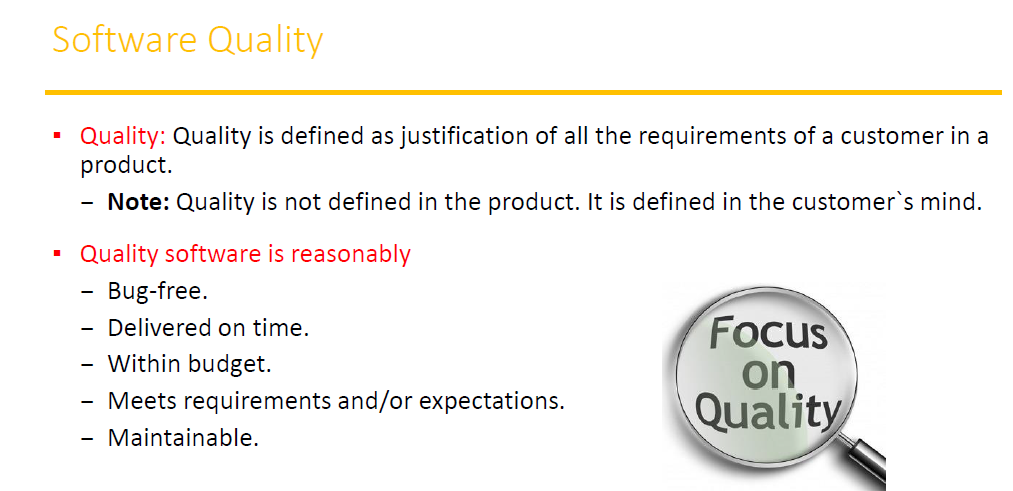
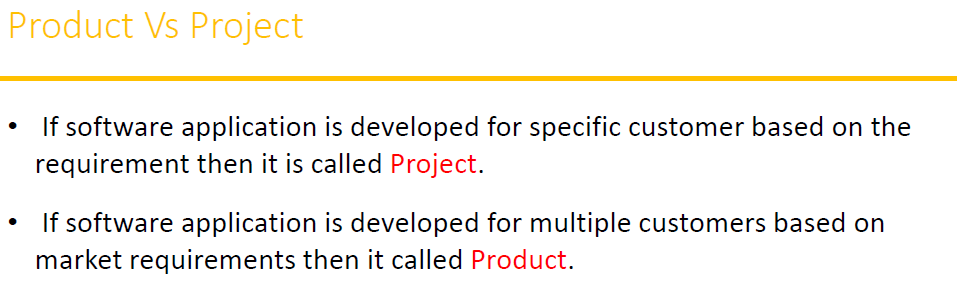
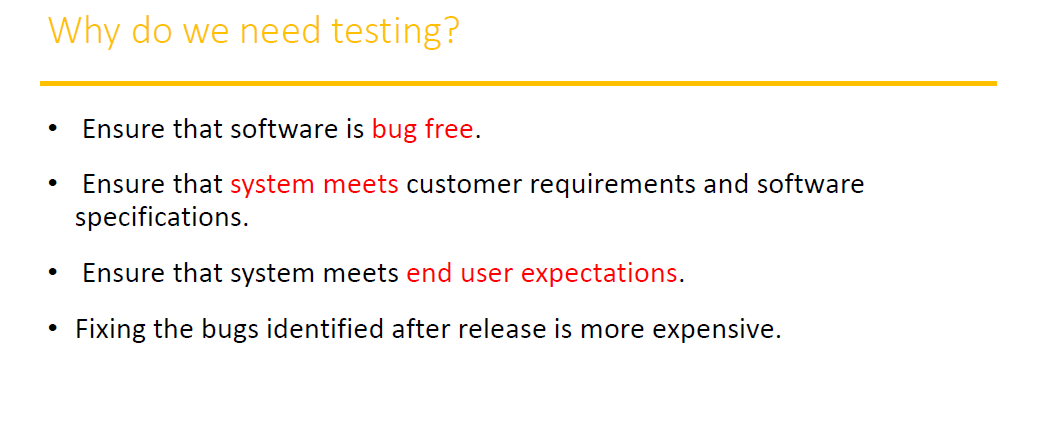
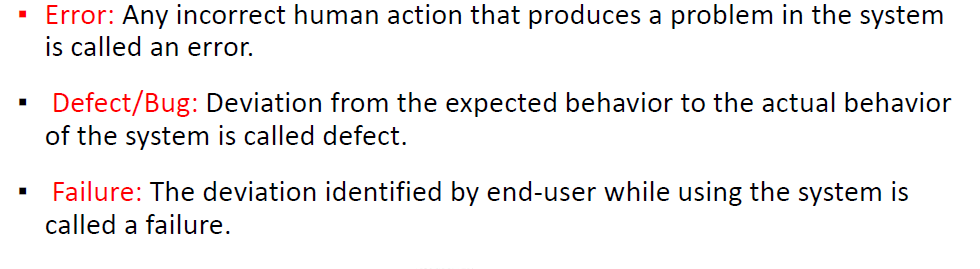
**Manual testing**

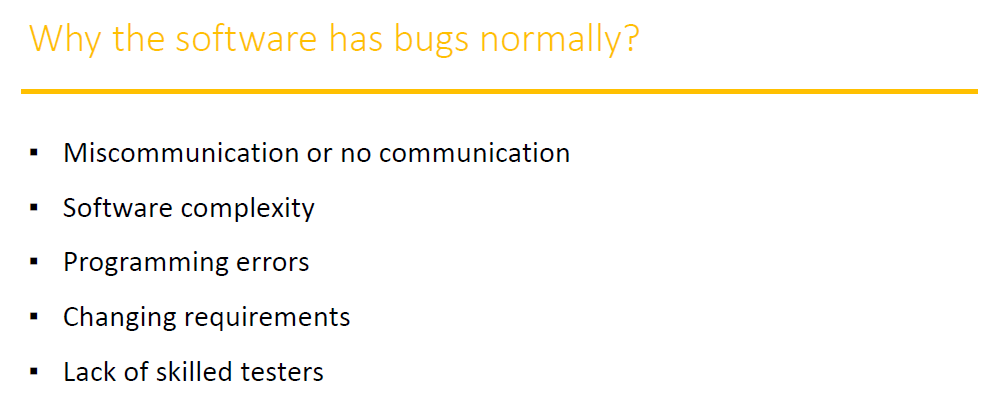
****

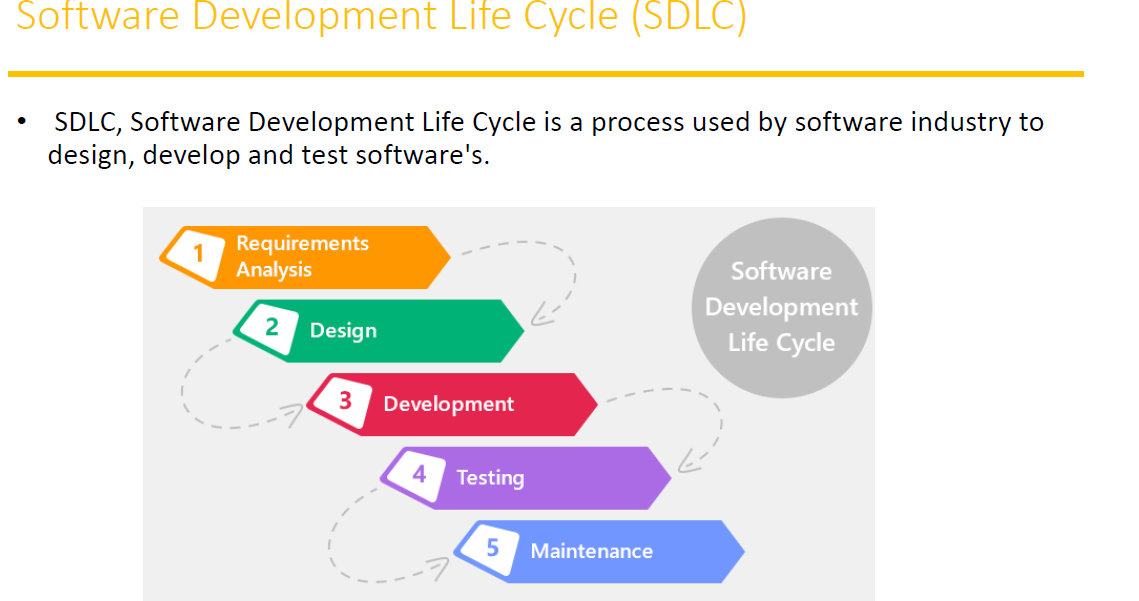
****

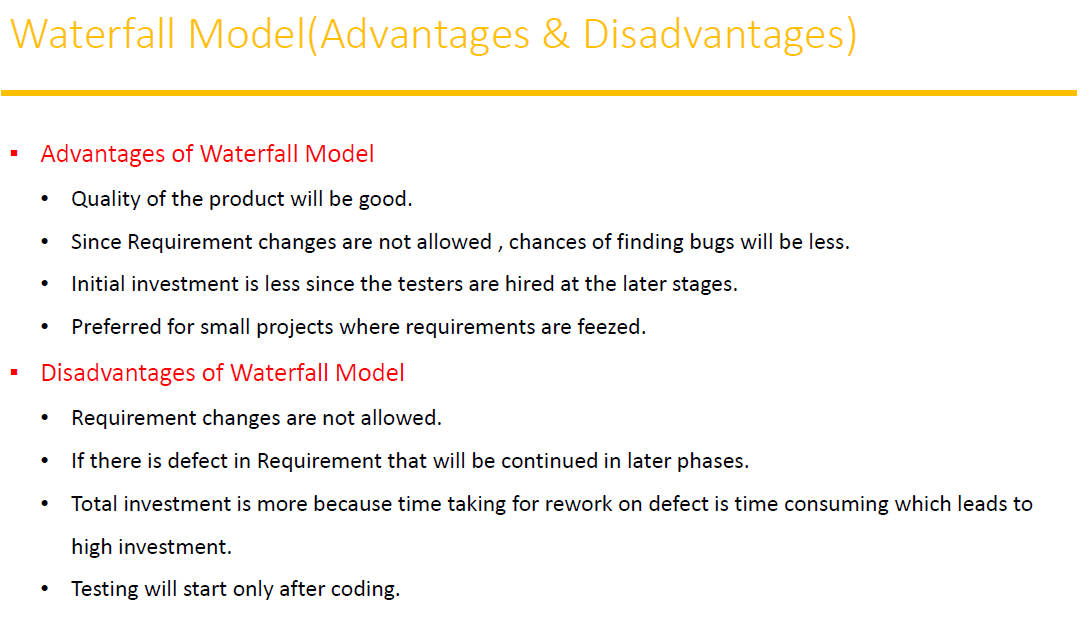
****

****

****

****

****

****

*The goal of a software tester is to find bugs, find them as early as possible, and make*

*sure they get fixed.*

**WHY TESTING IS NECESSARY?**

If any level of testing cannot declare that there is no defect in the product, then Why is it required at all?

Development people assume

that whatever they have developed is as per customer requirements and Will always work. But, it is

imperative to create real-life scenario and undertake actual execution of a product at each level of

software building (including system level) to assess whether it really works or not.

Developers may have excellent skills of coding but integration issues

can be present when different units do not work together, even though they work independently

One must bring individual units together and make the final product, as some defects may be possible when the sources are developed by people sitting at different places.

The primary role Of software testing is not to demonstrate the correctness of

software product, but to expose hidden defects so that they can be fixed. Testing is done to protect the

common users from any failure of system during usage.

Testing is a process Of demonstrating that errors are not present in the product,

This approach is used in acceptance testing where if the application meets acceptance criteria, then it must be accepted by the customer,

Testing gives number of detects present which indirectly gives a measurement of software quality.

More number of defects Vindicate bad software and bad processes Of development,

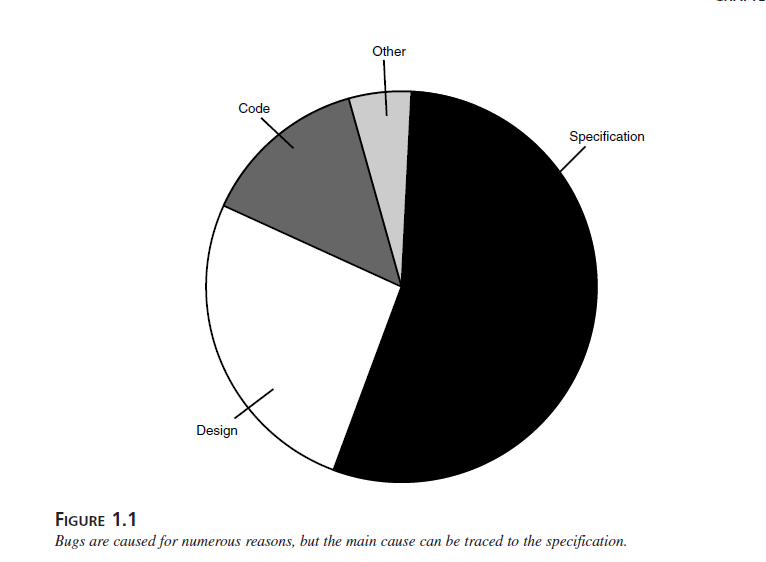
***software bug* occurs when**

1. The software doesn’t do something that the product specification says it should do.

2. The software does something that the product specification says it shouldn’t do.

3. The software does something that the product specification doesn’t mention

The number one cause of software bugs is the specification



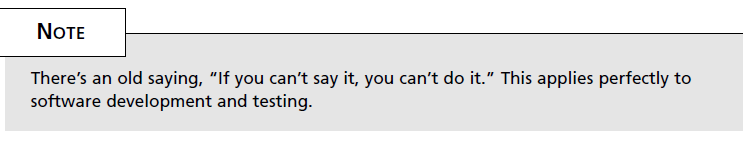
**The bugs are caused for many reasons but main one is Specification.**

specifications are the largest bug producer

it’s constantly changing, or it’s not communicated well to the entire development team.

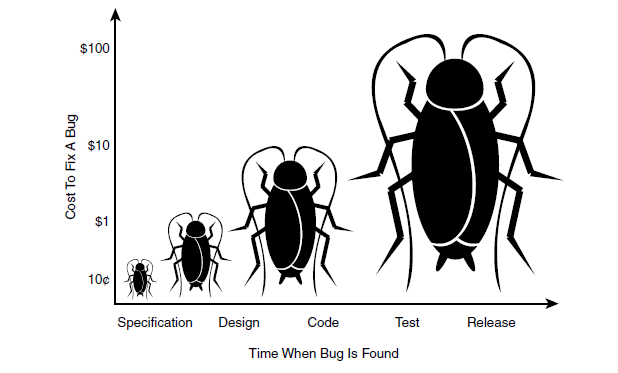
Planning software is vitally important. If it’s not done correctly, bugs will be created.

**The next largest source of bugs is the design if not designed well then bugs are occurred**



This means that wrong design ,, more the bug

The Cost of Bugs



The costs are logarithmic—that is, they increase tenfold as time increases. A bug found and

fixed during the early stages when the specification is being written might cost next to nothing,

or 10 cents in our example. The same bug, if not found until the software is coded and tested,

might cost $1 to $10.

But If a customer finds it, the cost could easily top $100.

Example

root cause of the problem was that the software wouldn’t work on a very popular PC platform.

If, in the early specification stage, someone had researched what PCs were popular and specified

that the software needed to be designed and tested to **work on those configurations**, the

**cost** of that effort would have been **almost nothing**.

If that didn’t occur, a backup would have been for the software testers to collect samples of the popular PCs and verify the software on them. They would have found the bug, but it would have been more expensive to fix because the software would have to be debugged, fixed, and retested.

*The goal of a software tester is to find bugs, find them as early as possible, and make*

*sure they get fixed.*

Software testing- it is process to test an application to find out error in it.

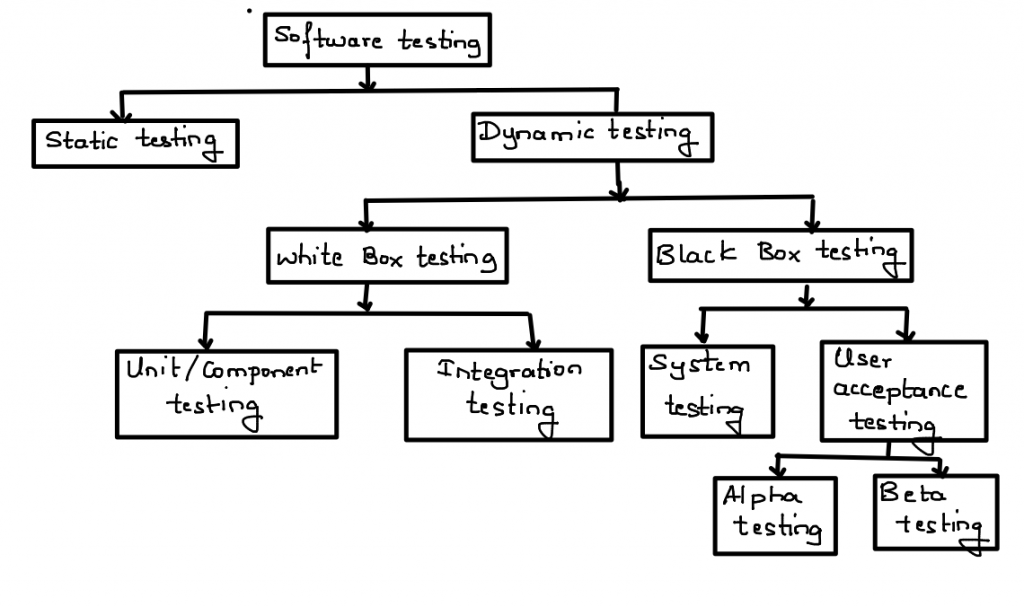
Checking the software is ok.

The goal of software tester to find bug.

 verifying and validating that a software or application is bug free

Testing Types

1. Manual testing :- Manual testing includes testing a software manually, i.e., without using any automated tool or any script
2. **Automation Testing:** Automation testing, which is also known as Test Automation, is when the tester writes scripts and uses another software to test the product. This process involves automation of a manual process. Automation Testing is used to re-run the test scenarios that were performed manually, quickly, and repeatedly.



