as well as it is not usable, than such type of products are known as Defective Products.

QUALITY ASSURANCE: It is a dependent, which checks each and every role in the organization, in order to confirm whether they are working according to the company process guidelines or not.

QUALITY CONTROL: It is a department, which checks the develop products or its related parts are working according to the requirements or nt.

NCR: If the role is not following the process, the penalty given for him is known as NCR (Non Conformances Raised). Ex: IT-NCR, NON IT-MEMO.

INSPECTION: It is a process of sudden checking conducted on the roles (or) department, without any prior intimation.

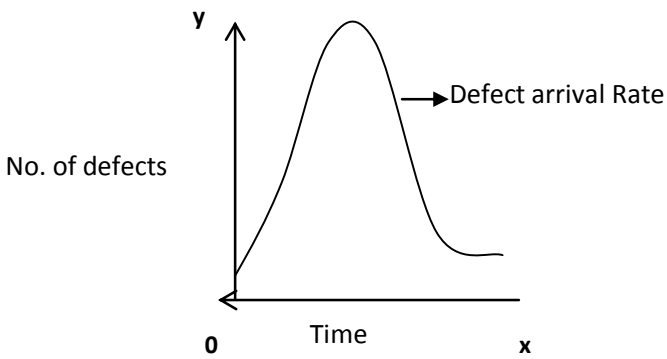
AUDIT: It is a process of checking, conducted on the roles (or) department with prior intimation well in advance.

There are 2 types of Audits:
1. INTERNAL AUDIT
2. EXTRNAL AUDIT

INTERNAL AUDIT: If at all the Audit is conducted by the internal resources of the company, than the Audit is known as Internal Audit.
INTERNAL AUDIT: If at all the Audit is conducted by the external people, than that Audit is known as External Audit.

AUDITING: To audit Testing Process, Quality people conduct three types of Measurements & Metrics.

1. QAM (Quality Assessment Measurement):
These measurements used by Quality Analysts / PM during testing process (Monthly once).



Stability:
20% testing → 80% defects
80% testing → 20% defects

Sufficiency:
→ Requirements Coverage
→Type-Trigger analysis

Defect Severity Distribution:
→ Organization- Trend limit check.

2. TMM (Test Management Measurement):
These measurements used by Test Lead during testing process (weekly twice).

Test Status:
→ Completed
→ In progress
→ Yet to execute

Delays in Delivery:
→ Defect arrival rate
→ Defect resolution rate
→ Defect age

Test Efficiency:
→ Cost to find a defect (No of defects / Person-Day)

3. PCM (Process Capability Measurement):
These measurements used by Project Management to improve capability of testing process depends on feed back of customer in existing maintenance software's.

Test Effectiveness:
→ Requirements Coverage
→Type-Trigger analysis

Defect Escapes (Missed defects):
→ Type-Phase analysis

Test Efficiency:
→ Cost to find a defect (No of defects / Person-Day)

CAPA (CORRECTIVE ACTIONS & PREVENTIVE ACTIONS):

CORRECTIVE ACTIONS: Whenever the role has committed repairable mistake, than the Corrective Actions will be taken care, in order to correct such type of mistakes.
PREVENTIVE ACTIONS: Whenever the role has committed irreparable mistake, than the Preventive Actions will be taken care, in order to correct such type of mistakes in future.
SCM (SOFTWARE CONFIGURATION MANAGEMENT): It is a process where in 2 tasks are perform:
1. CHANGE CONTROL
2. VERSION CONTROL

CHANGE CONTROL: It is a process of updating all the related documents, whenever some changes are made to the application, in order to keep the Documents and Applications in sink with each other.
VERSION CONTROL: It is a process in which one will take care of Naming conventions and Versions.

COMMON REPOSITORY: It is basically a server, which can be accessed only by the authorized people, where in they can store the information and retrieve the information.

CHECK IN: It is a process of uploading the information from the Common Repository.
CHECK OUT: It is a process of downloading the information from the Common Repository.
BASE LINE: It is a process of finalizing the documents.

PUBLISHING/PINNING: It is a process of making the finalized documents available to the relevant resources.

RELEASE: It is a process of sending the application from the development department to the testing department (or) from the company to the market.

DELIVERY: It is a process of sending the application from the company to the client (or) from the market to the client.

