MANUAL TESTING

15-04-2022

Prepared By: Karimunnisa

<u>Software</u>

A set of executable programs or collection of programs in a computer is called Software.

<u>Types of Software</u>:

1)<u>System Software</u>:

Software which is used to operate the system is called System software.

Ex: Device drivers, Operating systems, Utilities , Compilers, Interpreters, Debuggers etc ..

2)<u>Application Software</u>:

Software which is applicable for particular purpose is called Application software.

Application software is of 2 types:

(i)<u>Standalone/Desktop-oriented Application Software</u>:

Applications which are installed on the system are called Standalone/Desktop-oriented Application Software.

eg: Ms-Office, calculator, notepad, IDE's , /mp3 players etc.

(ii)<u>Distributed Application Software</u>:

Applications which runs in the context of server and internet is called Distributed Application Software.

Eg: all websites, all social media applications etc..

Software Testing

Testing:-

Comparison between expected values and actual values.

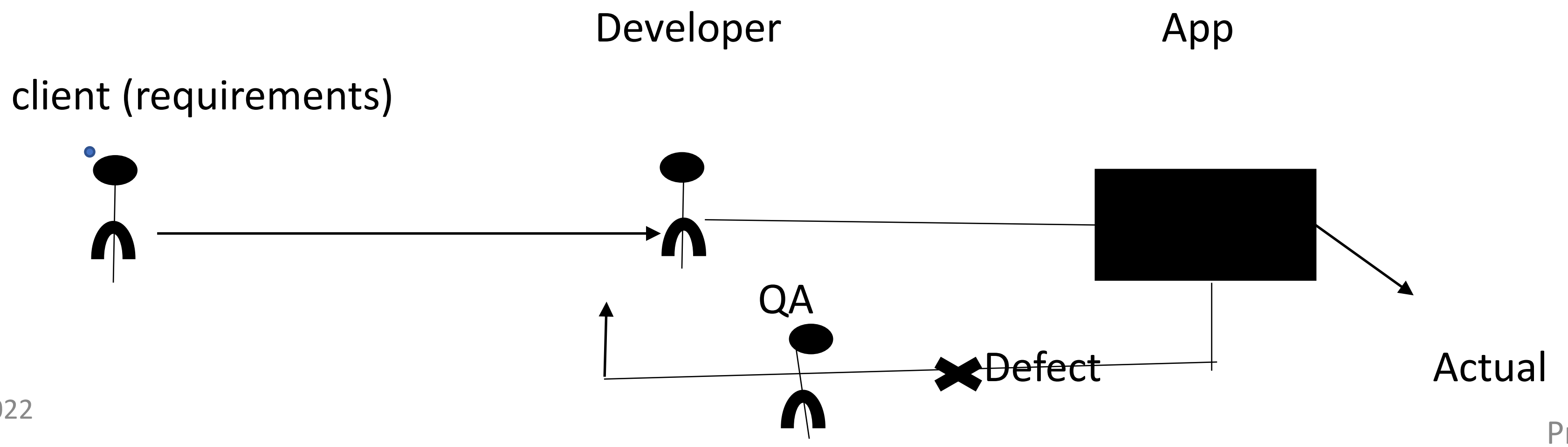
- Before developing the application, the client requirements are called Expected values.
- After development, the final output of the application is called Actual value.

Software Testing

Definition:

It's a process of executing an application with an intention of finding defects.

- The objective of testing is to release a quality product to the client.



Software Quality:-

Customer/Client/End user satisfaction is called Quality.

- Bug-free
- Delivered on time
- Within budget
- Meets requirements and/or expectations
- Maintainable.

Difference between Error, Defect, Bug and Failure:-

Error:-	While developing the application if programmer find out any mistake in coding part is called Error.
Defect:-	During testing, if any expected value not equal with actual value is called Defect.
Bug:-	Defect accepted by development team then it is called a Bug / Anomaly.
Failure:-	After development and testing, during utilization if customer facing any problem is called Failure.

Difference between Project and Product:-

Project:- An application developed for specific client requirement is called Project.

Eg:- TCS, TechMahendra companies are examples for project/service based companies.

Product:- An application developed for multiple client requirements or entire market requirement is called Product.

Eg:- Gmail, Yahoo mail, Rediff mail etc..

Google, Microsoft companies are examples for product based companies.