**Static Testing Techniques**

Analysis of a program carried out without executing the program

Done during verification process i.e before development

As we know 85% errors found in desgin phase

Is code is tested in static testing ? “No” || the documentation is tested

Software development starts, continues, and ends with documentation

Early documentation → is used to defi ne the software to be built.

Later documentation covers → the software training, installation, and operation (user guides)..

Static = not while running

The primary goal of static testing is reduce defect by reducing defects in the

documentation from which the software is developed

**Review :** is type of static testing **|** done before execution

Review is a process or meeting during which a work product or set of work products, is

presented to managers, users, customers, or other interested parties for comment or

**walkthroug Review :**

 It is not a formal process

 It is led by the authors

 Author guide the participants through the document according to his or her thought process to achieve a common understanding and to gather feedback.

 Useful for the people if they are not from the software discipline, who are not used to or cannot easily understand software development process.

**Insepction Review:**

It is the most formal review type

It is led by the trained moderators

During inspection the documents are prepared and checked thoroughly by the reviewers before the meeting

**Informal Review**

Unplanned and Undocumented

**Technical Review**

Documented || Defined fault detection process || Includes peers and technical experts

No management participant

Walkthrough |not Formal

Insepction | formal| finding fault

improve process

Informal review Unplaned and undcoument

Technical Review , falut process

**Dynamic Testing Techniques**

The process of evaluating a system or component based upon its behaviour during execution.

System Testing

Black Box Tesing (Functinal Testing)

UAT Testing

High Level ( Main Testing by Tester) = System + UAT (Tester)

Low Level (Programer ) = Unit + Integration

Unit Testing

White Box Testing (Non Functional ):

Integration Teting

**Levels Of Testing**

1. Unit Testing :- in unit testing individual component of software tested. The purpose of

this testing is that each module is working properly

It focuses on the smallest unit of software design

(done by developer by using sample input and observing its sample output)

Eg. In a program we are checking if loop, method or

function is working fine

1. Integration Testing: in integration Testing individual units are combined and tested as group (developer)
2. Top-down
3. Bottom-up
4. Sandwich
5. Big-Bang

Main Purpose of Integration Testing :

To check moudules are communicating each other as DFD Data Flow Diagram which is specified in TDD ( Technical Document Diagram )

1. System Testing : in this testing we can test whole application (complete / integrated software is tested) done by tester
2. Acceptance Testing : a level of software Testing in which software is tested for user acceptance

UAT done at client location where software is actually used

→ Alpha Testing : done by tester in company in presence of customer

→ Beta Testing : done by customer to check software is ok , satisfy requirement

**Unit Testing**

White box testing >> |internal logic | developer

**Integration Testing**

**System Testing**

Black box testing → | no need code |only functional |tester

( Actual Testing | Tester)

**User acceptance UAT**

Testing Types

**Functional testing**

is the process through which QAs determine if a piece of software is acting in accordance with pre-determined requirements. It uses black-box testing techniques, in which the tester has no knowledge of the internal system logic. **Functional testing is only concerned with validating if a system works as intende**

Functional Tesing : Testing what system does

Functional testing is a type of black-box testing. “does this actually work?”

The ultimate goal of functional testing is to ensure that software works according to specifications and user expectations

input values

test cases

Compare actual and expected output



Eg. Login funcatinallity, registration funcationality

1. **Non Functional Testing :**

* Load testing
* Reliability
* The readiness of a system
* Usability testing

Eg. A practical example would be: checking how many people can simultaneously check out of a shopping basket

1. **Black Box Testing : (without code) High level**

Black box testing is that kind of software testing you can do when you do not have the source code,

just the executable code in hand

The testing is done without the internal knowledge of the products

1. **White box testing : (with code) low level**

Monitoring internal structure . check internal logic . done by developer

1. **Smoke testing** : - (Testing on newly released build → **compulsary requirement** )

It is first testing on newly released build … (Build Verification Testing)

Check → the deployed software build is stable or not.

Eg

QC → Build s/w → QA → Testing

Compulsory Testing → smoke testing

name

Compulsary

password

Remember Me

optional

Sign In

**Sanity Testing** : ( Testing on newly released build → check **Compulsory + optional** )

In above fig all filed are compulary then it is sanity Testing

**Retesting :** Testing functionality once again

Name

password

Enable keyboard

Defect Send to QC

Reset

Login

QA Test Again

**Regression Testing**

**(Re- running if code changes) :**

**It is overall testing when ever new change is occurred .**

Re-running functional + non Functional test

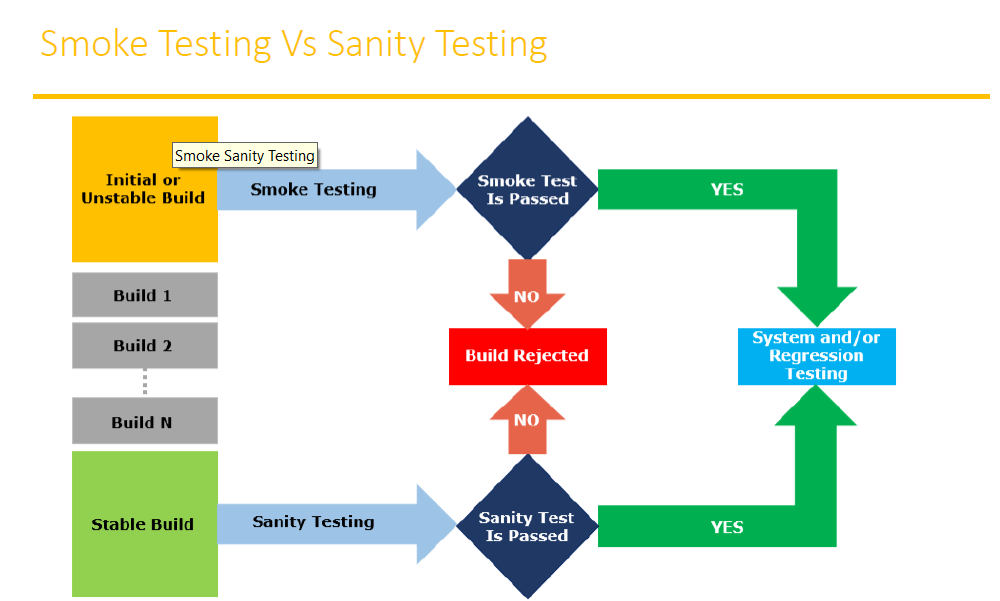
Code change , does not impact existing function.

After change code , is software works Ok.

Every time a new module is added leads to changes in the program.

This type of testing makes sure that the whole component works properly even after adding

components to the complete program.



Only compulsary feature test on new build

Check compulsary + optional because its stable build

**Static Testing :**

Review

Inspection

Walkthrough

**Test software without executing software** ( just test documents )

This test done for avoid bugs in early stage (review testing) look and feel

Why Static testing

→ Early defect detection and correction

→ To get fewer defect at a later stage of testing

Static technique has 3 types

**Review : review before development i.e simple document**

Read the document, correct or not

Document s/b correct and complete

Requirement review , design review , Test plan review

Review testing can do anybody , manager|developer|tester| coworker etc.

**Walkthorough :**

its is informal , anytime , not planned , done when ever required

author of document will explain to their team

**Inspection :**

Most formal , 3- 8 people in meeting

Proper meeting , schedule which is intimated by mail

**Dynamic Testing : A testing which is done after code development.**

The main purpose of dynamic testing is to test software behaviour with dynamic variables

dynamic testing requires code to be executed

static testing → just analyse the code, no need execution

Alpha Tesing : Its is final testing in development

Advantage : immediate solution is possible

Beta Testing : it is 1st testing in client side . it is also called user user acceptance testing UAT

Disadvantage : no immediate solution if defect is found

Installation Testing : providing required resources at client location

It is type of testing in which test engineer check deployment process is successful as per user guideline

Deployment document /user manual : it is document prepared by project manager

Usability Testing : checking application for user friendliness

Monkey Testing : used for game testing, used for random input

To check the application or system will crash

Portablity Testing : Developed application Shoud support multiple enviorment

Forced error Testing : to check valid error message will display

Exploratary Testing : When test enginer does not have idea of functinal testing then he is learnig through

exploring application

End to End Testing : We can check all internal componant for successful response

Internal componant like Client , Network, Server Database etc are working fine

Means Testing internal componant

Security Testing : Checking Secuirity of application

Relablity Testing : The Developed application Should Support Longer Duration i.e. Stablity

Audit : it is independent evolution of software .

Inspection : it is formal evolution of software

Concurrency Testing : multiuser Testing

Debugging : executing program line by line for finding errors.

***Some of the most popular SDLC models are:***

*→ no matter which module is used each has same phases*

\* Waterfall Model  
\* V-Shaped Model  
\* Incremental Life Cycle Model  
\* Spiral Model

SDLC : software development life cycle

It is process used by software company to develop , design , test software .

* + 1. Requirement analysis
    2. Design (blue print)
    3. Coding or development
    4. Testing
    5. Maintenance

**Waterfall Module** : it is old and traditional model

It is linear model i.e. steps 1 after another

Each phase must complete to start new phase ( i.e called one after another)



→ In waterfall module quality of product is good because every phase has clear

Documentation

→ SRS (system requirement specification ) not changed hence no bug

→ Initial investment is less because no tester involved

→ no changes in middle

→ testing will start after coding

**V shaped Model** : verification and validation

Verification → done before development → check we are doing correct ?

Verifying document >> because no software ready

Verification = before s/w = static

Review | walkthrough | inspection

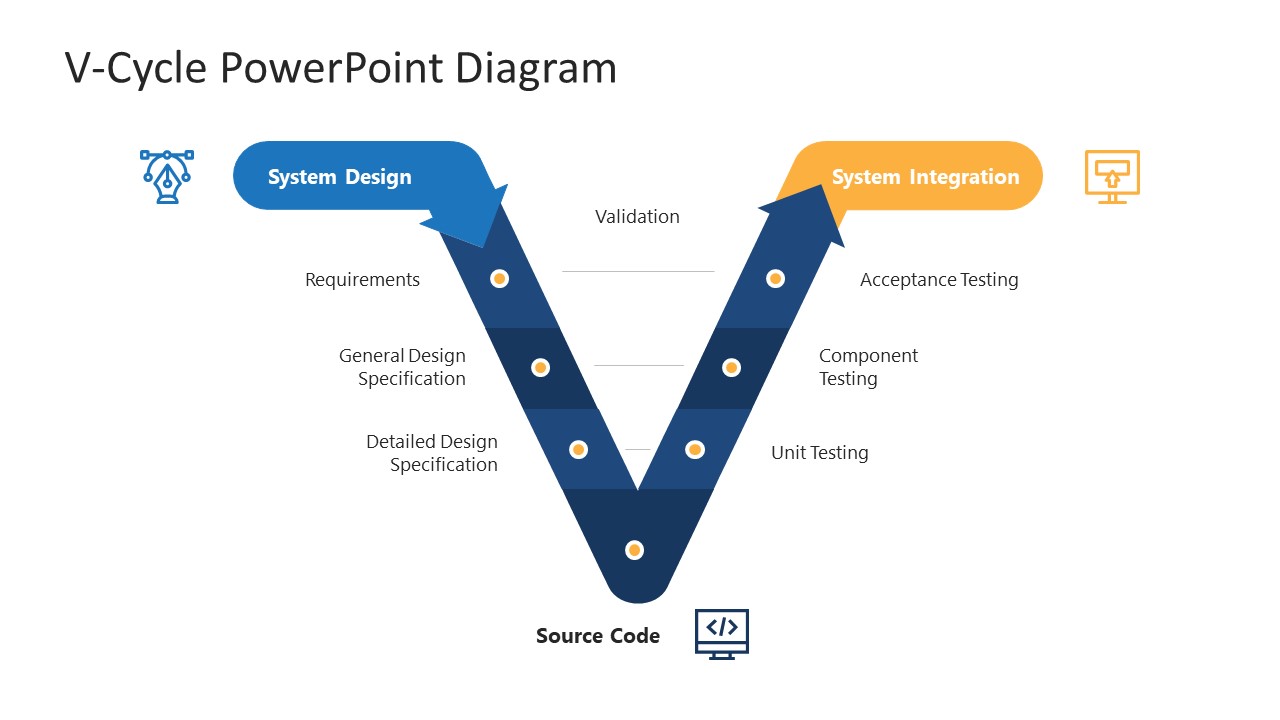
Static Testing = verifying doc

Validation **→** Actually testing software → done after software ready → done Right ?

Product is ready just check ok or not

Validation = after s/w = dynamic

Dynamic Testing = unit | integration | system | user acceptance



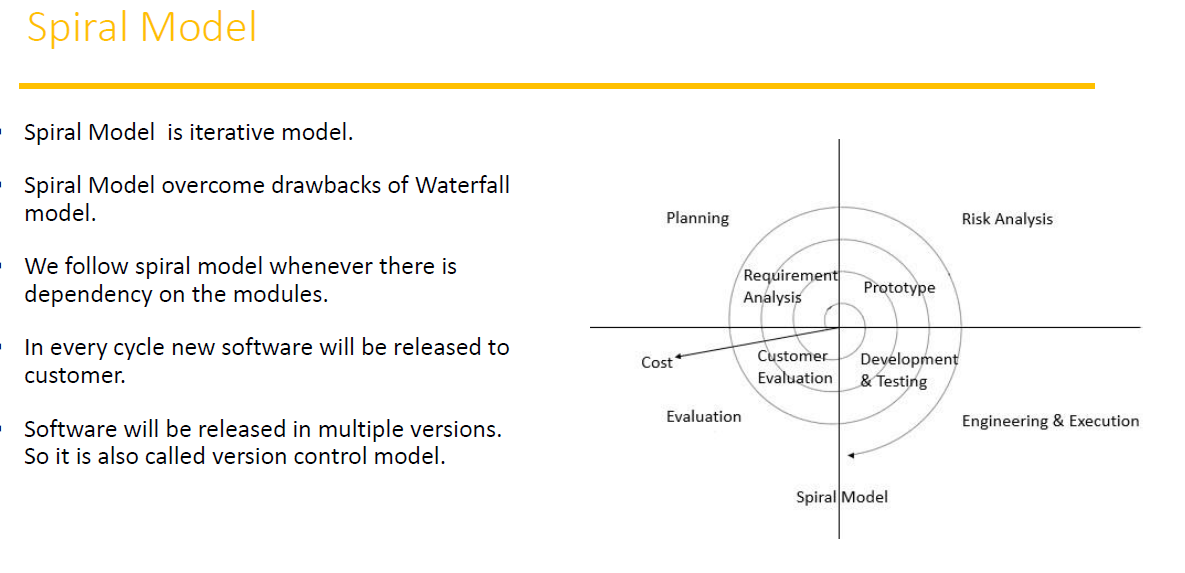
System design = before development = verification = static testing = verify documents

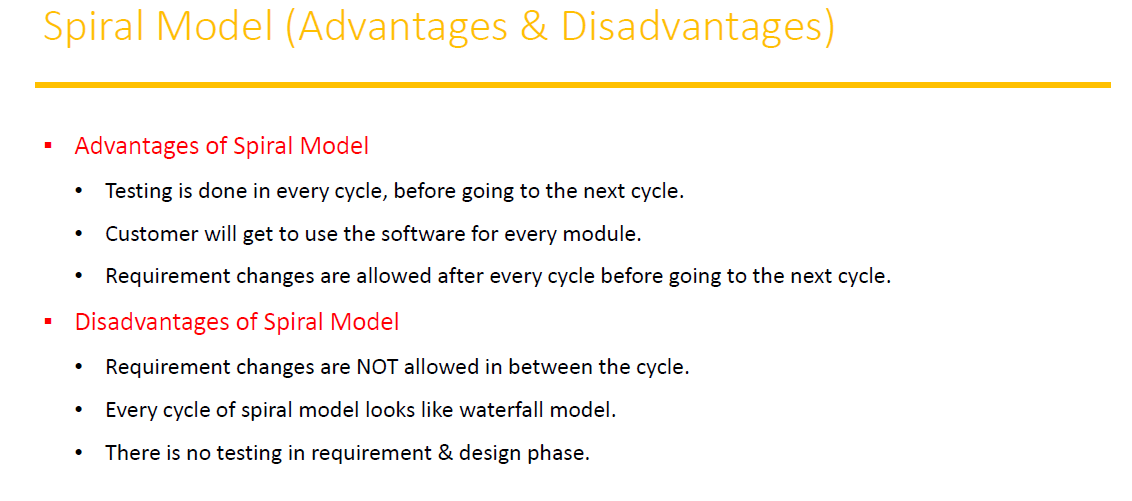
System Integration =after devep = validation = dynamic testing = unit | integration | system | UAT

In V model = testing is involved every phase

Disadvantage : more documents

**Spiral Model**

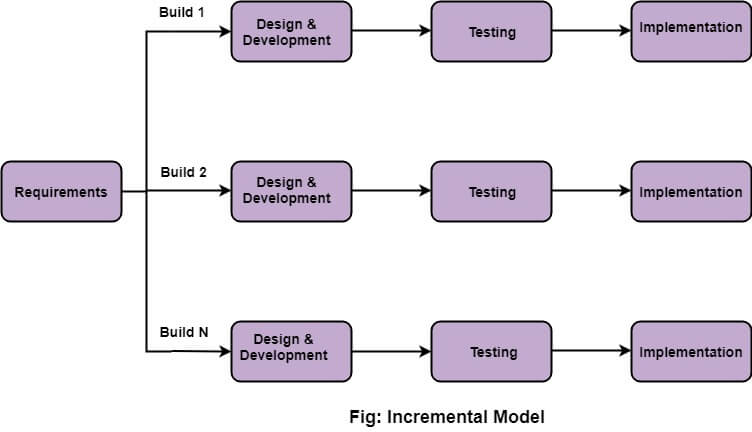




**Incremental Model** : requirement are divided into multiple module each module goes through

SDLC phase i.e. analysis, desing, coding , testing , maintenance

Requirement → module 1 + module 2 +…… module n



When use incremental model :

→ A project has a lengthy development schedule.

→ When the requirements are superior.

Disadvantages :

 Need for good planning  Total Cost is high.

**QA | QC | QE**

**QA = Highest Possible quality**

quality assurance | process related | high level management

Process designed by QA | Responsible for highest possible quality

**QC = quality control | product related | actual tester**

QA is process orientated | QC product orientated (that work on product actual tester )

QA = responsible for preventing defect (involves all phase i.e. design coding ..)