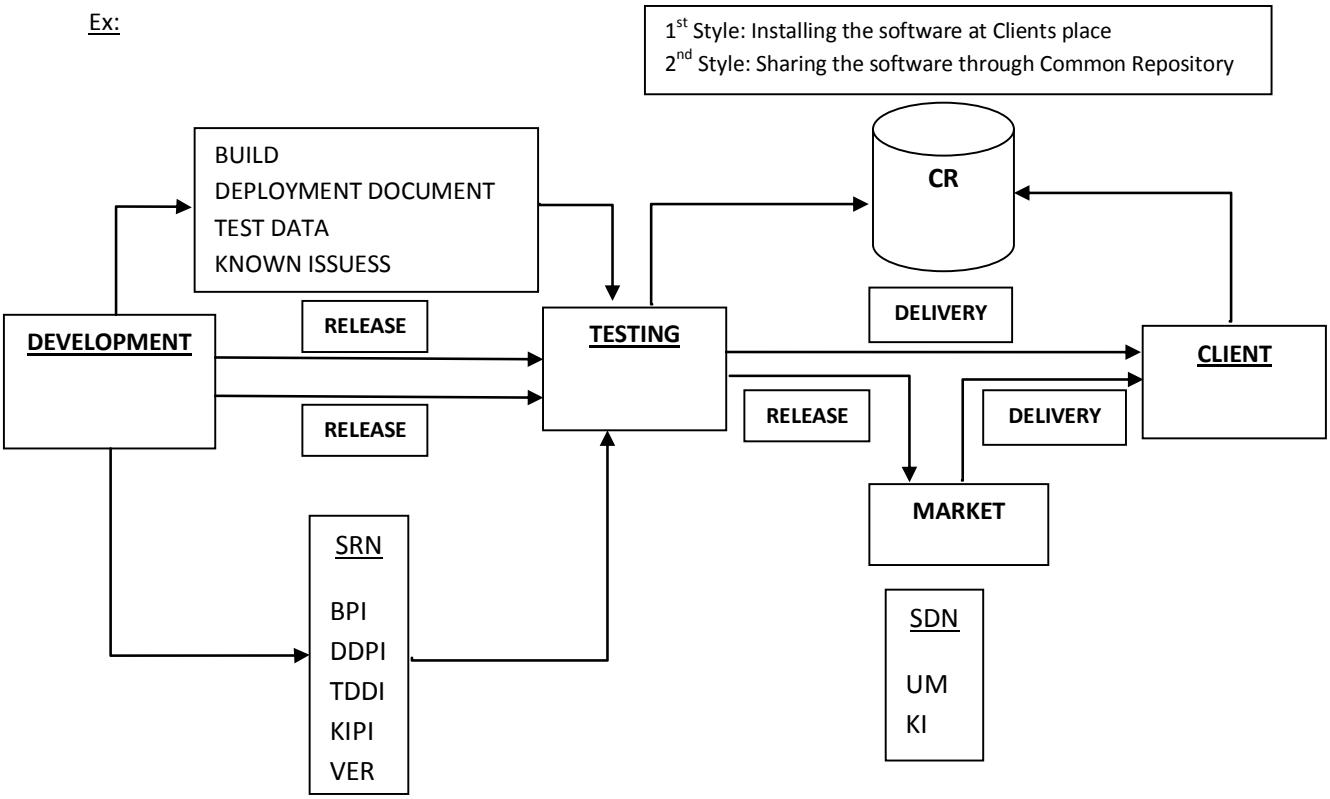
SRN (SOFTWARE RELEASE NOTE): It is a note prepared by the development department and sent to the testing department during the release, it contains the following information:

- Build path information.
- Deployment document path information.
- Test data path information.
- Known issues path information.
- Release manager name.
- Release date.
- Build number.
- Version number.
- Module name……etc

SDN (SOFTWARE DEVELOPMENT NOTE): It is a note prepared by team of members under the project managers guidance and given to the customer during the delivery, it contains the following information:

- User manual
- Known issues

Ex:



REVIEW: it is defined as either of process of studying (or) process of checking depending upon the role involved.

REVIEW REPORT: It is an outcome document of review, which may contain either list of doubts (or) list of comments.