Why the software has bugs?

- Miscommunication or no communication
- Software complexity
- Programming errors
- Changing requirements
- Lack of skilled testers

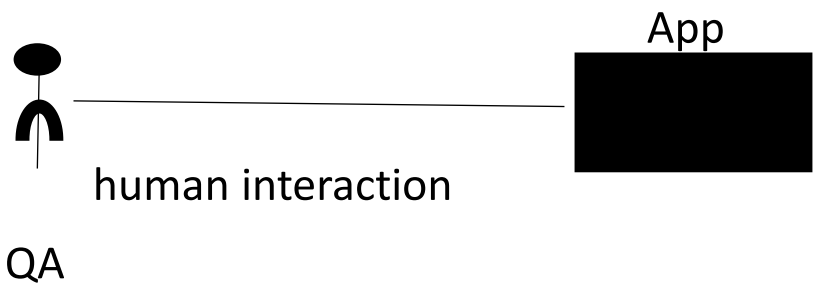
Software Testing 2 ways:

1)Manual Testing

2)Automation Testing

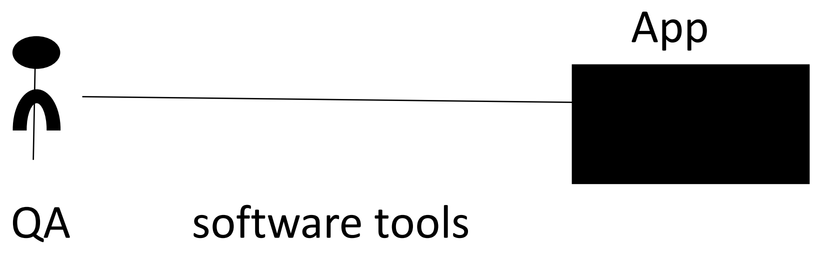
Manual Testing:-

It's a process to conduct testing on application by human interactions is called Manual Testing.



Automation Testing:-

It's a process to conduct testing on application by the help of software tools is called Automation Testing.



SDLC

SDLC stands for Software Development Life Cycle. It will explain all the implementation activities of project development.

Software Bidding

↓

Kick-off/Start-up meeting with team members with PM

↓

Project Initiation Note

↓

Requirement Gathering - BRS – BA and SRS/FRS/UC - SA

↓

Project Plan

↓

Design HLD LLD - system architects/technical team

↓

Coding

↓

Testing

↓

Release

↓

Support / Maintenance

1. <u>Software Bidding</u>:-

At this phase, C.E.O (Chief Executive Office) go to client side to confirm the project and make agreement between software company and corresponding client.

2. <u>Kick-off Meeting</u>:-

Once project is confirmed by C.E.O, he will conduct meeting with Project Manager's to select Project Manager for current project.

Note:-

Any kind of start-up meeting can be called as Kick-off meeting.

3. <u>Project Initiation Note</u>:-

Once Project Manager confirmed by C.E.O, Project Manager will prepare initiation/information document called P.I.N. It contains information of client, project, domain etc.,.

4. <u>Requirement Gathering</u>:-

Task:- Collecting requirements of clients.

Role:- Business Analyst, System Analyst (Domain Experts)

Documents: Business Requirement Specification document (BRS)

System Requirement Specification document (SRS)/ FRS

Process:- By the help of Project Initiation Note, Business Analyst will collect requirements from client prepare as a BRS document. Based on BRS, with the help of System Analyst (SA), prepares number of SRS documents for every team. At this phase, to understand the project requirement, every software engineer can

5. <u>Project Plan</u>:-

Task: Prepare plan for project implementation.

Role: Project Manager with the help of development manager.

Documents: Project plan contains development plan, Testing plan, Design plan documents.

Process: As per client requirement, in order to complete the project, Project plan will be prepared by Project Manager, with the help of Development Manager, Test Manager. This plan document contains scope, resources, roles and responsibilities, what to develop and how to develop etc.

6. <u>Design</u>:-

Task: Requirements designing for application development.

Role: Architect/ Technical team

Document: HLD [High Level Design]

LLD [Low Level Design]

Process: Based on requirements and plan, before start coding by programmer, Architect / Technical team design the requirements as a HLD / LLD documents.

HLD defines the requirements as a module wise.

LLD defines the requirements as a functionalities.