QC = responsible for finding defect ( involves only testing)

QE = Quality engineer

Responsible for write code for testing ( automation engineer)

System Testing : GUI Testing | Usability Testing | Functional Testing | non functional

1. GUI Testing :

:- Testing GUI application , user interface testing

:- such as menus , check boxes , icon, images

:- not functional , just look and feel

:- check size and position of element

:- image quality, spelling check , alignment

:- Fonts are understanding or not

1. Usability Testing :

:- check the easiness of application

:- helping messages are display if user confuse

:- check user friendly application or not ?

1. Functional Testing :

:- check behaviour of application

:- check database testing (work with database ok ?)

:- error handling . display error message ok ?

:- calculation and manipulation

Eg. 5+5 = 15 ( user requirement ) => we follow this

5+5 = 10 ( math calculation)

:- check text box disable or enable as user requirement

:- Check database operation DML table, column , records etc

Database

White box

User form

UI Testing

Black box

Checking database operation

Black box testing + white box testing = Gray box Testing

1. Non Functional testing :

:- once functional testing done i.e. s/w work user requirement then do non

functional testing

:- performance testing

Load testing – gradually increase the load

Stress testing – suddenly increase the load (Eg. Online filling form)

Volume testing – how much data handle

:- security of software

:- recovery of application

: - Compatibility testing – work with all platform

End to End Testing:

:- testing overall application after including all module

Eg : login → add customer

→ delete and edit customer

→ logout

Testing all function i.e. add delete edit and logout

**Test Case Design Technique:**

It helps better design and reduce the number of test case to be executed

Reduce data and more coverage

Black Box Testing

(Just design test case & run

1. ECP Equivalence Class Partition
2. BVA value analysis
3. Design Table All this are Test Case Design technique
4. State transition ( all this are in Black Box , Its Main testing)
5. Error guessing

**ECP → Equivalence Class Partition**

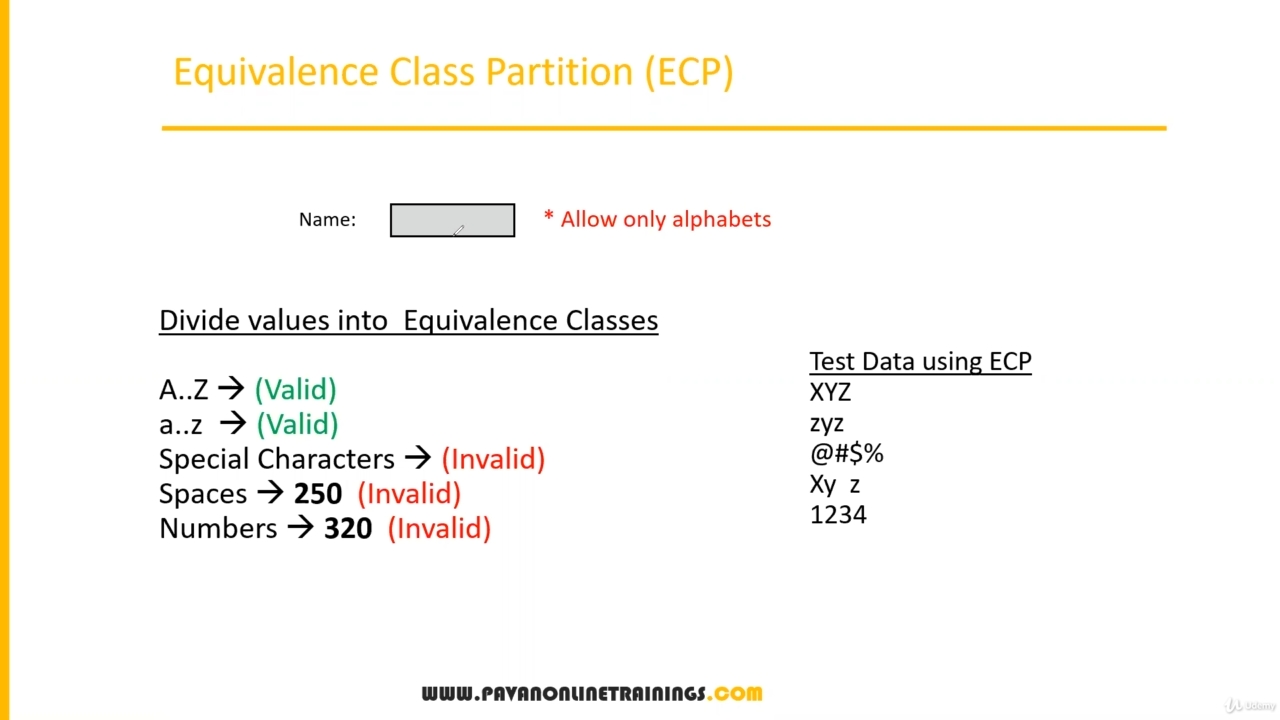
:- value check

:- classify | divide | partition data in → multiple classes to save testing time

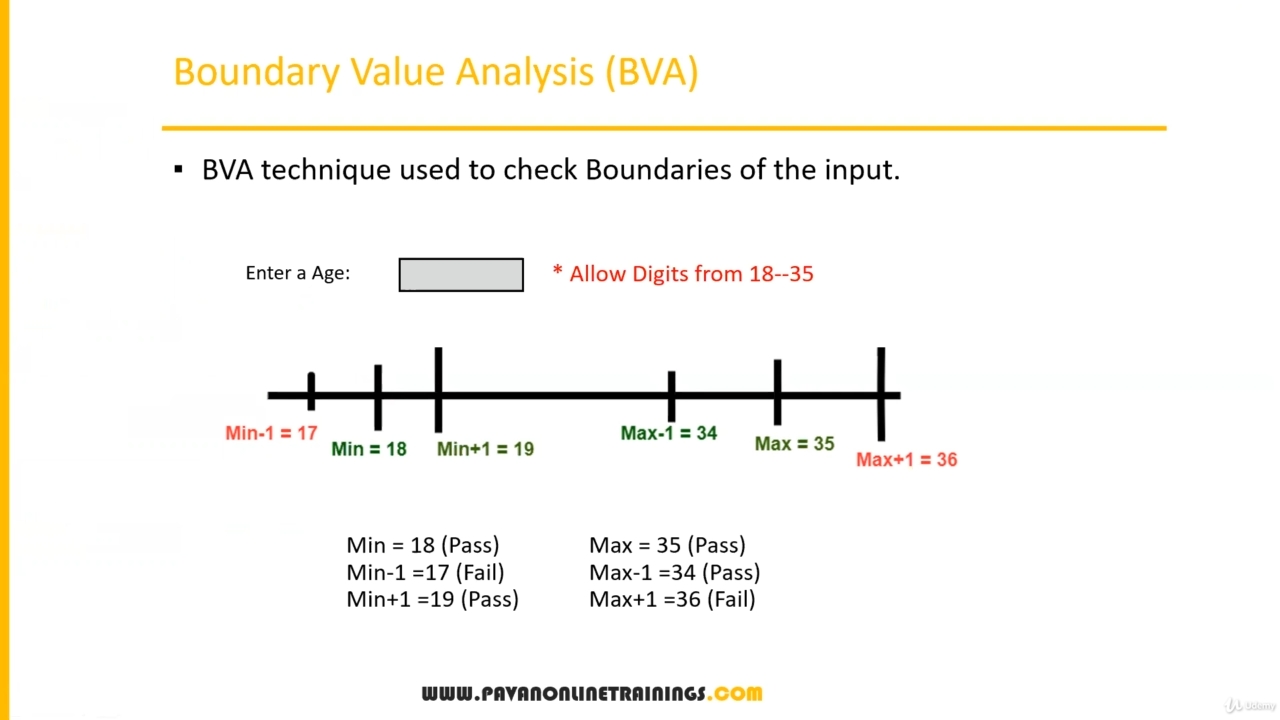
Eg.

Enter number \* allow digit form 1-500

|  |  |  |
| --- | --- | --- |
| Normal test data to check txt box value | Divide data in ECP | Final test data after ECP done to test case |
| 1 | -100 to 0 → -50 (invalid) | -50 |
| 2 | 1 to 100 → 30 (valid) | 30 |
| 3 | 101 to 200 → 160 (valid) | 160 |
| 4 | 201 to 300 → 250 (valid) | 250 |
| 5 | 301 to 400 → 350 (valid) | 350 |
| 6 | 401 to 500 → 420 (valid) | 420 |
| Up to 500 | 501 to 600 → 550 (invalid) | 550 |



**BVA: Boundary value Analysis:**



**Design table technique:**

this technique is used if we have more conditions and based on condition we have to perform the action

eg. transfer money from account to account

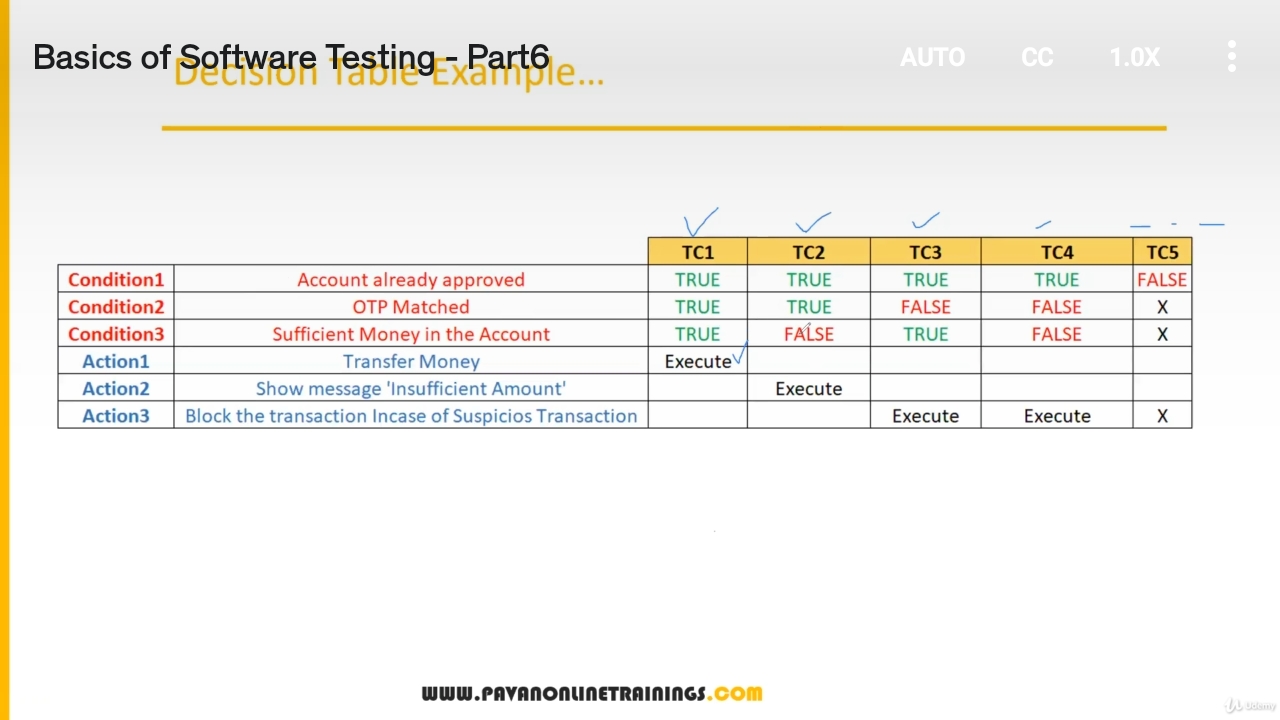
Condition :

1. Account no has to approved 🗸
2. OTP Matched 🗸
3. Sufficient money in account 🗸

If condition are ok then do action

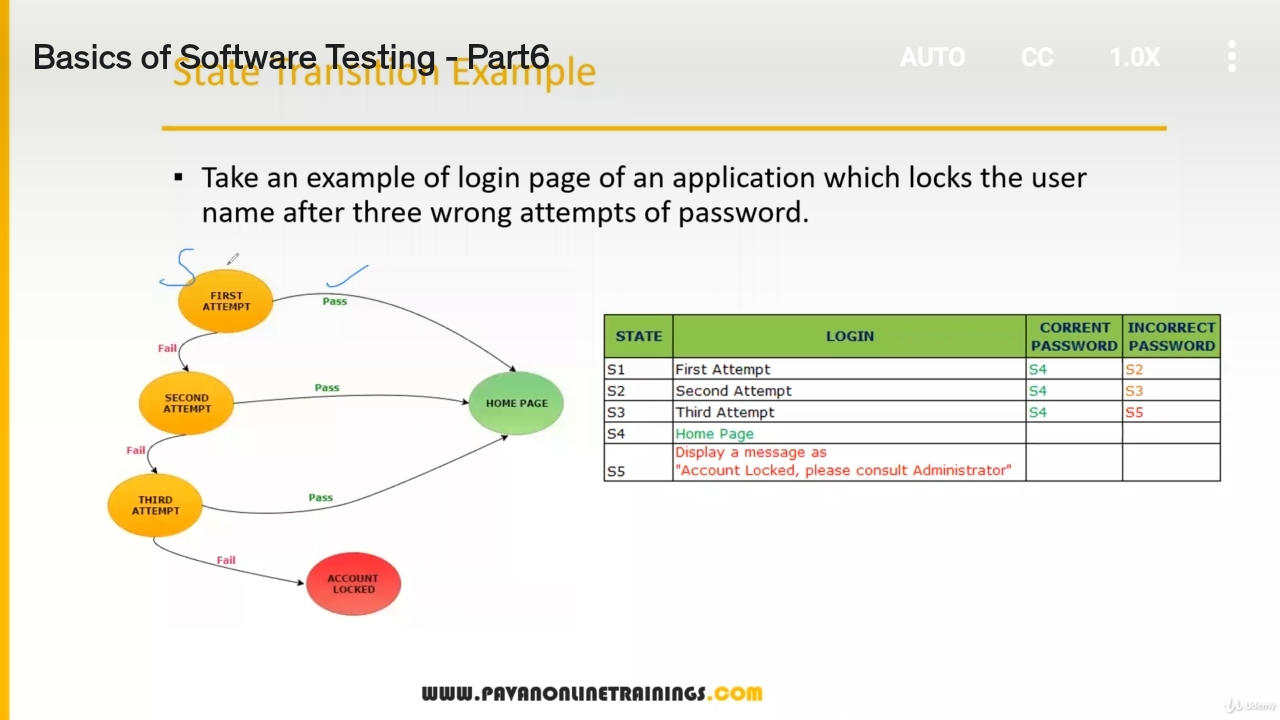
Action :

1. Transfer money
2. Show message insufficient money (if any)
3. Block if any suspicious activity



**State Transition Technique :**

Take action depends of on state



**Error Guessing Technique :**

:- no any specific rule

:- this test based on tester skill eg. Submit form empty

and guess error

**Test case Scenario** → simply the name of test what to test (name of Test)

**Test case** → how to test i.e. step

Group of steps that is to be execuated to check functionality

Eg. Test scenario = check functionality of login button

Test case = TC01 , TC02 , TC03 , … etc

Test suite → group of test case

Test suite 1

TC01, TC02, TC03

Test case group

Test plan

Test suite 2

TC01, TC02, TC03

**Test case group**

**Test case document**

:- Test case Id

:- Test case Title

:- Description

: Precondition

:- Priority

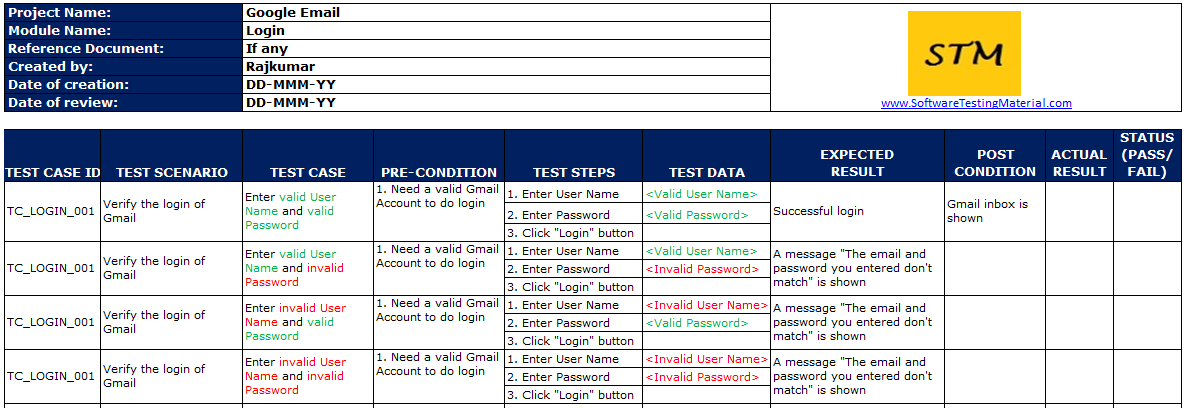
:- Request id

:- Steps/ Action

:- Excepted result

:- Actual result

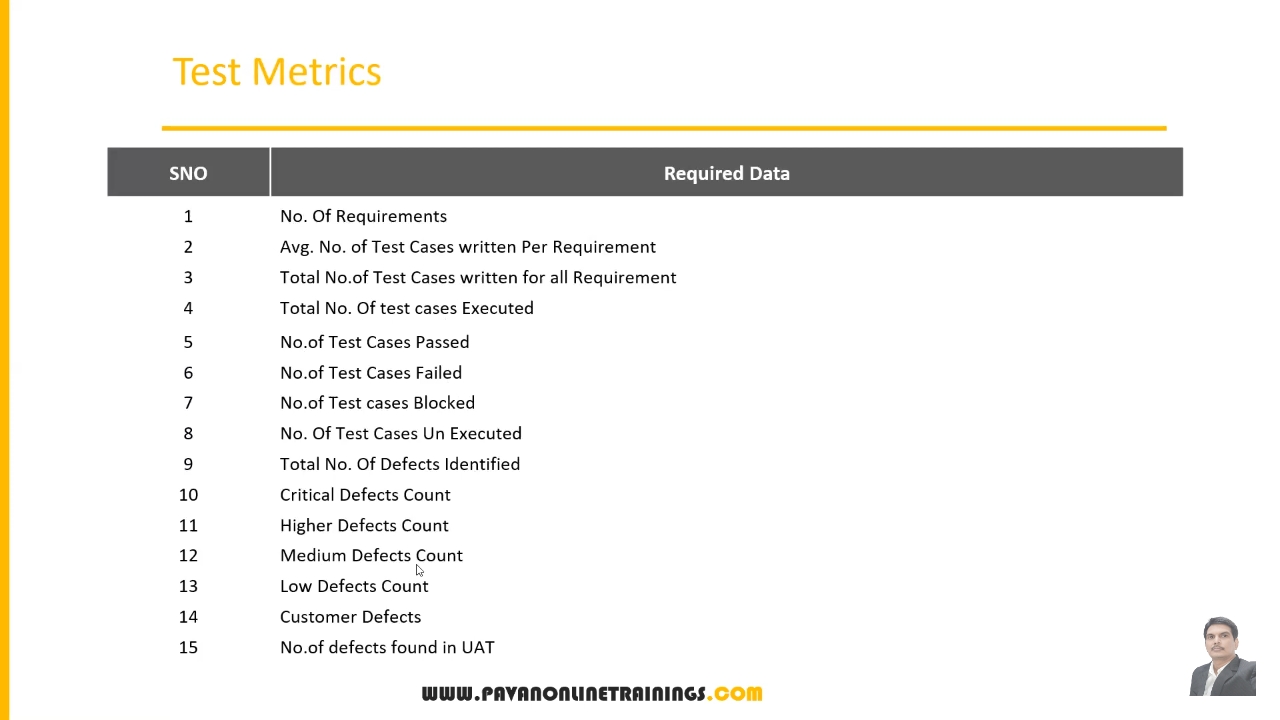
:- Test Data



**Requirement Traceblity matrix : (RTM)**

**:-**  Trace how many Test case are execuated or covered

: in simple keep track of test cases



**Test Case Execution:**

Executing test case based on test plan

Mark status Pass | Fail | Blocked

Reports defects in bug report

**Defect Reporting Tool :**

ClearQuest :- only bug report

Devtrack : only bug report

Jira → test management tool (track each activity)

Bugzilla → test management tool (track each activity)

**Defect Report Details :**

:- Defect id

:- Defect Version

:- Step : details of step along with developer what to do

:- date

: - detected by

:- status

:- fixed by in process | fixed

: severity – impact

:- priority – high| medium | low

**Severity Of defect** :- Blocker | Critical | Major | Minor

Seriousness of application

Testing engineer decides the severity level of the defect.