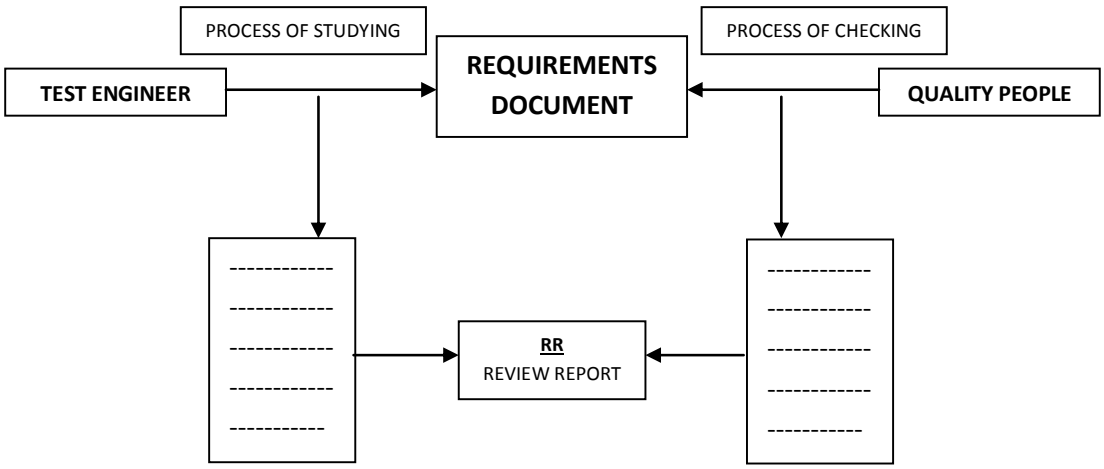
COLLEAGUES: People working in same company with different designations.

PEER: Colleagues with same designation in the company.

PEER REVIEW: It is a process of checking, conducted by the colleagues at same designation.

PEER REVIEW REPORT: It is an outcome document of peer review, which contains the list of comments.

Ex:



TEST SUIT: Combination of all the different types of test cases is known as Test Suit.

TEST BED: Combination of Test Environment and Test Suit.

HOT FIX: Fixing the defect immediately.

DEFECT AGE: The time gap between opening date and closing date of the defect is known as Defect Age.

LATENT DEFECT: The defect that is found late after some releases are known as Latent Defects.

SLIP AGE: The extra time taken to accomplish a task is known as Slip Age.

ESCALATION: It is a process of intimating the issue related information to the next level of authority.

METRICS: Clear measurement of any task is known as Metrics.

TRACEABILITY MATRIX: It is a document which contains a table of linking information used for tracing back for the reference in any kind of confusion or questionable situation.

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

TEMPLATE: It is predefined format, which is used for preparing a document easily and perfectly.

BENCH MARK: The standard with which usually we compare with is known as Bench Mark.

CHANGE REQUEST: It is a process of requesting the changes; to do the same customers will use the change request template.

REQ ID.	PURPOSE OF CHANGE	DESCRIPTION OF CHANGE	MODULE NAME

IMPACT ANALYSIS: Whenever the customer proposes some changes, the senior analyst will analyze, How much impact will fall on the already developed part? This process is known as Impact Analysis.

WALK THROUGH: It is defined as an informal meeting between two or more roles, may be for checking something (or) for transferring something.

CODE WALK THROUGH: It is a process of checking conducted on the source code document, in order to confirm whether it is developed according to the coding standards or not.

CODE OPTIMIZATION/FINE TUNING: It is a process of reducing the number of lines of code (or) complexity of code, in order to increase the performance.

PPM (PERIODIC PROJECT MEETING): It is a meeting conducted periodically, in order to discuss the status of the project. Usually they discuss the following points:

- Percentage covered in the project during the period
- Percentage not covered in the project during the period
- Tasks completed during that period.
- Defects found during that period.
- Slip ages.
- Reasons for the slip ages.
- Technical issues.
- HR related issues.

PPR (PERIODIC PROJECT REPORT): It is a report prepared by Team Lead by interacting with the team members before the PPM is conducted. This document contains the information related to all the above said points.

MRM (MANAGEMENT REPRESENTATIVE MEETING): It is a meeting conducted, in order to discuss the status of the company. Usually they discuss the following points:

- Success rate and growth rate of the company.
- Projects that are recently signed off.
- Projects that are in pipe line.
- Customers' appraisals (Good comments).
- Future plans.
- Internal audit reports.
- Individual appraisals.