7. <u>Coding</u>:-

By following requirements, plan, design, programmer start writing code for responsible modules development using different technologies such as Java, C#, Python, Ruby etc.

8. Testing:-

Whenever all implementation activities completed, to confirm the correctness of code which is carried out by programmer called “White Box Testing”. This testing collectively of 2 levels – i) Unit Testing and

ii) Integration Testing.

Whenever 100% coding and White Box Testing completed, to confirm the correctness of customers requirements, testing an front end application called “Black Box Testing”. It is a collectively of 2 levels – i) System Testing and ii) User Acceptance Testing.

9. Release Process:-

Once all activities are successfully completed, if client also accepted in user acceptance testing, then the delivery team will release the project to the customer.

10. Support/Maintenance:-

After delivery to the customer, during utilization, if customer is facing any problem called “Failure”. If any failure occurred, to handle those requirements, Maintenance team will provide support to the project.

Difference between Verification and Validation

Verification :- It’s a process of verifying developing the project is right or not called “Verification”, also called as “Static Testing”. Or whether app is properly implemented or not is verification.

Verification is to check, whether software confirms to the specifications and it is done by Development team, it is a in-house activity of the development, it is an Quality Assurance (QA) activity.

Validation :- It’s a process of validating the developed project is right or not, called “Validation”, also called as “Dynamic Testing”. Or whether app is properly working or not is validation.

Validation is to check, whether the software meets the customer expectations. It is a Quality Control [QC] activity, which detects the defects during testing of the product.

Software Development Models

There are various Software Development Approaches defined and designed which are used during development process of software. Those approaches referred as "Software Development Models".

Software Development Models classified into –

1) Sequential Models

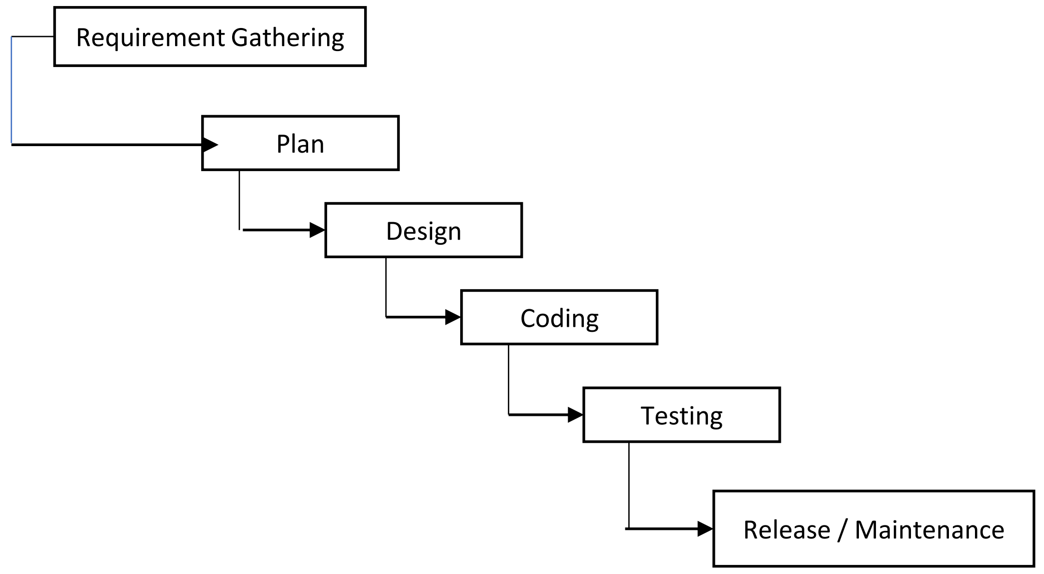
2) Incremental Models - agile process

1) Sequential Models :-

These models are best suitable for small size of projects, where all SDLC activities will be carried out one after another, for all entire project.

Waterfall model and V model are best examples for sequential model.

I) Waterfall Model :-



- It is a sequential model, suitable for small size of projects.
- Waterfall model same as SDLC process.
- Whenever client requirements are clear in Waterfall model all implementation activities will be carried out step by step process.
- In Waterfall model, only Validation required, verification not required.
- In this model, flow of activity looks like a waterfall, so this model is titled as Waterfall model.