Blocker : this defect show application not processed

Critical : main function not working

Major : some undesirable behaviour eg. Email sent but msg not display

Minor : look and feel

**Priority of defect**: - High | Medium | Low

importance of defect

On which priority defect will be solved or fixed

P0 - High fixed immediately in same version

P1 - Medium fixed in next release

P2- Low next version

Manual Testing project: E commerce

:- Project introduction

:- Understanding and explore the functionality

:- Test Plan

:- Writing test scenario

:- Writing test cases

:- Environment setup and build and development

:- Test execution

:- Bug reporting and tracking

:- Sanity testing , smoke testing , regression testing

:- test sign off

E commerce project

* + Login
  + Search for product and item
  + Add them to cart
  + Do payment
  + Product will be delivered
  + Return the product
  + Etc.

**SRS document**

A **software requirements specification** (SRS) is a document that describes

what the software will do and how it will be expected to perform.

Refer SRS Document

E commerce project

:- Project information

:- Understanding the functionality of project

:- Test Plan → a detailed document of testing activity

:- writing test scenario

:- Test case and review

:- Environment setup and deployment for testing application

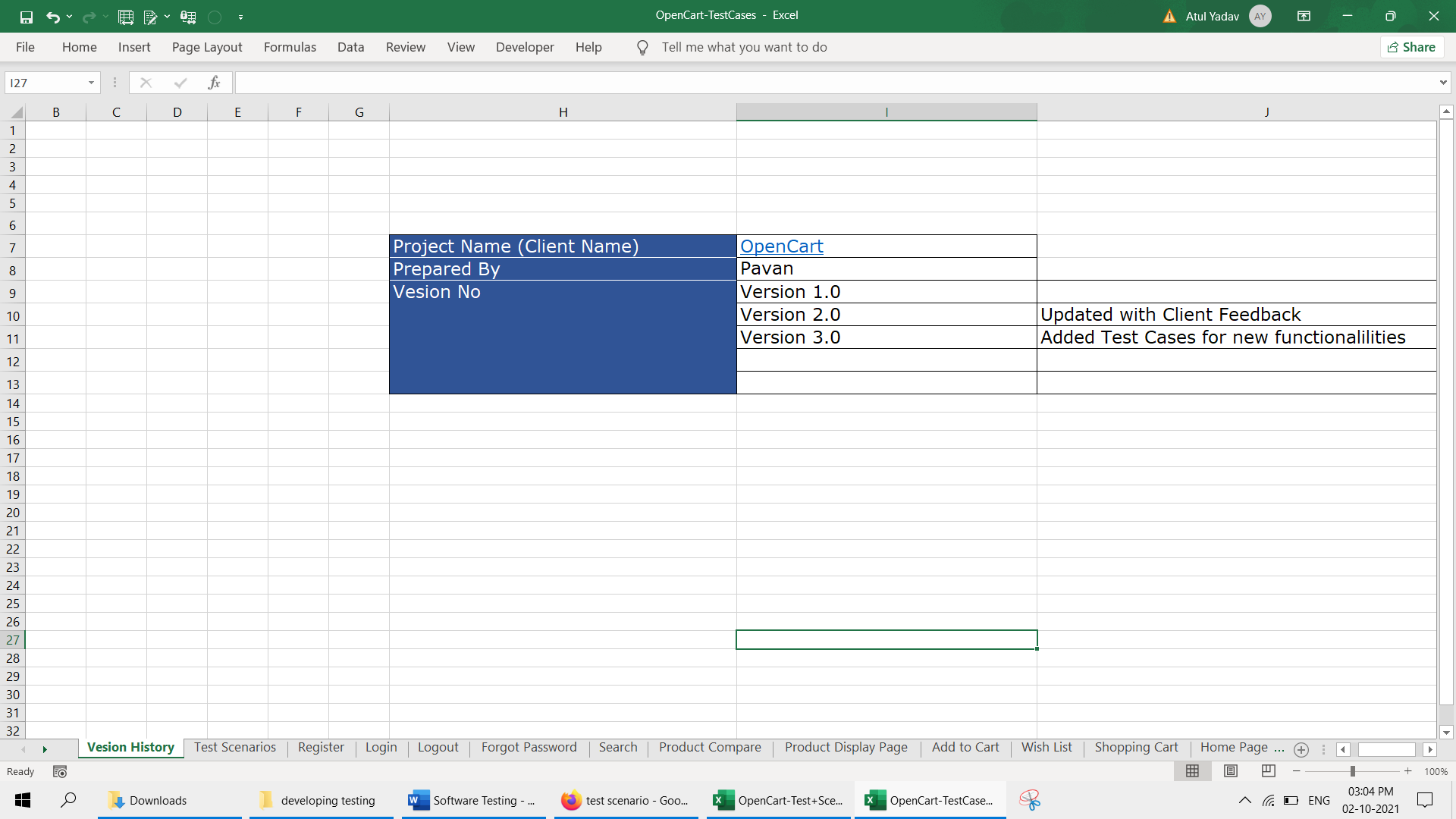
:- **Test Execution**

:- bug reporting and tracking

:- Sanity and regression testing

:- Test Sign off

Version page



Opencart.com project

1. FRS Document : → How Software works

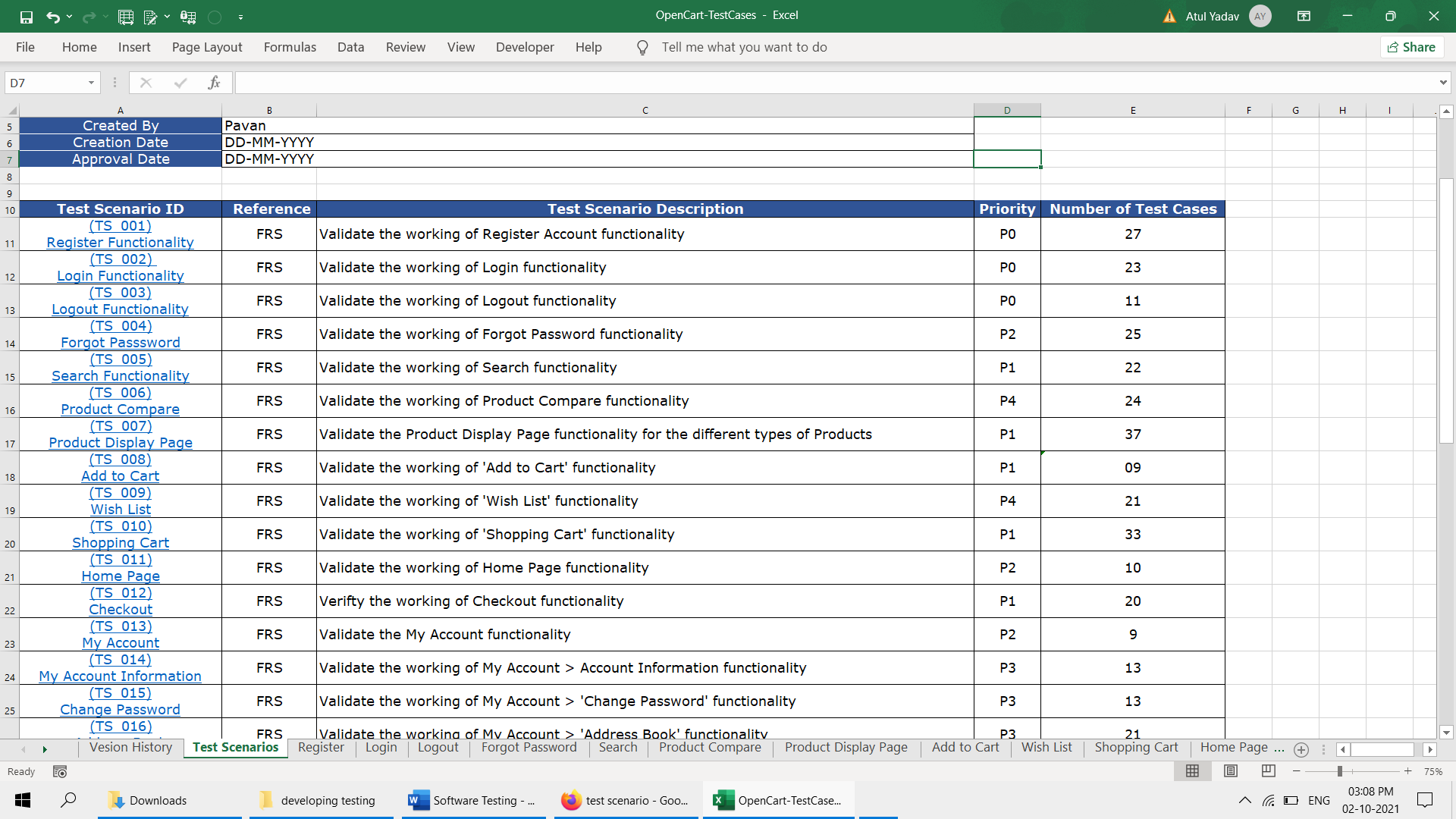
An FRS, or **functional requirement specification** is the document that describes all the functions that software or product has to perform