PATCH: Whenever the test engineers suspend a build the developers will rectify the problems and release the same build as Patch.

WORK AROUND: A workaround is a method, sometimes used temporarily, for achieving a task or goal when the usual or planned method isn't working. In information technology, a workaround is often used to overcome hardware, programming, or communication problems. Once a problem is fixed, a workaround is usually abandoned.

WAYS OF TESTING:

There are 2 ways of Testing:

1. **MANUAL TESTING**
2. **AUTOMATION TESTING**

1. MANUAL TESTING: Manual Testing is a process, in which all the phases of STLC (Software Testing Life Cycle) like Test planning, Test development, Test execution, Result analysis, Bug tracking and Reporting are accomplished successfully and manually with Human efforts.

DRAWBACKS:

1. More no of human resources are required.
2. Time consuming.
3. Less accuracy.
4. Tiredness.
5. Simultaneous actions are almost impossible.
6. Repeating the same task again and again in same fashion is almost impossible.

2. AUTOMATION TESTING: Automation Testing is a process, in which all the drawbacks of Manual Testing are addressed properly and provides speed and accuracy to the existing testing process.

DRAWBACKS:

1. Automated tools are expensive.
2. All the areas of the application can't be tested successfully with the automated tools.
3. Lack of automation Testing experts.

Note: Automation Testing is not a replacement for Manual Testing, it is just continuation for Manual Testing.

Note: Automation Testing is recommended to be implemented only after the application has come to a stable stage.

PC SURENDRA REDDY M.C.A.

Suri.poluchalla@live.com

Q. On what basis PM select test factors/issues or Test Methodology/TRM

- Type of project – Traditional project, Off the share(testing), Maintenance
- Project requirement
- Identity scope of application
- Identify tactical risk
- The main objective of test methodology is to develop the TRM or finalized TRM.

Q. What are the different types of Levels of testing?

- In my organisation we have 4 levels of testing,
- Level 0 →Sanity/ Smoke Testing
- Level 1 →BBT/ System & function testing
- Level 2 →Regression Testing on Modified built
- Level 3 →Final Regression
- I have worked in all level of testing.

Q. What are Test Batch/ Test Suite/ Test Bed/ Test Set?

- It is a clubbing of independent test cases are called Test Batch/ Test Suite/ Test Bed/ Test Set

Q. What parameter/inputs will consider while preparing Test plane?

1. Requirement /Development document
2. TRM
3. Team formation
4. Identify tactical risk

Q. What are the risk factors / issues have faced during your 3 years careers?

- Lack of knowledge on that domain (require more training)
- Lack of resources (over time work due to resources)
- Lack of development process (Simple mistake, Label name wrong, Inform these problem to development manager)
- Lack of test data (Performing Ad-hoc testing)
- Delay & delivery (Overtime)
- Lack of communication between development team & testing team
- **Lack of budget**

Q. What are main parameters in test plane?

- Test plane is a project level document & prepared by Test Lead.
- Test lead concentrate on
    1. Job allocation-
    2. Resource allocation
    3. Estimation
    4. Main aim of test plane to decide start date & end date of testing activity.

Q. What is test plane & what it will contain?

- After completion of team formation & analyzing risk. Test lead focuses on preparation of test plane document.
- Test plane contains-
    1. Test plan ID
    2. Iteration
    3. Test items
    4. Features to be tested
    5. Features not to be tested
    6. Test pass/ Fail criteria
    7. Test environment
    8. Suspension criteria
    9. Test deliverable
    10. Testing Task
    11. Staff & training need
    12. Responsibility
    13. Scheduled
    14. Risk & mitigation
    15. Signature & approvals

Q. How you design the test cases?

- After completion of test plane by Test lead then test engineer prepares test cases for respective user story.
- There are 3 types of test cases
    1. Business logic based test cases
    2. Input domain test cases
    3. UI based test cases

Q. How to write test cases?

- While writing test cases we are considering following points
    1. Identify the test scenario of application
    2. Write positive & negative test cases of respective test scenario.
    3. Start test case with initial world as **Verify**

Q. Which test cases will be considering as good test cases?

- It should be simple
- It should be easy to understand
- It should be specific
- It should cover all the functionality
- It should be more precise