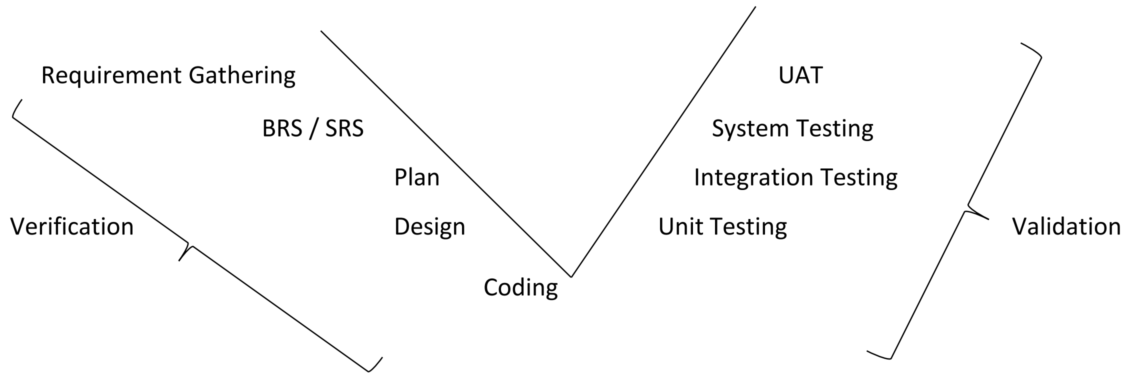
Advantages:-

- Easy to understand and simple.
- Works well for small projects.
- Easy to manage step by step activities.

Disadvantages:-

- Not suitable for big size projects.
- In middle of the project, if any change requirements needed then going back and changing requirements are tedious job.
- Testing is conducted only after the development, bcz of this, it is effecting on the release.

II) **V Model** :-



‘V’ stands for Verification and Validation. This model is suitable for small size of application, where the requirements are clear and chances of making mistakes are more, while implementing the application to reduce this at every step of implementing software testing apply both verification and validation.

In ‘V’ model, once requirement documents are ready, based on those requirements, along with developers, testers will start testing activities such as Test plan, Test Scenario, Test cases documentations. So testing activities will start before coding.

Advantages:

- Simple and easy to use.
- Save time.
- Testing activities will be start before coding.
- Ensure 100% quality.
- Implementing both verification and validation .

Disadvantages:

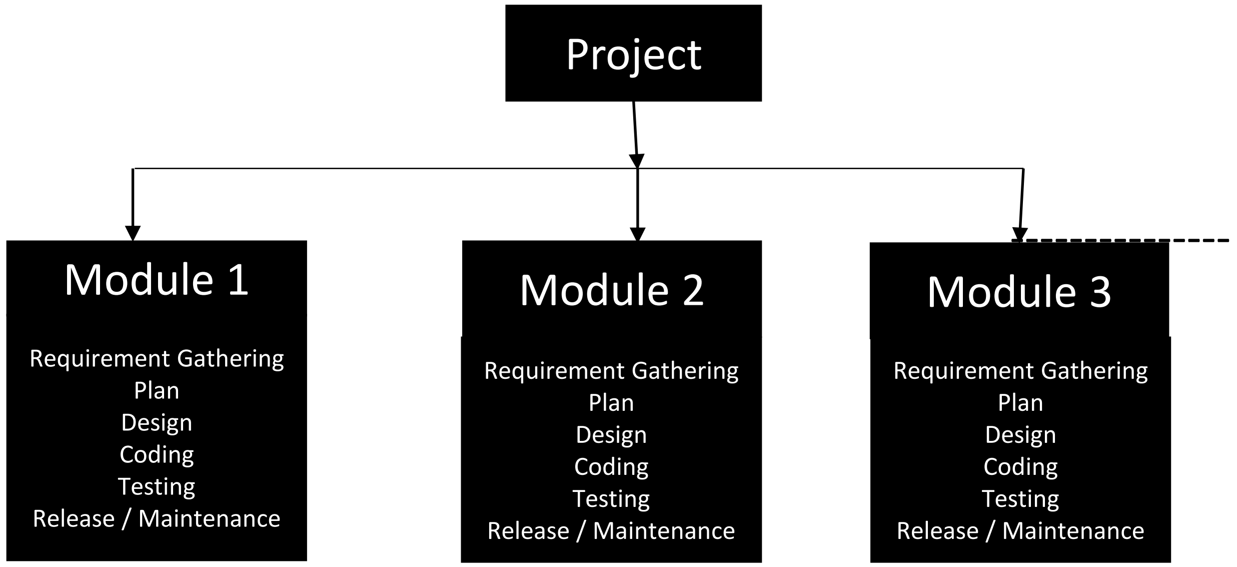
- Not suitable for big size of the projects.
- If any changes happen in middle of the project, along with requirement, development and testing documents also has to be updated.