2 Test Plan Document → a detailed document of testing activity

3. Test Scenario Document : → Anything that can be tested is a Test Scenario.

→ Simply name of test case

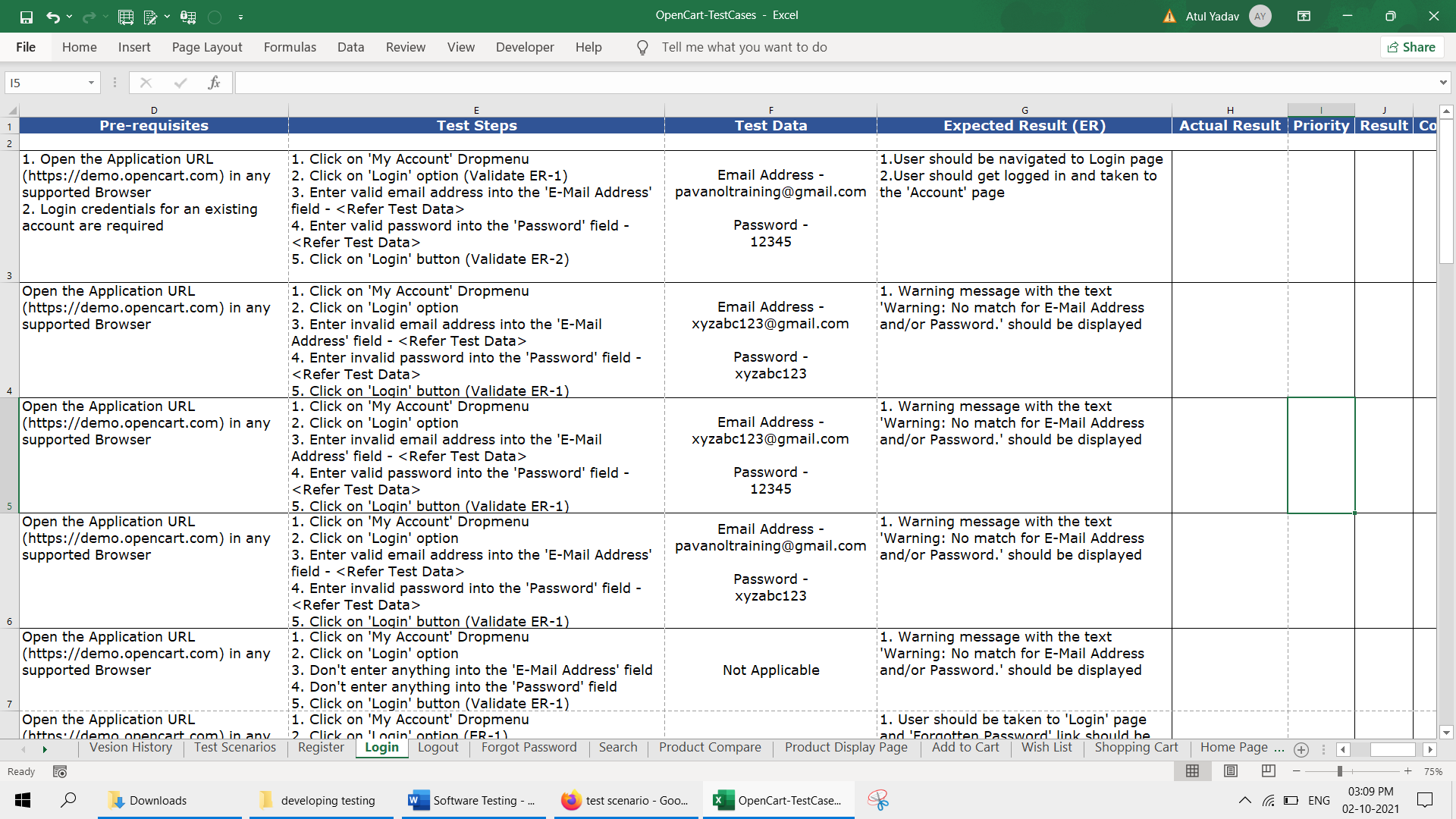
Test Scenario Document

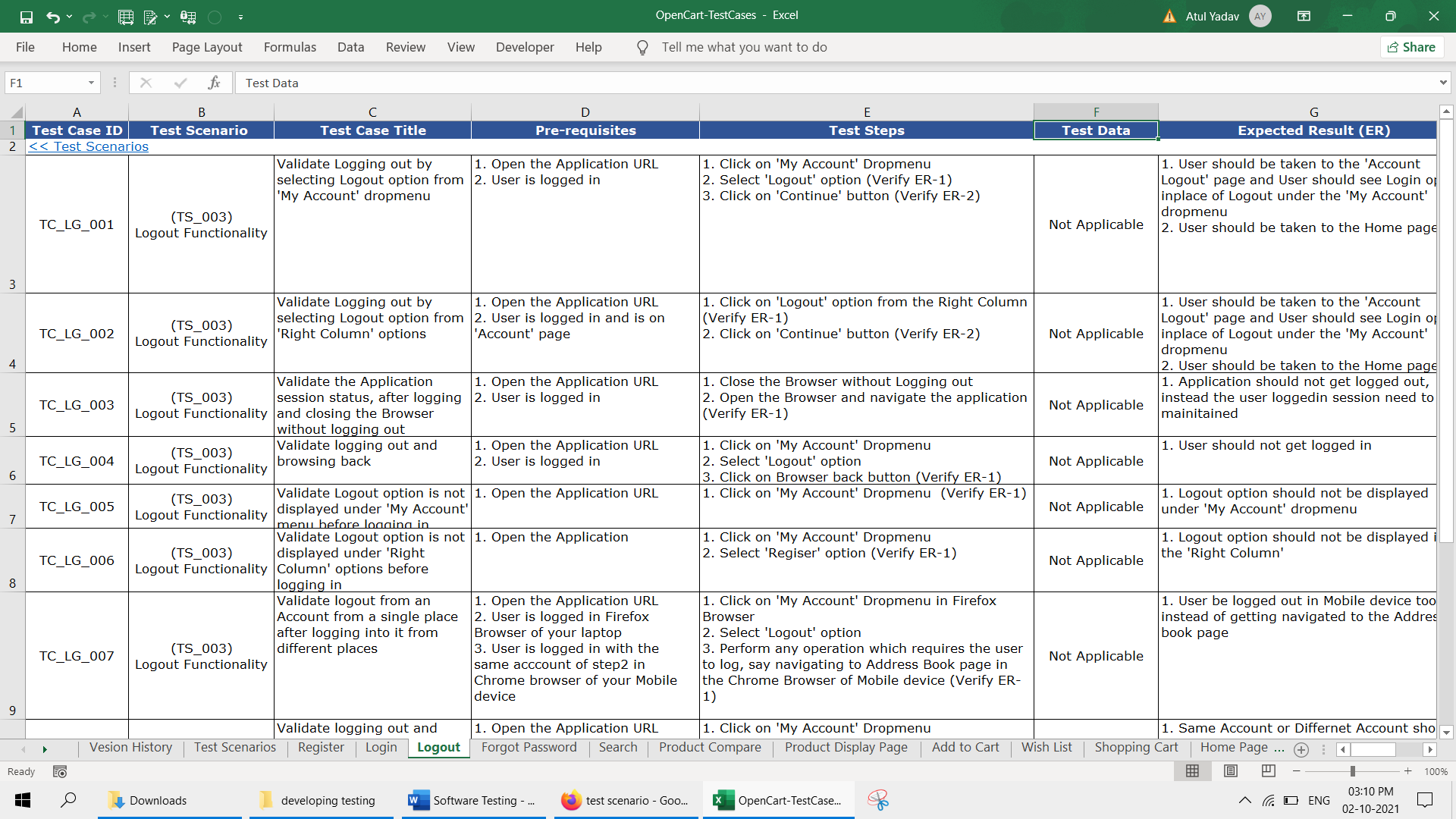


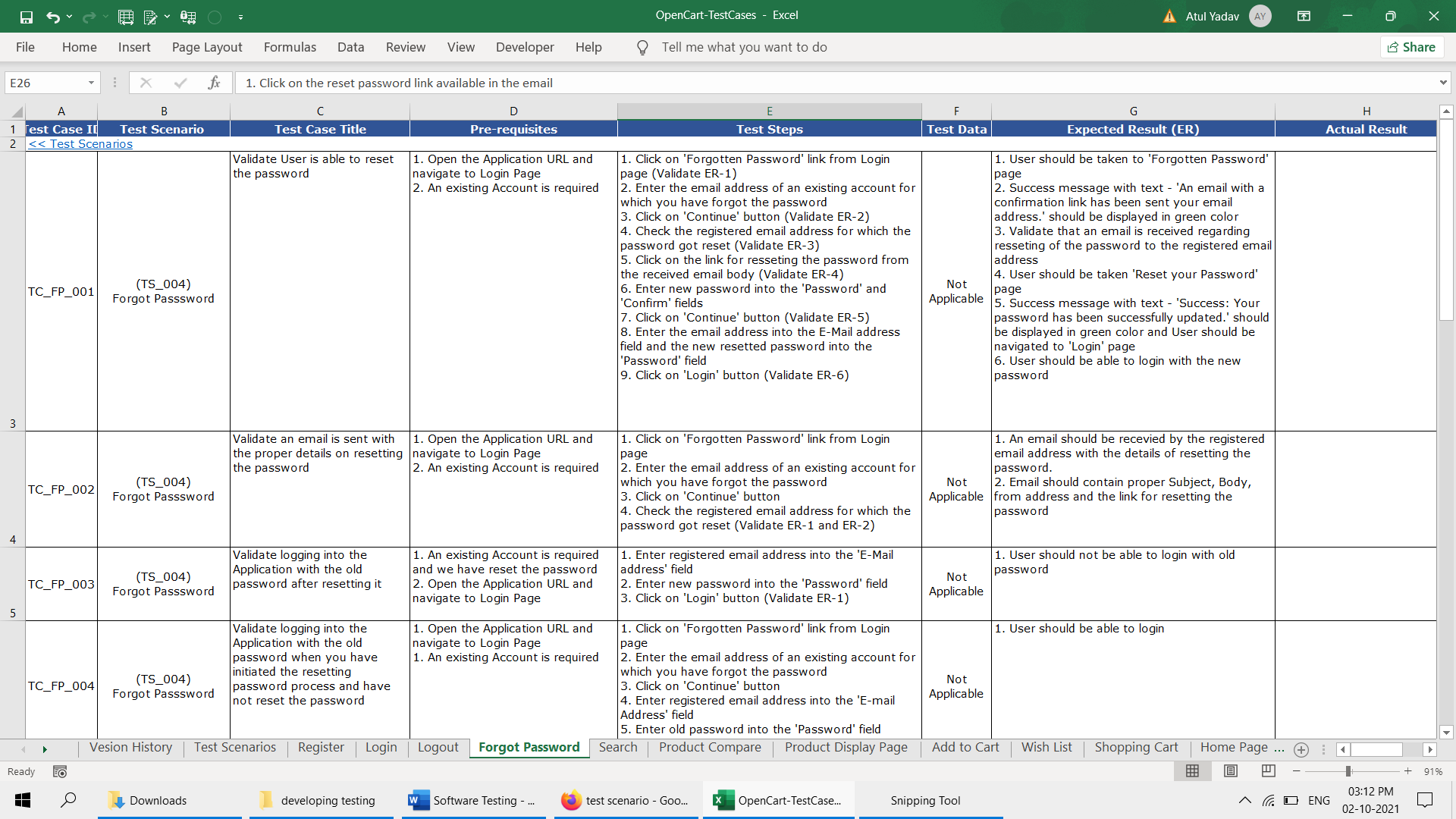
1. Test Case Design :

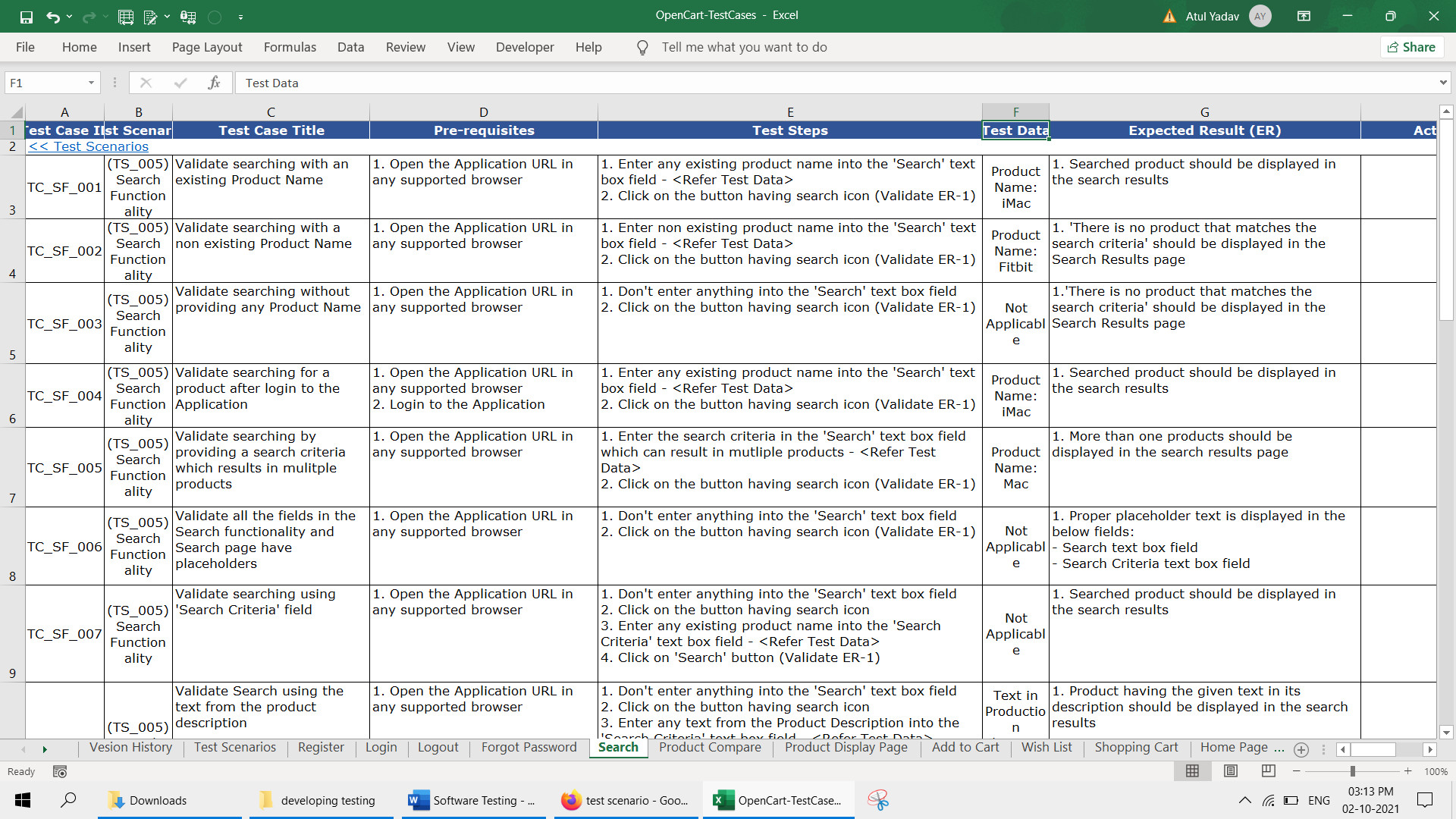
While writing test case refer the test Test Scenario and FRS Document

Test case template









All above are sample test case

After executing above steps i.e. test case update the field

Actual result

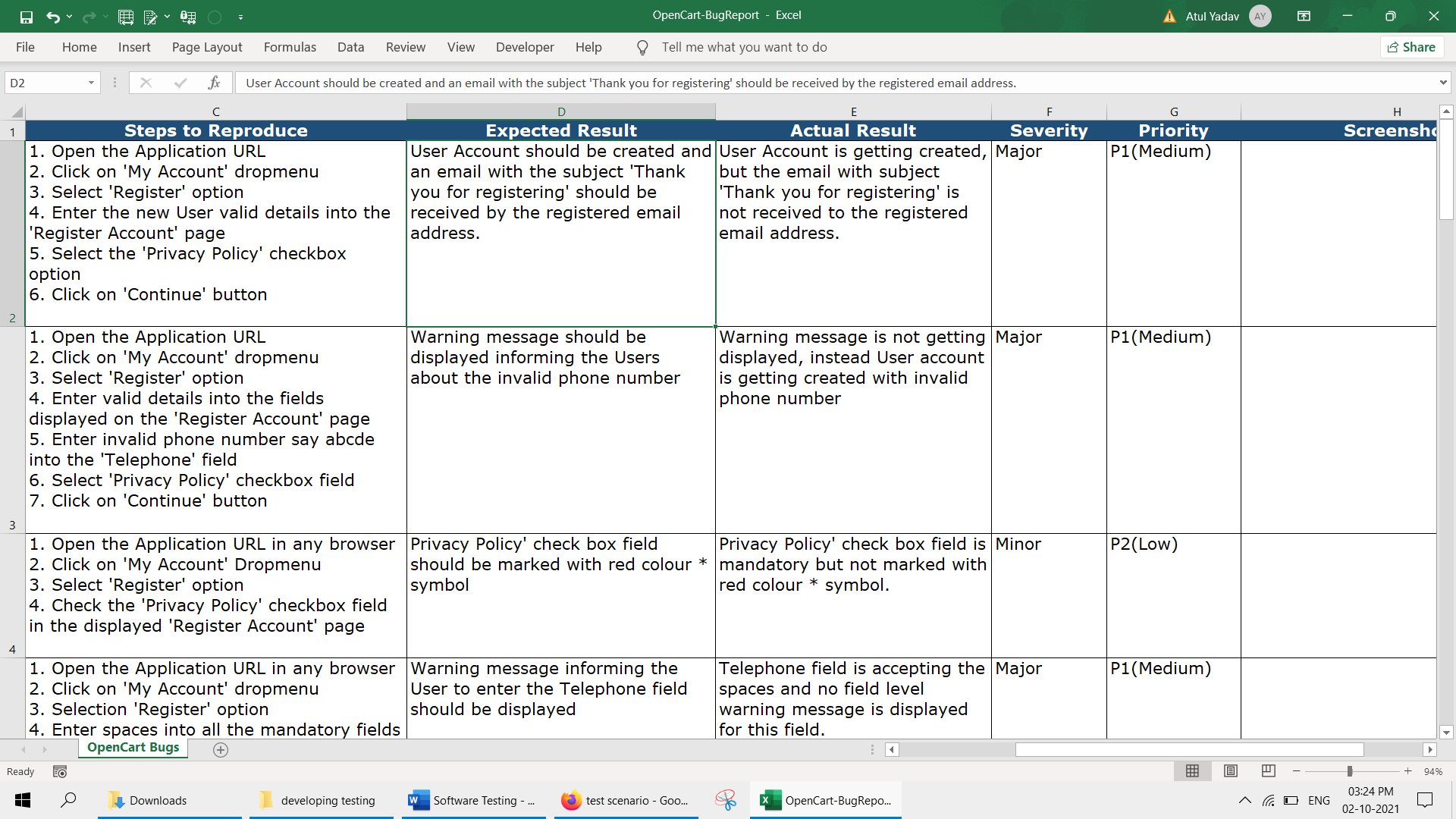
Result → pass or fail

Priority

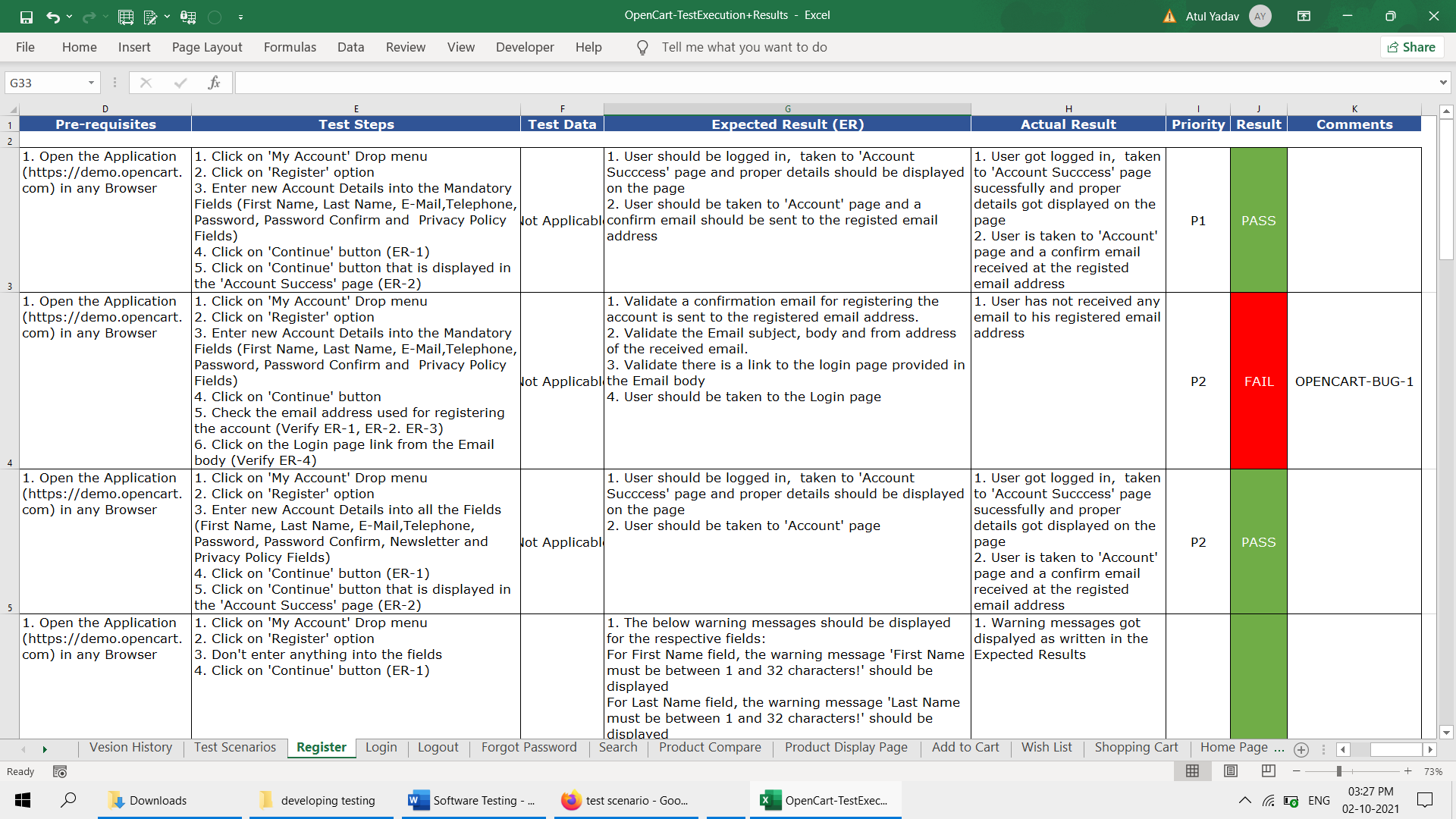
After executing test case maintain or update sheet RTM eg. No test executing or blocked etc

If the bug found in testing report it in bug tracking file

Eg.



Test case with result

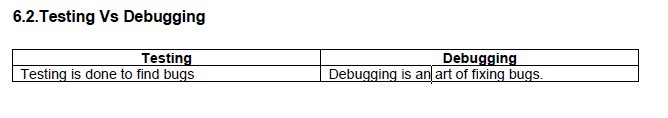


Big Bang Testing Approach

Big bang' approach involve testing software system after development work is completed. This is also termed \*system testing' or final testing done before releasing software to the customer for acceptance testing.

Big Bang == System Testing== Final Testing == Before Release

This is last part Of Software development as per waterfall methodlogy



• **Black Box testing:** Mainly perform by testers

• **White box testing:** Mainly perform by developers

• **Unit testing:** Part of White box testing

• **Acceptance testing:** This is the final testing done by Customer based on the agreements Load / stress / performance testing : Testing an application load capacity

• **Usability testing:** Testing to determine the user friendly ness of the application

• **Install / Uninstall testing:** Testing of full, partial, or upgrade install / uninstall processes.

• **Recovery / failover testing:** Testing to determine how well a system recovers from crashes, failures,

or other major problems.

• **Incremental integration testing:** Continuous testing of an application as new functionality is added

• **Ad-hoc testing:** Conducting testing without requirements

• **Comparison testing:** Comparing software weaknesses and strengths to competing products.

• **Alpha testing:** Part of UAT

• **Beta testing:** Part of UAT

• **Integration testing:** validating combined modules of an application

• **Functional testing:** part of black box testing

• **System testing:** part of black box testing and validating the system requirements

• **End to End testing:** similar to system testing

• **Sanity testing or smoke testing:** An initial validation of a New build or release

• **Regression testing:** validating the existing functionality of the application once new fixes added

• **Compatibility testing:** Testing an application in different environments.