Q. What are the things basically focus during test case review?

- In review they focus on
  1. Business requirement based coverage
  2. Use case based coverage
  3. Functionality coverage
  4. Spelling mistakes
  5. Grammar
  6. Duplication /Replication
  7. Standard/Guidelines follow

Q. Who define priority & severity of Test case & defect?

- Test case – Priority – define by - BA
- Test case – Severity – define by - Tester
- Defect – Priority – define by – (Initial stage tester) PM / BA / Designer
- Defect – Severity – define by – Tester

Q. How many reports in your project or Do you involved in reporting?

- Yes, In my organisation, we have different reports we are preparing
  1. Defect report -- Tester
  2. Test report / Test proof -- Tester
  3. Weekly status report (it is depend on project) -- Test Lead
  4. Sprint report -- Test Lead
  5. Test closer report -- Test Lead

Q. What is a Re-producible defect?

- **Re-producible defect-**Those defects will be generated again and again in testing; these types defects are **Re-producible defect.**
- **Re-producible defect** will be generated due to **Deployment process/ JAR file not available in SIT environment/ Configuration file is not updated in SIT environment**.

Q. What is a duplicate defect?

- **Duplicate defect-** The defects which is related or same or similar as current defects, these type of defects called Duplicate defect
- Ex. Paytm- Recharge module – All mobile no. not accepted tester has raised the defects
- Again Airtel, BSNL, VI mobile no is not accepted- defect- Duplicate defect

Q. What are parameters or filed are present while created defects?

- Defects will created in JIRA/ **HPALM**
- Parameters or filed are present while created defects,
  1. **Summery*-** Short description
  2. **Details description***- Step to re-product a defects
  3. **Assigned to*** – To developer name
  4. **Created by*** – Tester name
  5. **Severity***-
  6. **Priority**-
  7. **Defected environment**-
  8. **Status***- New, Open, Fix, Re-open, Close, reject, Differed
  9. **Detected date***- System generated date
  10. **Screenshot/ Attachment**-

Defect report-

A	B	C	D	E
Defect ID	**Defect Description**	**Raised by**		**Root cause**
ID	**Title**	**Assigned To**	**State**	
61295	Before 8:00 off-peak and student limited contract access the access application	Akit partil	Done	configuration
61351	on public holiday access did not get for Off-peak and student limited contract	Akit partil	Done	Coding Issue
59415	when create one year contract using MembershipAPI for created contract showing wrong initial period dates and increase dates	Akit partil	Done	Conding error
61431	:when create one year and two year contract using MembershipAPI for created contract showing wrong end dates and increase dates	Akit partil	not done	Deployment Issue

Test summery report-

A	B	C	D	E	F	G
Sprint	**User ID**	**User Descrip[tion**	**Test case Desgin**	**Test case Execution**	**Test case Skipped**	**Defect**
Sprint 21	GR0023	Groww with Auto pay options	38	38	0	5
Sprint 21	GR0031	Groww with Mf	23	23	0	8

Test closer report-

Real time Interview question-

1. **What is your approaches when developer will not accepting the defects?**

- **Answer-** In my project, when we got defects then we will create / log into JIRA/HPALM
- In crate defects we will add attachment/ Screenshot but still developer is not accepting
- Then we will take call with developer & share the screen and show case to him
- But still developer is not accepting- We will go to another Tester system & check the defects
- If defects is present then we will inform to inform
- But still developer is not accepting- We will inform to Test lead & same we will mail to PM, Test lead, developer, designer & all tester
- In daily stand up we will raised these issue.

2. **What is an approach when a defect is not re-producible?**

- **Answer-** In my project, when we got defects then we will create / log into JIRA/HPALM
- In create defects we will add attachment/ Screenshot. If same defects is not re- producible in developer system
- Tester will check same defects in his system but if defects is not re-producible in your system
- We will go to another Tester system & check the same defects
- May be defects reasons is – **Deployment process**, Configuration file changes, Jar not update / environment problems. Due to cash or Cookies, etc.