2) **Incremental Models**:

These models are best suitable for big size of project. In incremental model, a big project will be dividing into modules, then all SDLC activities will be carried out module by module.

RAD model, Prototype model, Spiral model and Agile model are the best examples for Incremental models.

I. **RAD Model** :-



RAD stands for Rapid Application Development. Due to lack of time or within short term to complete big size of projects in RAD model, a big project will be dividing into modules and every module will be considered as a mini project, a separate team will be scheduled to implement all SDLC activities, for those modules parallelly. Once all modules are implemented, these modules will be combined and delivered to customer.

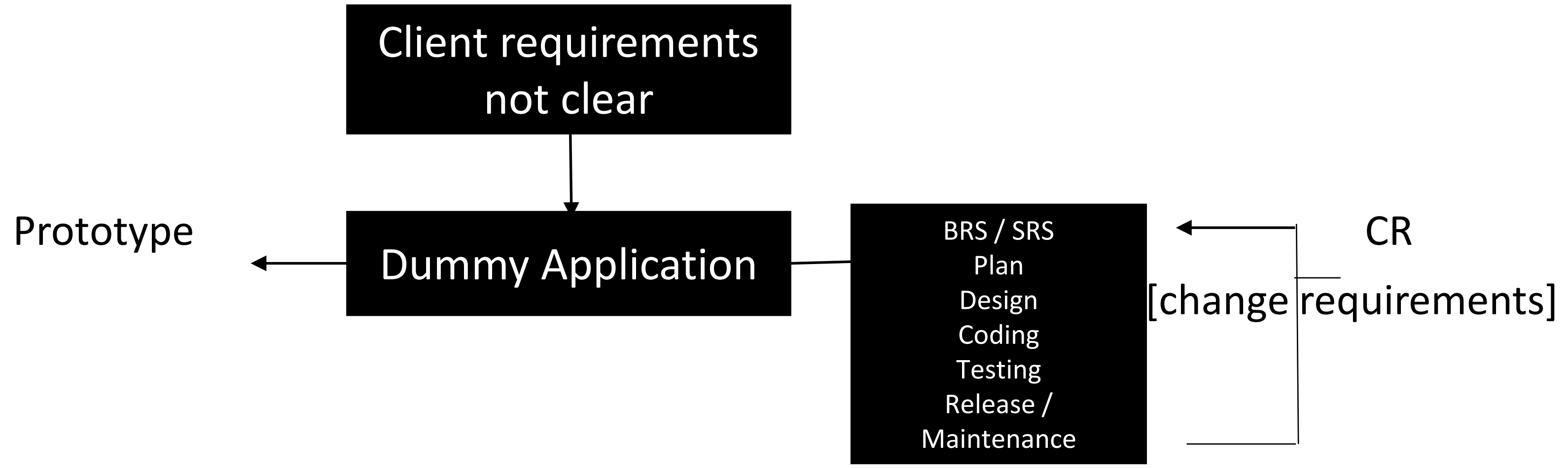
Advantages:

- Reduce development time.
- Suitable for big size of projects.

Disadvantages:

- Not suitable for small size of projects.
- Expensive model.
- Integration problem will be occurred.

II. **Prototype Model**:-



Due to lack of requirements or if customer confusing in the requirements, in this model instead of developing actual application, develop Dummy application called “Prototype “. It will be accepted by client.

Once the client approved the Prototype, based on the sample BRS/SRS will be designed, and based on those requirements all implementation activities carried out such as Plan, Design, Coding, Testing activities. If any changes requested by the customer after deliver, the same will be implemented as Change Requirement (CR). Those cycles will be continued with same process.