**A bug is a issue or error in code or any environmental issue.**

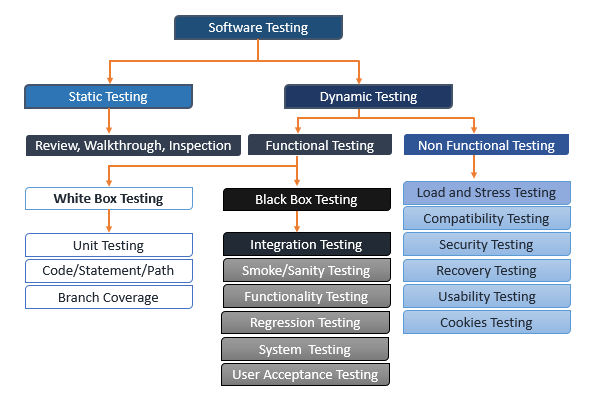
**What is Test Case?**

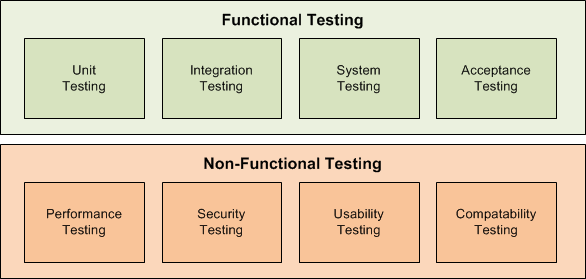
A test case is a detailed explanation of a scenario. Test case is a document which describes pre-condition, post-condition, test data, actors and navigation of the particular functionality.Each and every test case should have unique test name and test ID.

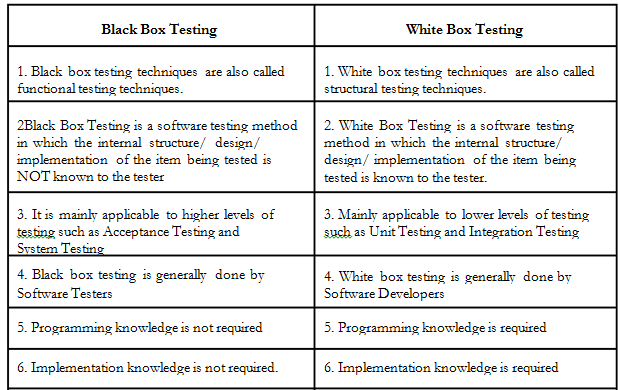
**What is Test Plan?**

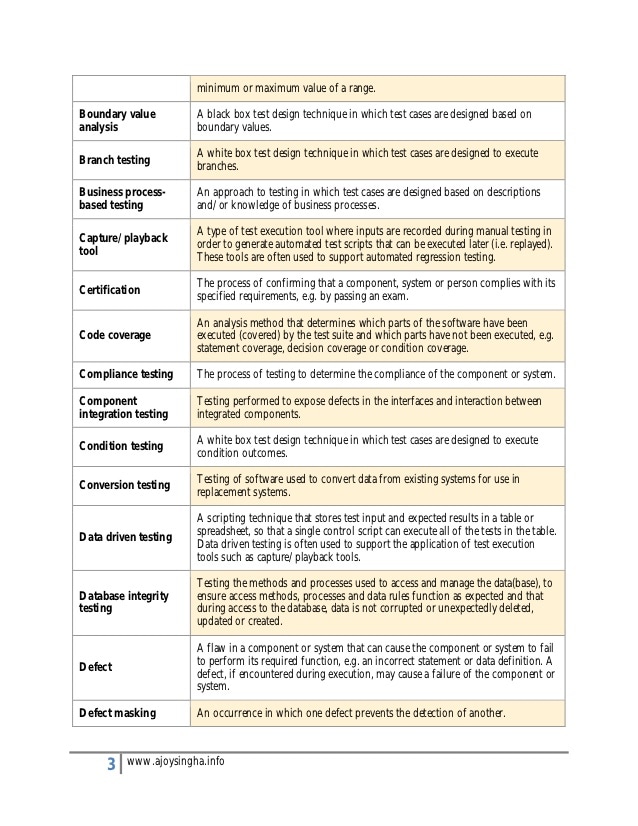
Test Plan is a document that explains what to test, when to test, where to test and when to complete

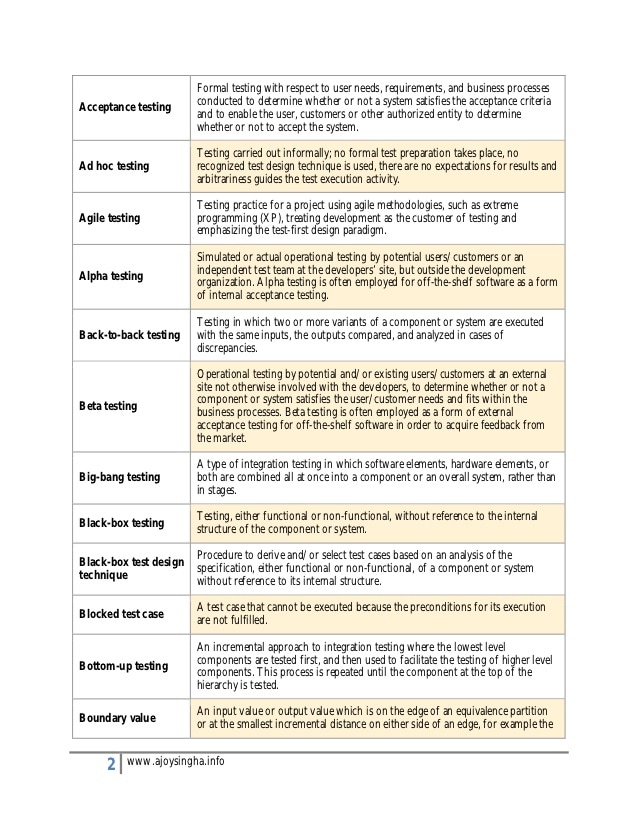
**3- Retesting is only done for failed Test cases while Regression is done for passed test cases**