3. **What are challenges you have in your whole carrier/ last Project?**

- **Answer-** Lack of knowledge about project/ domain
- Lack of test data in testing
- Lack of resources in testing-
- Lack of development process -
- Lack of time or deployment
- Lack communication in between BA, developer, tester
- Lack of budget in project

4. **When you should know testing is completed?**

- **Answer-** All TCE will be completed
- All defects will closed/ Fixed
- Traceability matrix is done

5. **How many test cases you are writing per day & how may your are executing?**

- **Answer-** It is totally **depends on US.**
- If US is complex the I will write **more test cases**
- If US is simple the I will write **less test cases**
- An average, I am writing **25 to 30** test cases **per day**
- An average, I am executing **15 to 20** test cases **per day**

6. **How many defects your are creating/lock per day/ Per US?**

- **Answer-** It is totally **depends on US**
- If US is complex & developer is not done coding properly then I **found more defects**
- If US is simple & developer is done coding properly then I **found less defects**
- An average, I am **creating/lock 3 to 4 or 4 to 5** test cases per day/ Per US

7. **What is your Roles& Responsibility?**

- **Answer- In project, my roles & Reasonability**

    1. **Requirement analysis –** When we got US/ SRS
    2. **Identify Test scenario –** For every US
    3. **Write a test cases-** Against test scenario
    4. **Test review-** Interview review
    5. **Test cases execution –** Against test cases
    6. **Defect report / Log/ Create –** Against Test execution
    7. **Test proof –** against US
    8. **Test Summery report – against every assigned US (TL/Tester)**
    9. **Client interaction/UAT–** When we give demo in **sprint review meeting**

Q. What is Critical defect or trick defect found in your carrier**? Issues facing in your previous project? Risk you have facing in your carrier?**

- Lack of knowledge of Test engineer on domain/ project
- Lack of resources – assign more task to existing resource
- Lack of test data – Ad-hoc testing
- Lack of development process Rigour –Simple mistake, Label name wrong,
- Lack of communication between development & testing team
- Lack of delay & delivery
- Lack of time & cost

Q. When would you come to know your testing is completed?

- Test case execution completed
- All defect resolved
- Traceability matrix

Q. How we are receiving the build?

-

Q. How many test cases you design and execute per day?

- **In manual -Test cases design**- exact we can't say, it totally depends on user story.
- An I average, I will write 25 - 30 test cases & review these test cases
- **Test cases execution –** Test cases execution 15 - 20 test cases
- **In automation-** We write 2 to 3 or 3 to 4 test script for automation.

Q. How many defect you are lock per day?

- Exact we can't say, it totally depends on which testing or user story you are testing.
- Yes, an average, I am locking 3 to 4 defect per day.

Q. What is Blocker defects?

- **Defect which is shows some part of** functionality is not working of application/ build
- **Ex.** Paytm → Recharge module- BSNL mobile no. not accepting

Q. What is Show stopper defect?

- Defect which is stop the functionality of application/ build
- **Ex.** Paytm → After click on Recharge Icon- Recharge module is not opening

Q. What is Bug leakage?

- Defect/ Bug which is missed from SIT and found in UAT or Product
- Occurred due to – Functionality is not understand, Deployment is not proper, Environments problem of UAT or Product

Reports-

- In my project, following

1. **Daily wise report** → Working report
2. **Test cases execution** → Tester will prepared Test proof
3. **Defect report** → Tester will prepared defect report
4. **Test summery report** → Test lead Current Sprint US (TCD, TCE, TCS, defects, etc)
5. **Test Closer report** →Test lead (Module test summery)

Tool Used in Testing–

- There are multiple tools in project as
- Project management tool – HP-ALM, **JIRA**, etc.
- Web Service / API Testing tool – SOAPUI, POSTMAN
- **Web serve API stored repository– Swagger, Tommcat server**
- Automation – Selenium tool, java langue's
- Data base testing – **Oracal 11g**, MySQL, SQL server 2014

1. **What is difference between test plan & test strategy?**

Sr.No	Test Plan	Test Strategy
1	It is prepared by team lead	It is prepared by Scrum master
2	It is project level document	It is organization level document/ company level document
3	This document is used to define scope of testing & different testing activities	This document contains planning for all the testing activities & delivering quality product
4	Component in the test plan Main Factors- 1. Job allocation 2. Resource allocation 3. Estimation Component List-	Component in the test strategy Component list-
5	Primary goal of the test plan is 1. What to test 2. How to test 3. When to test 4. Who will verify 5. Whom to test	Primary goal of the test strategy is 1. What techniques to follow 2. Which module to check
6	Test plan is a dynamic document it can be changed, when new requirements have occurred	It's a static document it can't be changed or modified