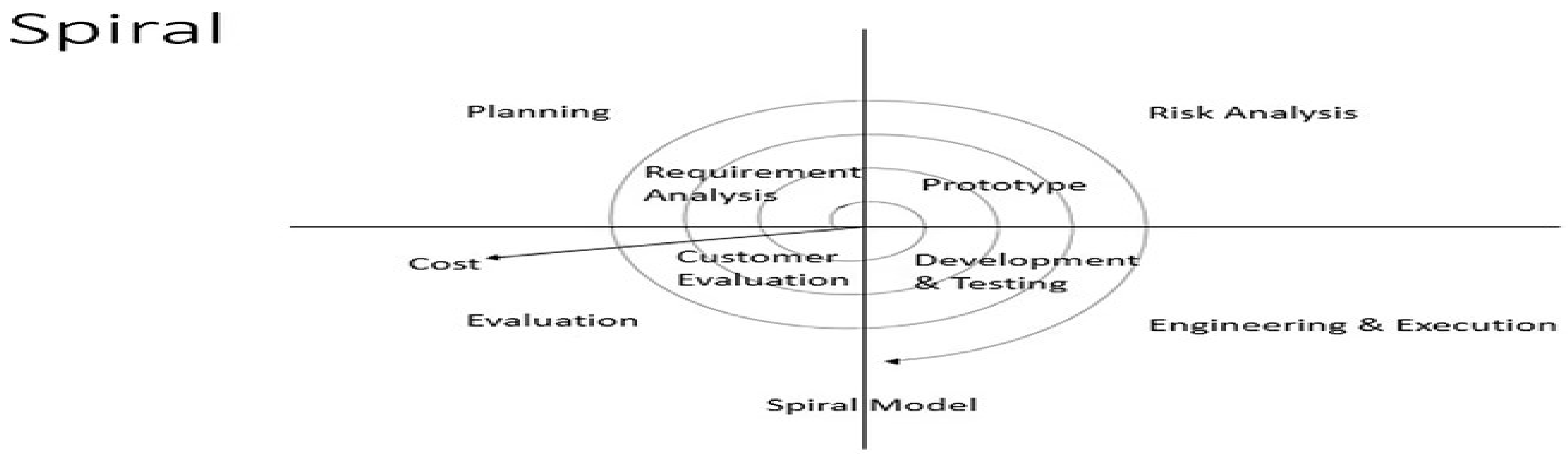
Advantages:

- Change Requirements will be implemented after deliver to the client.
- If client requirements are not clear also we can implement project based on prototype.
- Missing functionality can be identified easily.

Disadvantages:

- Not suitable for small size of projects.
- Expensive model.
- More time and resources required.

III. **Spiral Model**:



This model is suitable for Maintenance or Long term projects. Whenever customer requirements are frequently changing or dynamic requirements of the customer, in this model application will be implemented requirement by requirement.

In this model, the flow of activity looks like a spiral net, so this model titled as a “Spiral Model”.

Advantages:

- Additional functionality can be added in next cycles.
- Project will be deliver to customer early in the software cycle.
- Best suitable for Maintenance and Long term projects.

Disadvantages:

- Expensive model and
- Documentation will be more
- Require more time.
- Not suitable for short term or small size of projects.

IV. **Agile Model/ Agile Methodology:-**

Agile model is a Iterative (same process will be repeating again and again multiple times, means getting requirements, analysis, design, coding, testing will repeats) and Incremental (adding new features/modules on existing software) process or approach.

It is a day-by-day Waterfall model.

Agile Principles:

- Customer no need to wait for long time.
- We develop, test and release piece of software to customer with few number of features.
- We can accommodate / accept requirement changes from customer easily.
- In this model, work along with client representative team **from** day 1 onwards.
- Release is very faster within 2/3 weeks can release product/project.

It has more frameworks, such as XP, Scrum, Kanban, Iterative where we will discuss on Scrum process.

Scrum - Jira tool

Scrum is a framework through which software product is build by following Agile principles.

Agile is a defined process with some principles.

Scrum includes a group of people called Scrum Team (normally 5 to 9 members). They are Product Owner, Scrum Master, Development (Dev) team, Testing (QA) team and everybody work together to deliver quality product within the short time.

Roles and Responsibilities of Scrum Team:

1. Product Owner:

- Product Owner is the main person who writes or defines functionality of product by directly contacting with client and get the requirements.
- He will prioritize those features and assigns to developers and testers.
- Accept or reject work result.