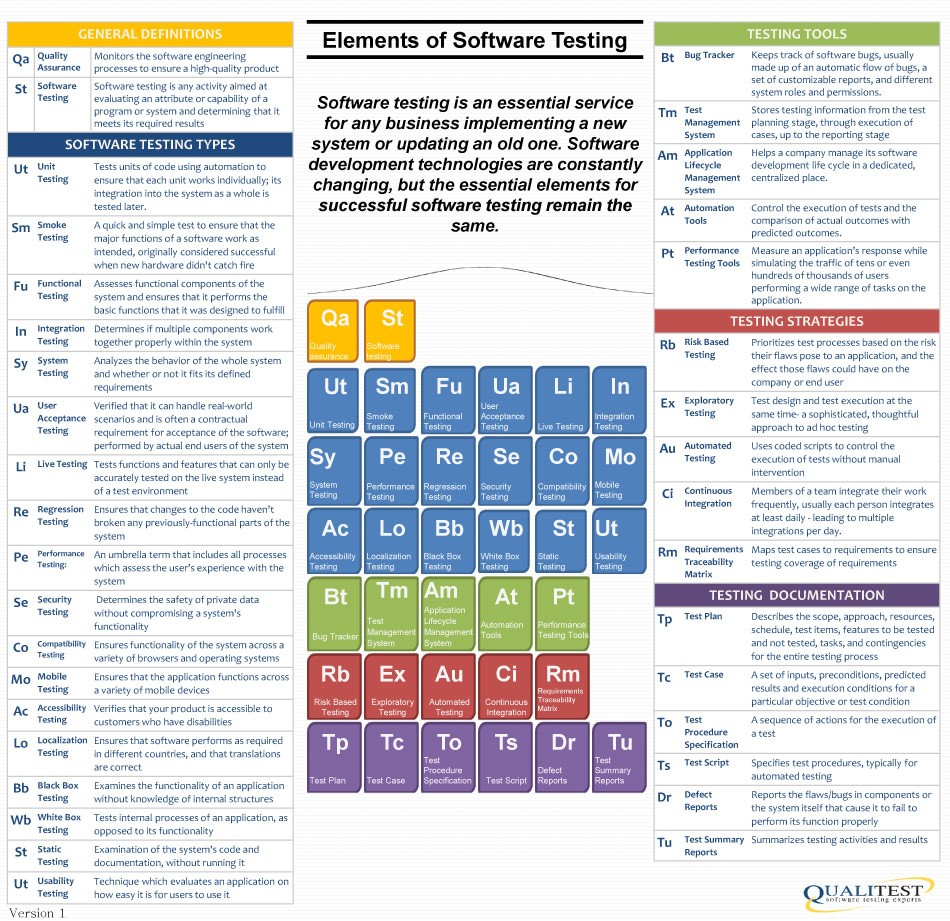


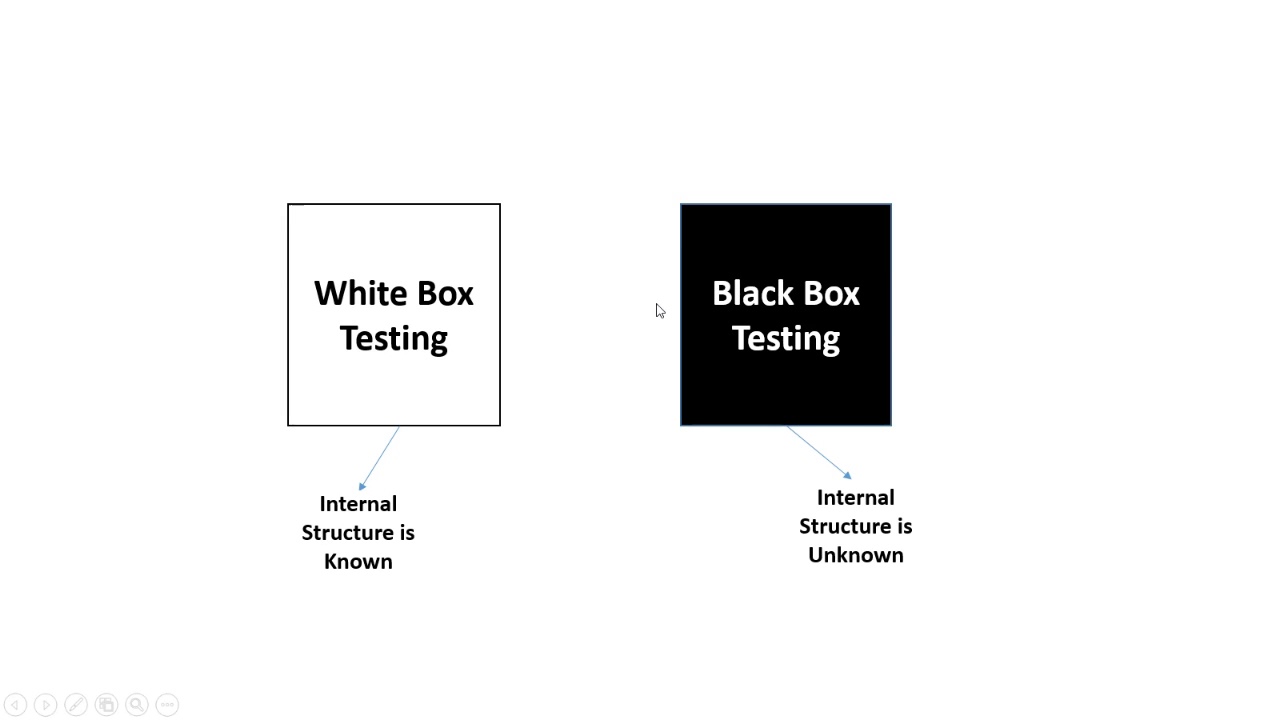


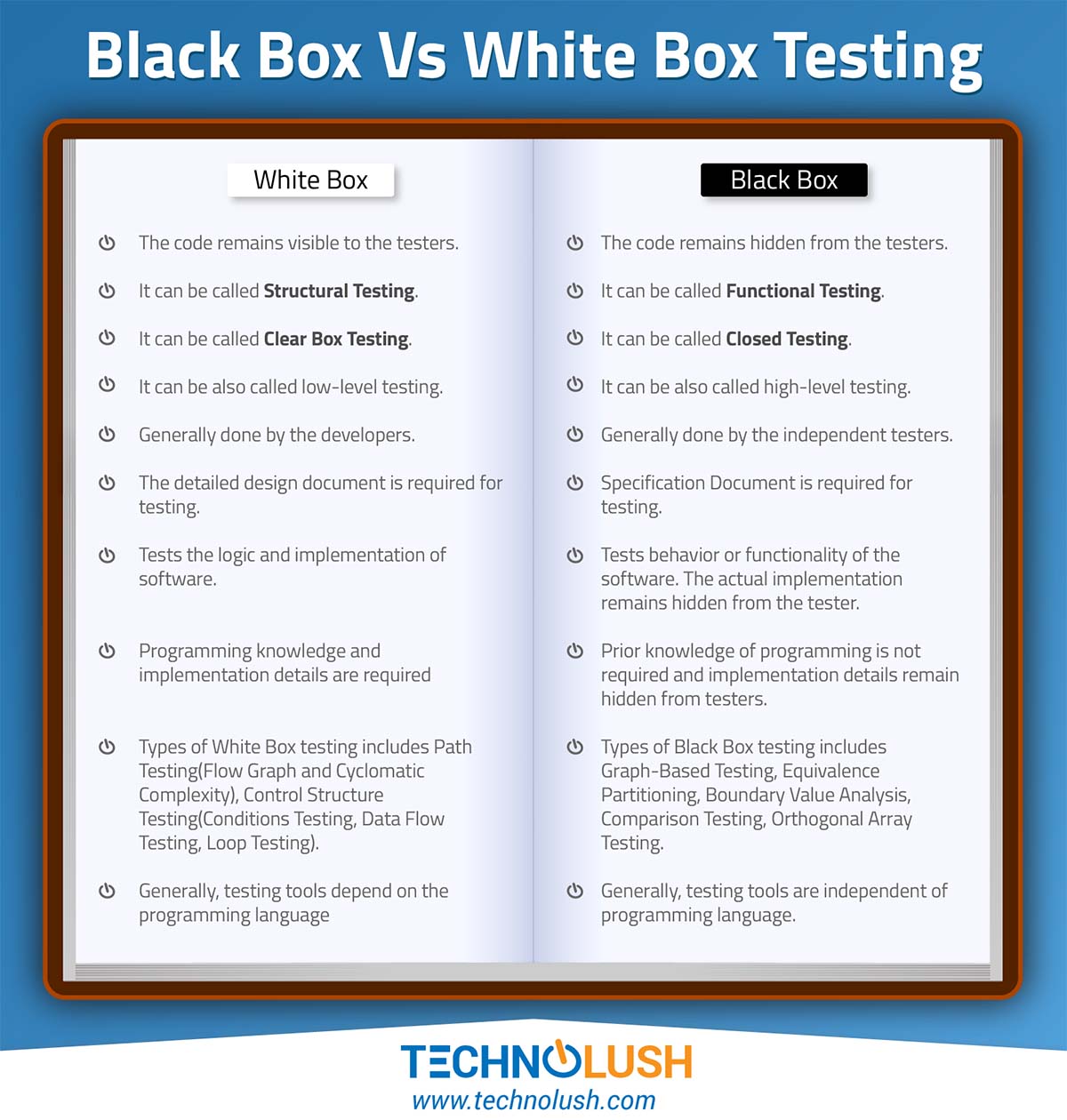


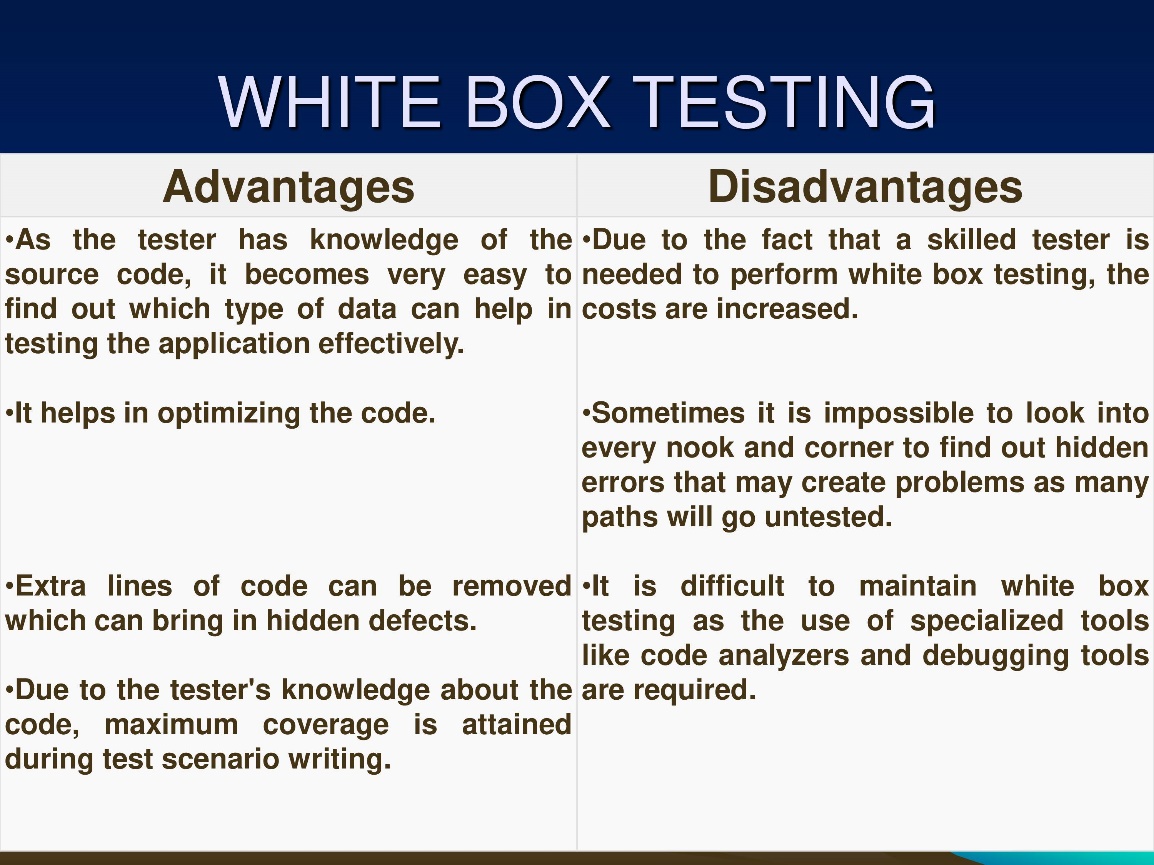






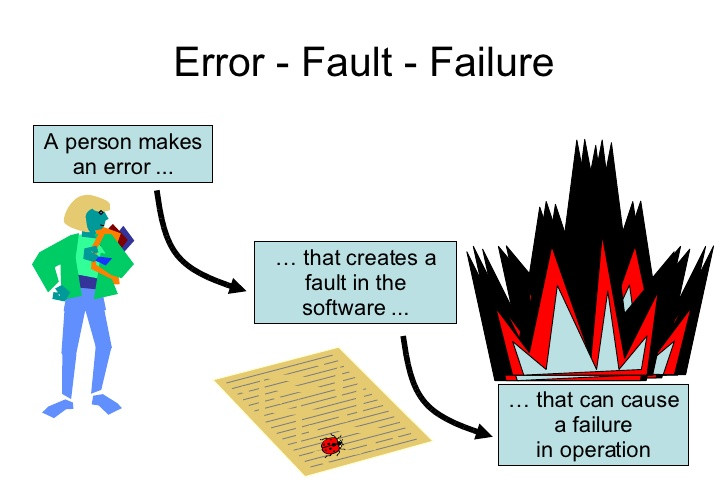


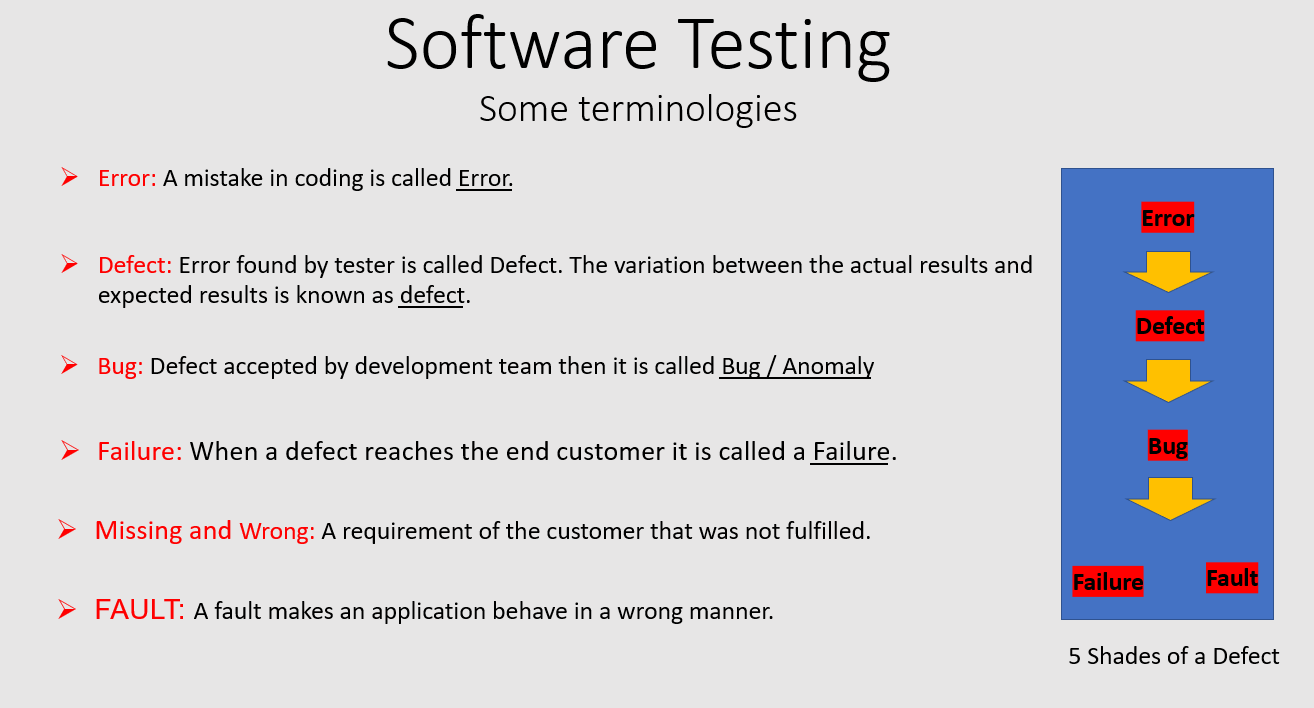


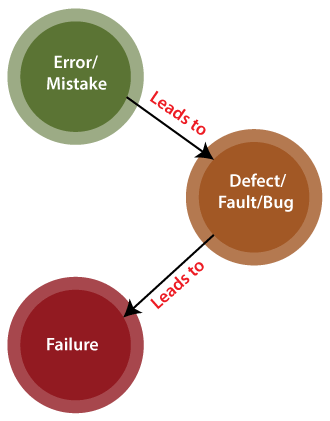


**There are two main categories of testing:**  
  
1) Static Testing  
2) Dynamic Testing

|  |  |
| --- | --- |
| **Static Testing** | **Dynamic Testing** |
| Static testing is completed without executing the program. | Dynamic testing is completed with the execution of program. |
| This testing is executed in verification stage. | This testing is executed in validation stage. |
| Static testing is executed before the compilation. | Dynamic testing is executed after the compilation. |
| This testing prevents the defects. | This testing finds and fixes the defects. |
| The cost is less for finding the defects and fixes. | The cost is high for finding and fixing the defects. |
| It consists of Walkthrough, Inspection, reviews etc. | It consists of specification based, structure based, Experience based, unit testing, integration testing etc. |







Mistake in codeing (Programmer)

Error found by Tester

Error found by end user



There are main 4 Lelves of Testing

But these are catagirised by 2 level

High and low level

High Level Testing (It is Main)

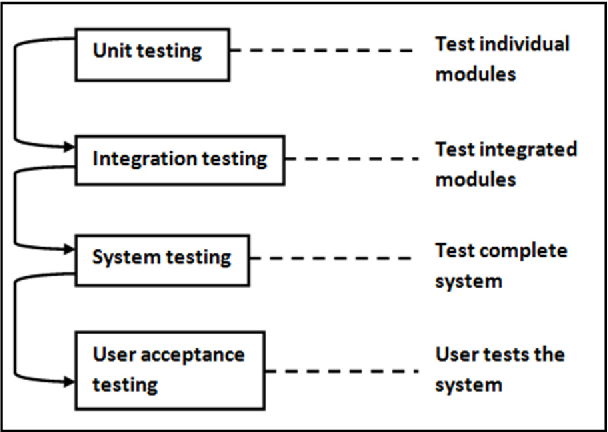
System Testing || UAT

(Tester)

Unit and Integration Testing

(Programmer)

Low Level Testing



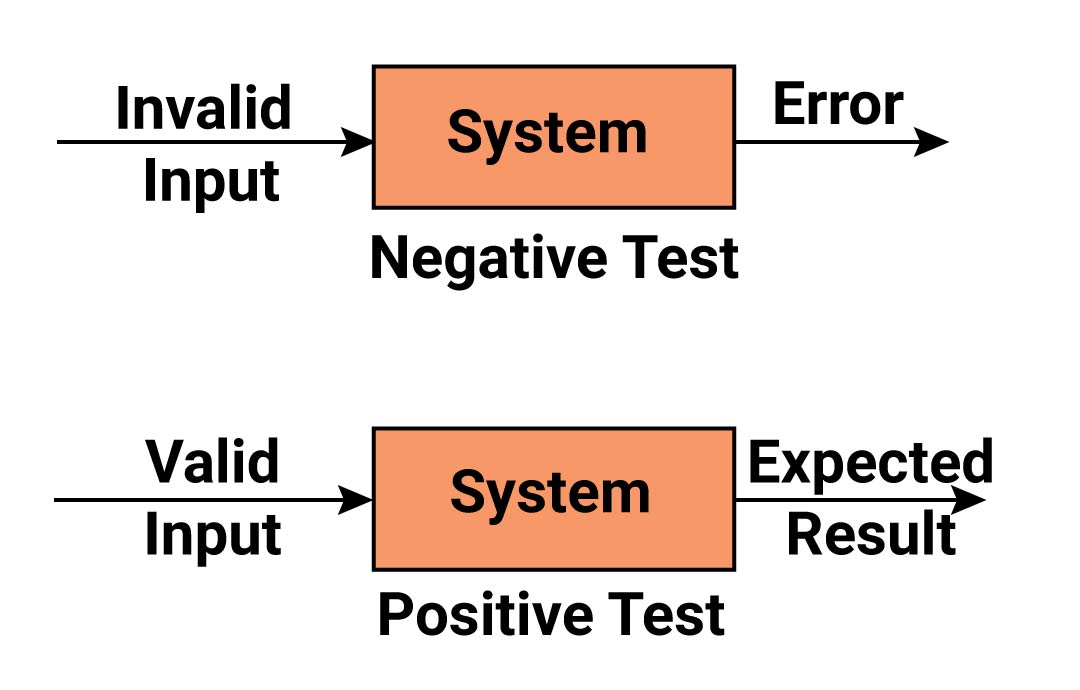
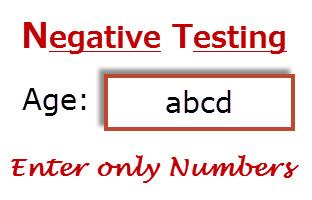
Test Planning : a document that describe how to perform testing on entire application

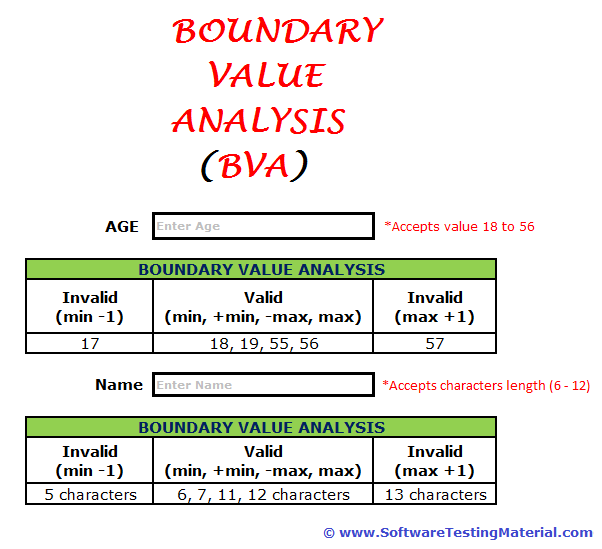
Quatliyt lead or QA Manager >> Prepare Test Plan

What to test ?

When to test ?

How to test ?

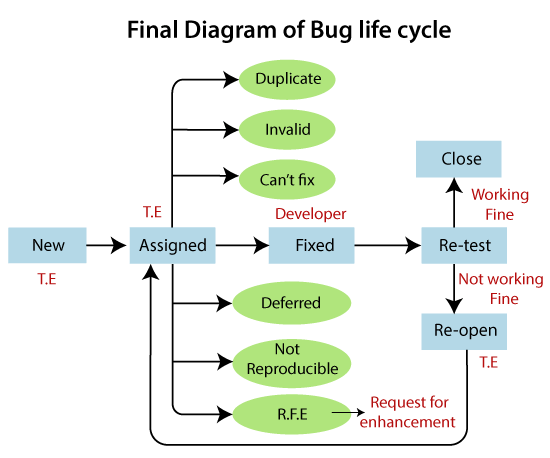


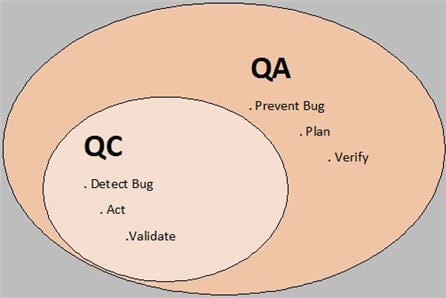


The main use of BVA is to reduce the test cases

It is Black Box Testing because checks functionality

i.e. AUT = Application Under Test





*Verification* is the process confirming that something—software—**meets its specification**

*Validation* is the process confirming that it meets the **user’s requirements**

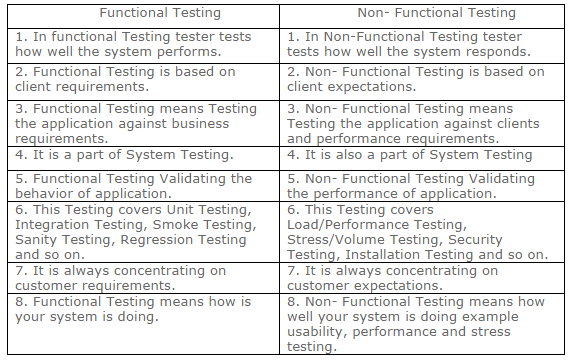
Verification = meets specication as document

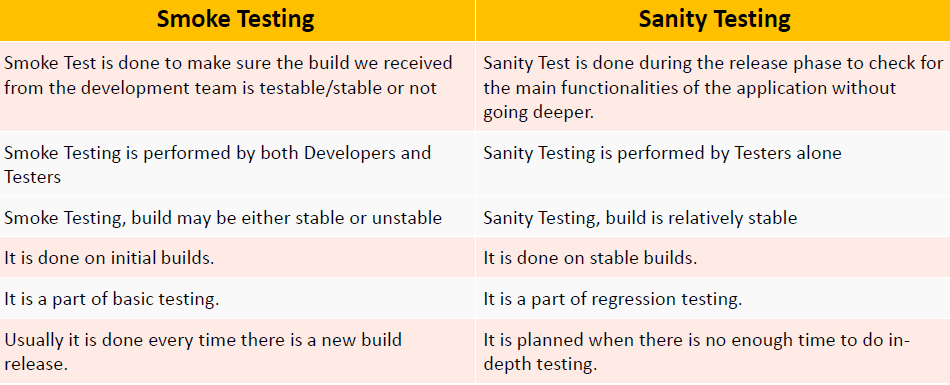
Validation = meets User requirement ( because software is ready hence main tested for User)

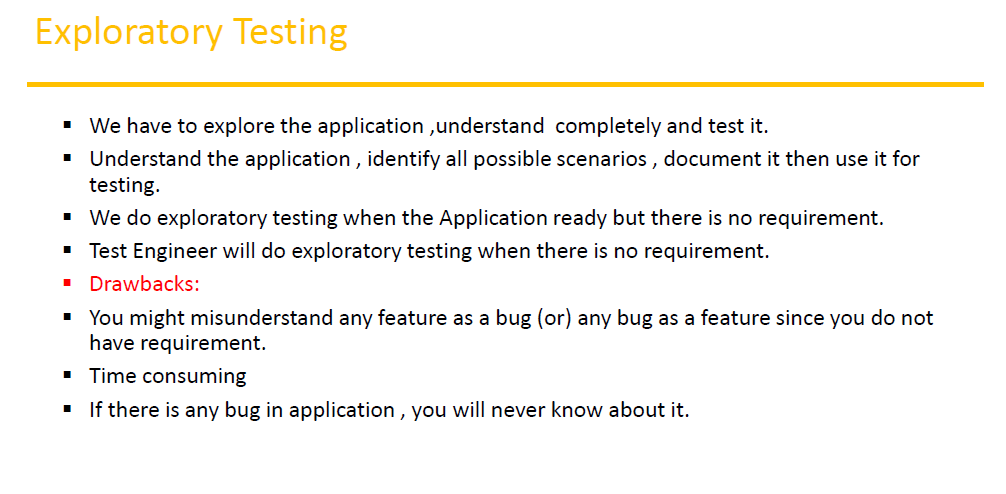
Verification : Its static process of analysing the document , not actural product

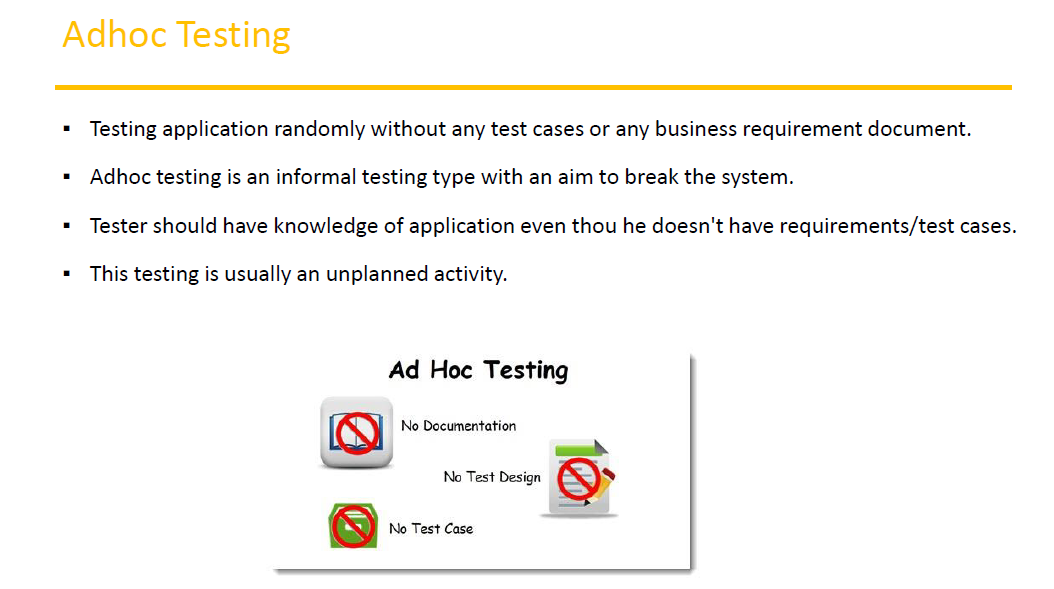
Validation : it involes Dynamic Testing (unit, integration ,system testing )