2. **What is difference between test case scenario & test case design?**

Sr.No	Test Scenario	Test Case
1	Test Scenario are derived from the user stories	derived from the Test Scenario Test Case are
2	Test Scenario focuses on "What to test"	Test Case focuses on "How to test"
3	Test Scenario high level actions	Test Case low level actions
4	Test Scenario describes test condition/requirements/functionality, which we have to validate	Test Case describes validation procedures of each functionality or requirements

Interview Question-

3. **What is SDLC & STLC?**
4. **What is STLC?**
5. **What is your organization Testing process?**
6. **What is test bed/ Test suite**
7. **What is Test plane & what is the purpose of test plane**
8. **Are you involved in Test plane?**
9. **What are different levels of Testing?**
10. **At what level you have done testing?**
11. **Do you involve in the test plan?**
12. **What is agile test plan?**
13. **What is difference between test plan & test strategy?**
14. **What is difference between test case scenario & test case design?**
15. **What is test case review?**
16. **What are the roles & responsibility of the test engineer?**
17. **What is test responsibility matrix/ test matrix?**
18. **What is traceability matrix?**
19. **What is excel sheet format for test cases?**
20. **What is defect/ bug life cycle?**
21. **What are the different defect component in the HPALM?**
22. **When developer reject your defect then what you do in that condition?**

Interview question-

1. **What is your origination testing process? OR Tell me about STLC?**
2. **What is difference between SDLC & STLC**
3. **How you're prepared test plane? What it will contains**
4. **What is difference between Test scenario & Test cases?**
5. **Consider a page which is used for admission; write Test scenario & Test cases?**
6. **What are different types of Reviews process? Which review is following in your project?**
7. **If you got comments/ suggestion in review, what you will do?**
8. **What is traceability matrix and where you are prepared these?**
9. **What is defect life cycle?**
10. **What are differed defects? Who will decide differed defects status?**
11. **What is re-producible defects & duplicate defect? What is your approach on these?**

Test scenarios for Whatsapp

- While preparing test scenarios always start with Verify word.

SR.NO.	TEST SCENARIOS
TS_1	Verify that user can able to download the what's app from play store
TS_2	Verify that user can able to install the what's app
TS_3	Verify that what's app user can able to register using new number
TS_4	Verify that whether what's app user get verification code
TS_5	Verify that how many times what's app user able to enter wrong verification code
TS_6	Verify that what's app user can able to get contacts in contact list from phone
TS_7	Verify that What's app user can back up the chat
TS_8	Verify that What's app user can able to set profile picture
TS_9	Verify that What's app user can able to edit his/her name
TS_10	Verify that what's user can we able to send messages to another user
TS_11	Verify that what's user can we able to send pictures to another user
TS_12	Verify that what's user can we able to receive pictures from another user
TS_13	Verify that what's user can we able to send videos to another user
TS_14	Verify that what's user can we able to receive videos from another user
TS_15	Verify that what's user can we able to share contacts to another user
TS_16	Verify that what's user can we able to receive contacts from another user
TS_17	Verify that what's user can we able to share current location to another user
TS_18	Verify that what's user can we able to receive current location of another user
TS_19	Verify that what's user can we able to share live location to another user
TS_20	Verify that what's user can we able to receive live location of another user

TS_21	Verify that what's user can we able to see how much time the live location of another user
TS_22	Verify that another user can we able to see how much time the live location of us
TS_23	Verify that the location shared from what's app user is disable after time limit or not.
TS_24	Verify that what's app user can able to share pdf document to another user
TS_25	Verify that what's app user can able to share word document to another user
TS_26	Verify that what's app user can able to share excel document to another user
TS_27	Verify that what's app user can able to check whether sent document delivered to another user or not
TS_28	Verify that what's app user can able to check whether sent document seen by another user or not
TS_29	Verify that what's app user send how much size of videos to another user
TS_30	Verify that what's app user send how much size of audio clip to another user
TS_31	Verify that what's app user send how much size of audio recording to another user
TS_32	Verify that what's app user can able to send voice message to another user.
TS_33	Verify that what's app user can able to send emoji's to another user.
TS_34	Verify that what's app user can we able to view particular contact by clicking on option bar
TS_35	Verify that what's app user can able to access search box
TS_36	Verify that what's app user can able to mute the notifications of messages received from another user.
TS_37	Verify that what's app user can we able to change wallpaper of chat window
TS_38	Verify that what's app user can able to block another user.
TS_39	Verify that what's app user can able to clear the chat.
TS_40	Verify that what's app user can able to export the chat.
TS_41	Verify that what's app user can able to add shortcut of contact of another user.
TS_42	Verify that what's app user can navigate to back page when user click on back arrow.
TS_43	Verify that what's app user can able to see profile picture of another user.