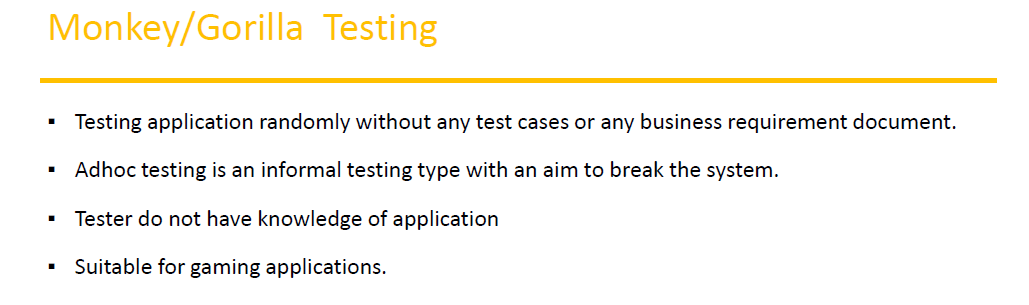


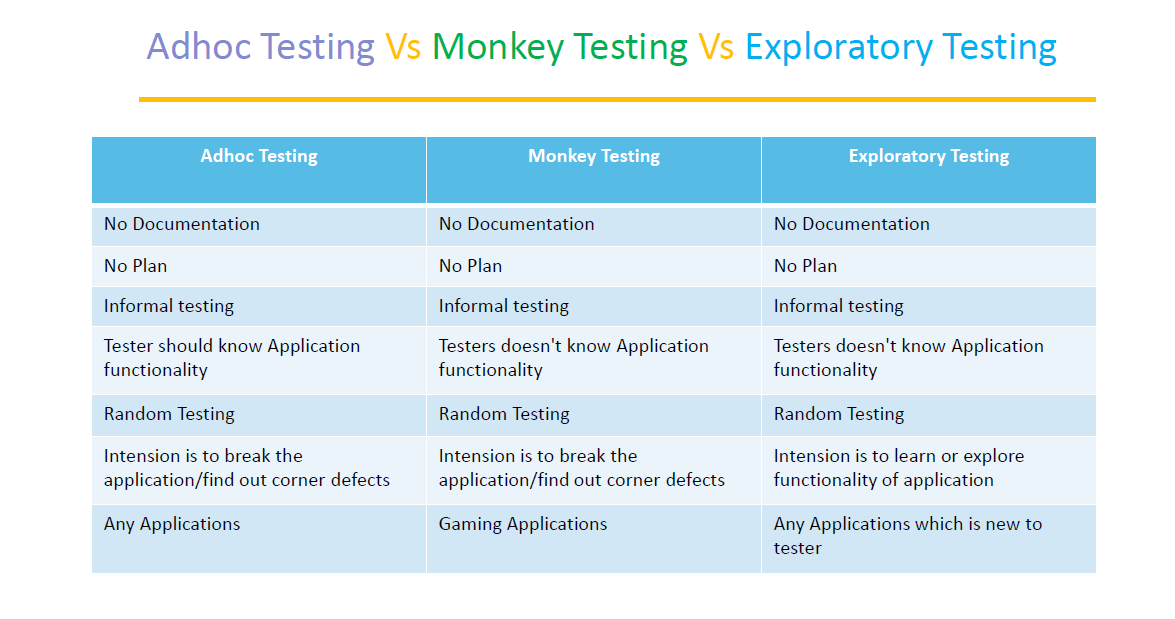


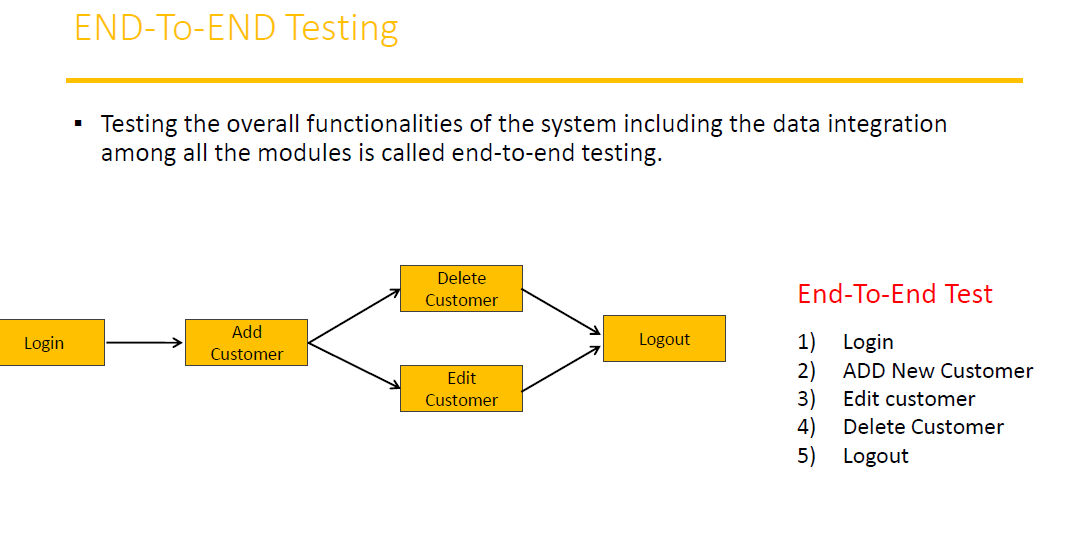




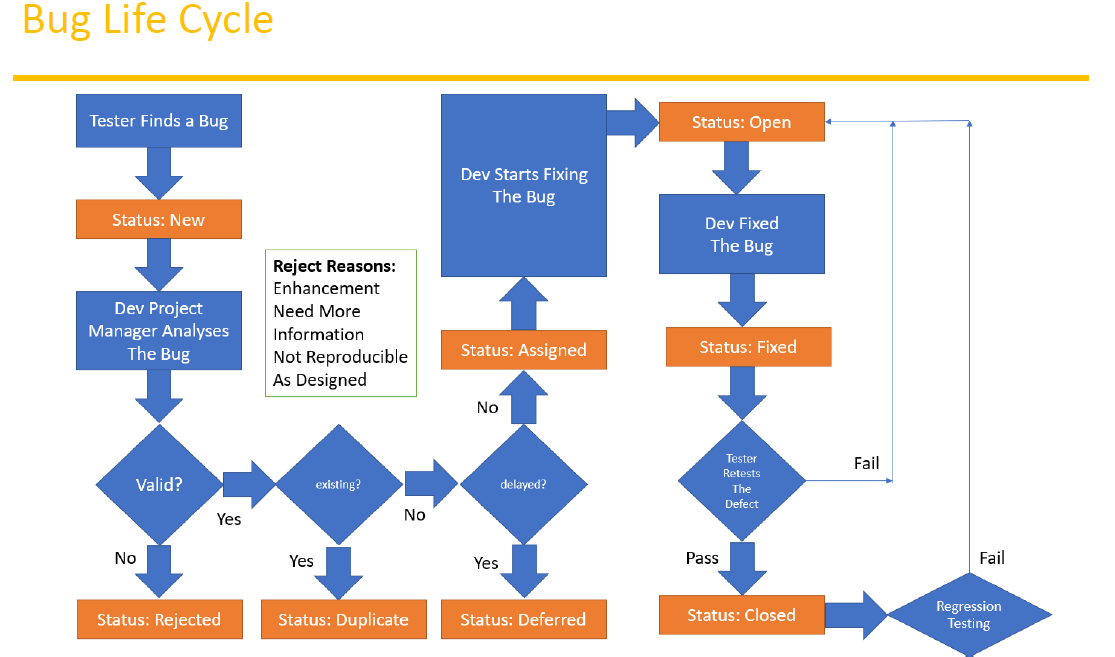




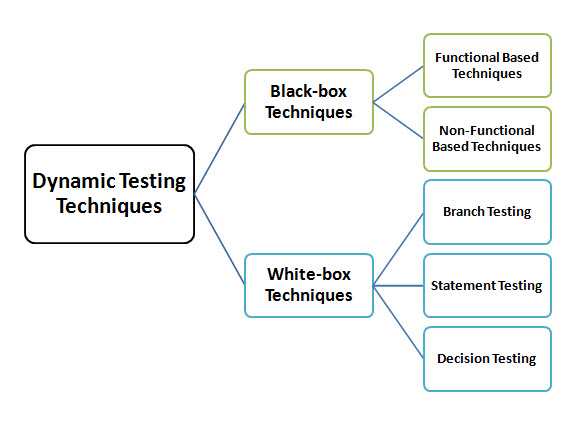


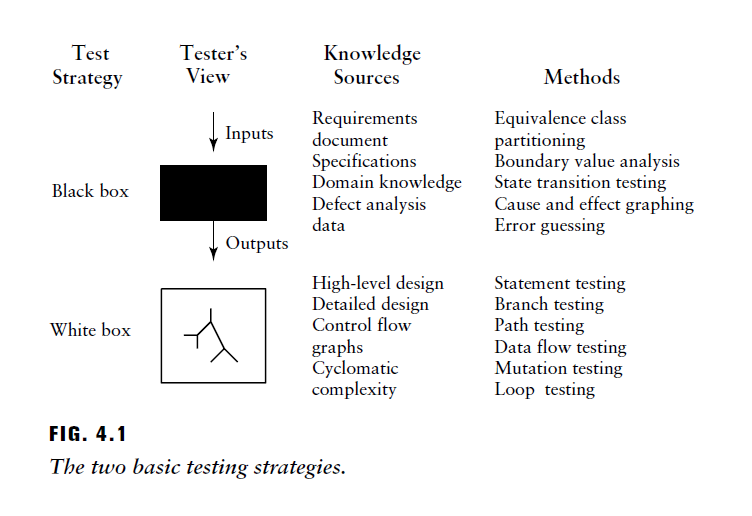






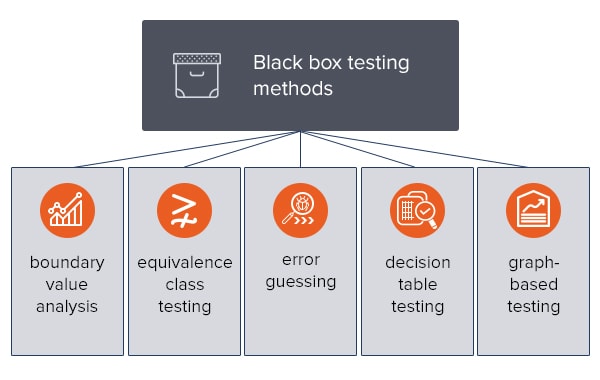
| **Black Box Testing** | **White Box Testing** |
| --- | --- |
| It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it. | It is a way of testing the software in which the tester has knowledge about the internal structure or the code or the program of the software. |
| It is mostly done by software testers. | It is mostly done by software developers. |
| No knowledge of implementation is needed. | Knowledge of implementation is required. |
| It can be referred as outer or external software testing. | It is the inner or the internal software testing. |
| It is functional test of the software.(Black box) | It is structural test of the software. |
| This testing can be initiated on the basis of requirement specifications document. | This type of testing of software is started after detail design document. |
| No knowledge of programming is required. | It is mandatory to have knowledge of programming. |
| It is the behavior testing of the software. | It is the logic testing of the software. |
| It is applicable to the higher levels of testing of software. | It is generally applicable to the lower levels of software testing. |
| It is also called closed testing. | It is also called as clear box testing. |
| It is least time consuming. | It is most time consuming. |
| It is not suitable or preferred for algorithm testing. | It is suitable for algorithm testing. |
| Can be done by trial and error ways and methods. | Data domains along with inner or internal boundaries can be better tested. |
| **Example:** search something on google by using keywords | **Example:** by input to check and verify loops |
| **Types of Black Box Testing:**    * A. Functional Testing * B. Non-functional testing * C. Regression Testing | **Types of White Box Testing:**    * A. Path Testing * B. Loop Testing * C. Condition testing |





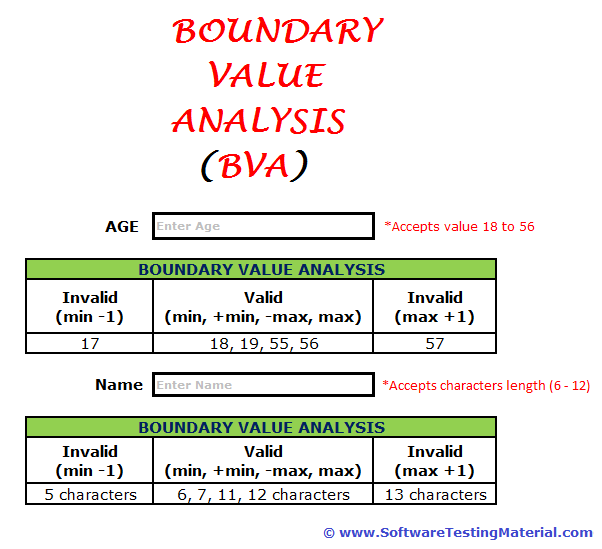
Black box testing == data driven testing = input/output testing

White box testing == logic drivern testing = Logic-Coverage Testing

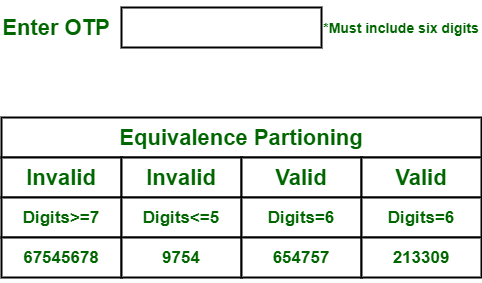


BVA = Boundary value analysis based on verification of only extreme boundary values, e.g., maximum,

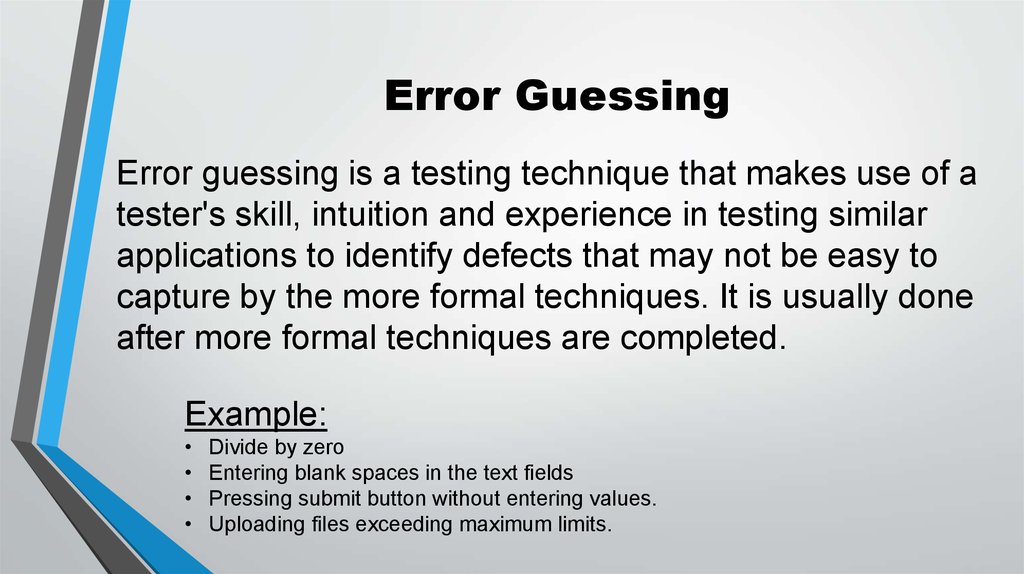
minimum, and typical (Eg lower boundery 18 higher 56)



ECP = Equivalence class testing based on checking one value from each partition

  
 1 2 3 4 4 partition

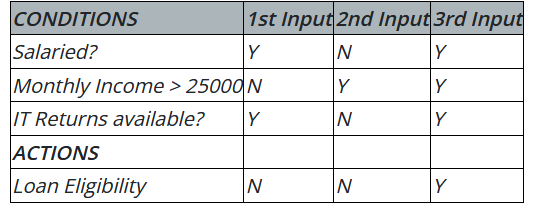
ERROR Guessing = based on the previous experience of a QA engineer also called experience based testing



**Decision table** testing based on a tabular representation of combinations of inputs and

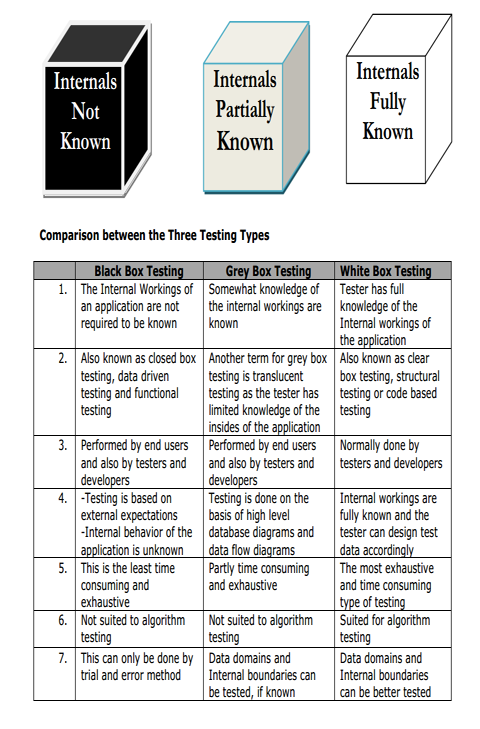
correspondent system behavior

depends of conition



Depends of condition or decision loan is given

**Graph-based** testing where a test case is written for each graph that represents the object



**White box testing**

is an approach that allows testers to inspect and verify the inner workings of a software system—its

code, infrastructure, and integrations with external systems

## White box testing techniques

There are many ways you can analyze software with white box testing. Most testers will use a process called code coverage analysis to eliminate gaps in the testing of the code. There are a variety of techniques you can use to accomplish this, including:

**Statement coverage:**

This technique ensures that each line in the code is tested at least once to find faulty code more easily.

**Branch coverage:**

Using this technique, each possible path or decision point of a software application is checked for accuracy.

**Condition coverage:**

All individual conditions are checked.

**Multiple condition coverage:**

All imaginable combinations of all the conceivable condition outcomes are tested at least once.

**Basis path testing:**

Control graphs are created from either flowcharts or code. Cyclomatic complexity is then calculated to define the number of independent paths so that the minimum number of test cases can be designed for each path.

**Flow chart notation**:

This technique uses a directed graph made up of nodes and edges, where each node represents a decision point or sequence of statements.

**Cyclomatic complexity:**

This is the measure of a software's logical and cyclomatic complexity. It is used to define how many independent paths are present.

**Loop testing:**

Loops are commonly used in white box testing and are fundamental to many algorithms. Problems are often found at the beginning or the end of a loop. Loop testing can be divided into simple loops, nested loops and concatenated loops.