TS_44	Verify that what's app user can able to download profile picture of another user.(-ve)
TS_45	Verify that what's app user can able to see about and mobile number of another user.
TS_46	Verify that user can able to see common groups
TS_47	Verify that what's app user can able to edit the contact details of another user saved in our phone
TS_48	Verify that what's app user can able to see the contact details of another user in our address book
TS_49	Verify that what's app user can able to verify security code
TS_50	Verify that what's user can able to delete message sent to another user as delete for me only
TS_51	Verify that what's user can able to delete message sent to another user as delete for all
TS_52	Verify that what's app user can able to forward same message to different users
TS_53	Verify that what's app user can able to forward same message to how many different users at a time
TS_54	Verify that what's app user can able to copy the sent message
TS_55	Verify that what's app user can able to see in info delivered or seen time
TS_56	Verify that what's app user can able to see notification of message received with count
TS_57	Verify that what's app user can able to download the images received from another user.
TS_58	Verify that what's app user can able to download the videos received from another user.
TS_59	Verify that what's app user can able to make payment to another user.
TS_60	Verify that what's app user can able to make audio call to another user.
TS_61	Verify that what's app user can able to make video call to another user.
TS_62	Verify that what's app user can see history of audio & video call history.
TS_63	Verify that what's app user can able to upload status as image.
TS_64	Verify that what's app user can able to upload status as video
TS_65	Verify that what's app user can able to write status manually.
TS_66	Verify that what's app user can able to change background colors of written status manually

TS_67	Verify that status uploaded by what's app user is visible for another user
TS_68	Verify that for how much time that uploaded status by what's app user is visible for another users.
TS_69	Verify that what's app user can able to see status uploaded by other user.
TS_70	Verify that what's app user can able to create group
TS_71	Verify that what's app user can able to add how many members in a one group.
TS_72	Verify that what's app user can able to set and change group icon
TS_73	Verify that what's app user can able to set and change group name
TS_74	Verify that what's app user can able to share photos into group
TS_75	Verify that other what's app group members can able to share images in group
TS_76	Verify that what's app user can able to share videos into group
TS_77	Verify that other what's app group members can able to share videos in group
TS_78	Verify that what's app user can able to share contact into group
TS_79	Verify that other what's app group members can able to share contacts in group
TS_80	Verify that what's app user can able to share location into group
TS_81	Verify that other what's app group members can able to share location in group
TS_82	Verify that what's app user can able to change group settings
TS_83	Verify that what's app user can able to see the info of images, videos, documents, contact shared on group as deliverd and seen by how many members
TS_84	Verify that what's app user can able to create broadcast
TS_85	Verify that what's app user can able to add how many members in to broadcast
TS_86	Verify that what's app user can able to set and change broadcast icon
TS_87	Verify that what's app user can able to set and change broadcast name
TS_88	Verify that what's app user can able to share photos into broadcast
TS_89	Verify that what's app user can able to share videos into broadcast

TS_90	Verify that what’s app user can able to share contact into broadcast
TS_91	Verify that what’s app user can able to share location into broadcast
TS_92	Verify that what’s app user can able to invite a friend by sharing link
TS_93	Verify that what’s app user can able to navigate to contact us page
TS_94	Verify that what’s app user can able to navigate to terms and privacy policy
TS_95	Verify that what’s app user can able to change notifications settings
TS_96	Verify that what’s app user can able to change chat settings
TS_97	Verify that what’s app user can able to change account settings
TS_98	Verify that what’s app user can able to open what’s app web by scanning QR on web browser
TS_99	Verify that what’s app user can able to change media auto download setting
TS_100	Verify that what’s app user can able to manage storage