2. Scrum Master:

. He will give awareness to people agile process

- Scrum Master takes care of entire Agile process (from beginning of software to till delivery).
- Project Manager or Client will work as a Scrum Master.
- He is responsible for sprint plan, sprint meeting and scrum meeting.
- He will handle any issues in team on Agile process.

3. Dev Team:

Developers will develop the software.

4. QA Team:

Testers will test the software.

Scrum Terminology/Agile/Scrum Ceremonies

1) User Story:

It is a feature / module in a software. It gives general explanation of software feature.

Note:

Product Owner will define these User Stories and Epic.

3) Product Backlog:

It is a list of all user stories prepared by Product Owner at the beginning of Agile process.

4) Sprint:

Duration of time to complete User Stories, decided by Product Owner and Team, usually 2-4 weeks of time.

5) Sprint Planning Meeting:

Meeting conducting by team (Product Owner, Scrum Master, Dev team, Testing team), to define what can be delivered in the sprint / particular duration of time and to review previous user stories completed as per sprint plan or not and if any user stories are unable to complete within sprint plan then it will consider as a back logs and to explain new sprint user stories.

6) Sprint Backlog:

List of committed stories by Dev/QA for specific sprint.

7) Scrum Meeting: scrum call/standup meeting

Meeting conducted by Scrum Master every day 15 mins, called as Standup meeting / scrum call, to discuss the status of the project.

Everyday it focus on 3 things in this meeting:

a. What did you do yesterday?
b. What will you do today?
c. Are there any blockers in your way?

8) Sprint Retrospective Meeting:

It is conducted only once at the end of every sprint.

It focus on 3 things :

a)What went well.

b) What went wrong and

c) What steps need to be taken.

9) Story Point:

Rough estimation of user stories given by Dev and QA. During sprint planning meeting itself, every story will be estimated by Product Owner, Dev and QA.

1 story point = 1 hr/ 1 day (6-8 hrs) depends on company

Suppose, for

Login -> Dev given 5 (means 5 hrs)

QA given 3 (means 3 hrs),

so totally 5+3 =8 hrs(means 1 day) 2 days

Based on user story complexity.

10) Burndown/Up Chart:

It shows how much work remaining in sprint. This will be designed by ScrumMaster daily. It is a graph.

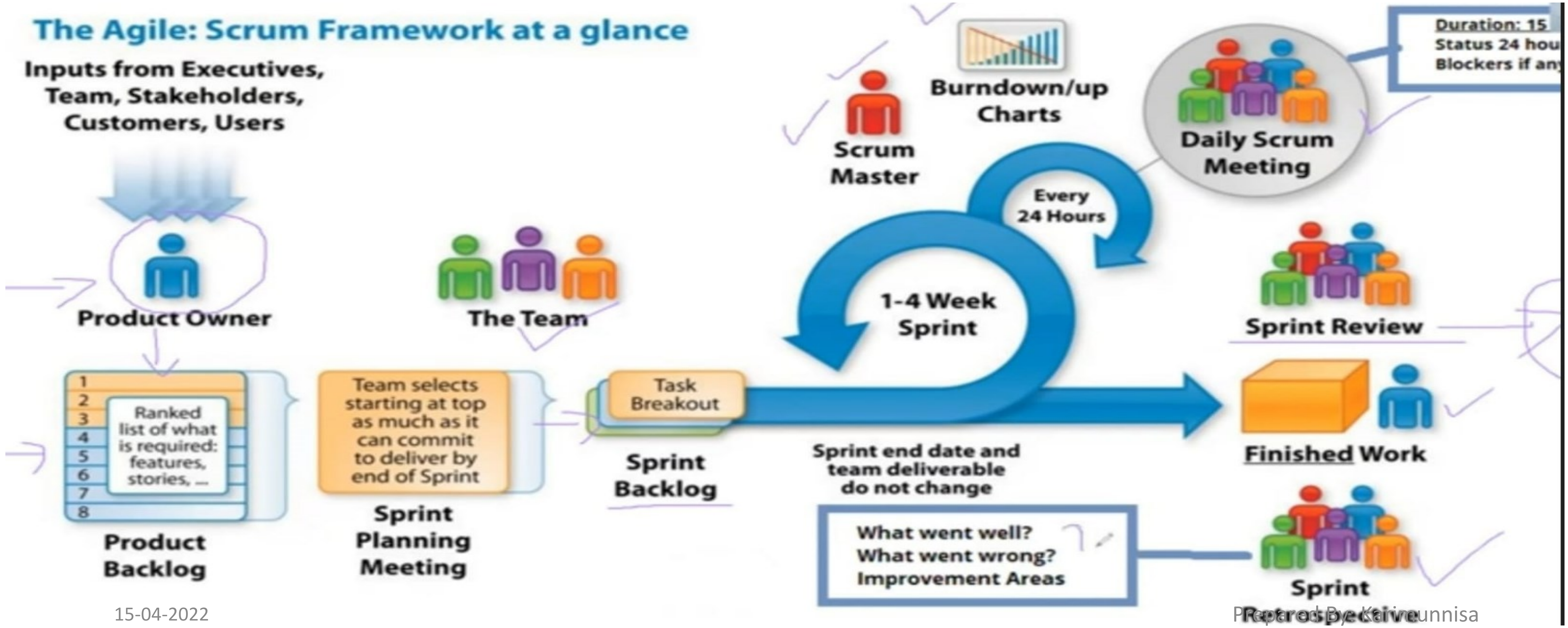
Advantages:

- Requirement changes are allowed in any stage of development.
- Release is very fast.
- Customer no need to wait for long time.

- Good communication between team.
- It is easy model to adopt. Flexible model.

Disadvantages:

- There is less documentation, since we release software very faster.



Software Testing Methods and Levels

Software Testing = Verification + Validation

Verification: developers + testers

It’s a process of developing a project is right or not is called Verification, also called as Static Testing.

Static testing means, testing a component or system at specification level or implementation level without executing the code is known as Static testing.

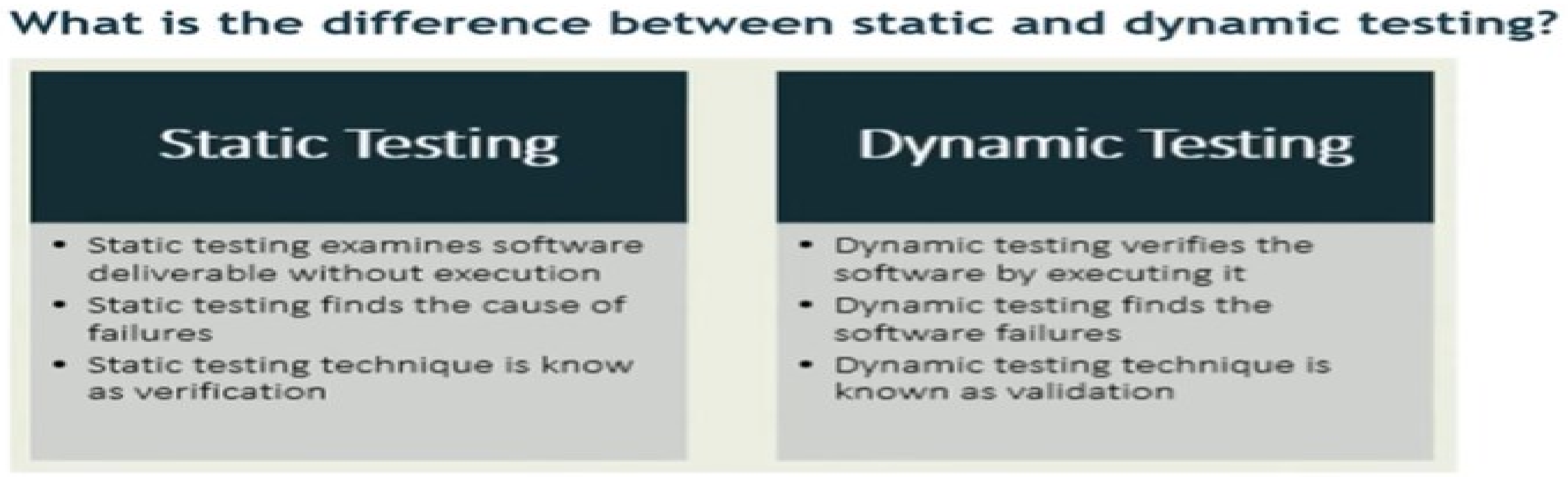
During static testing, software products are examined manually or with some tools but there is no execution done.

Validation: testers

It’s a process of validating the developed application is right or not called Validation, also called as Dynamic Testing.

Dynamic testing verifies the software by executing it.

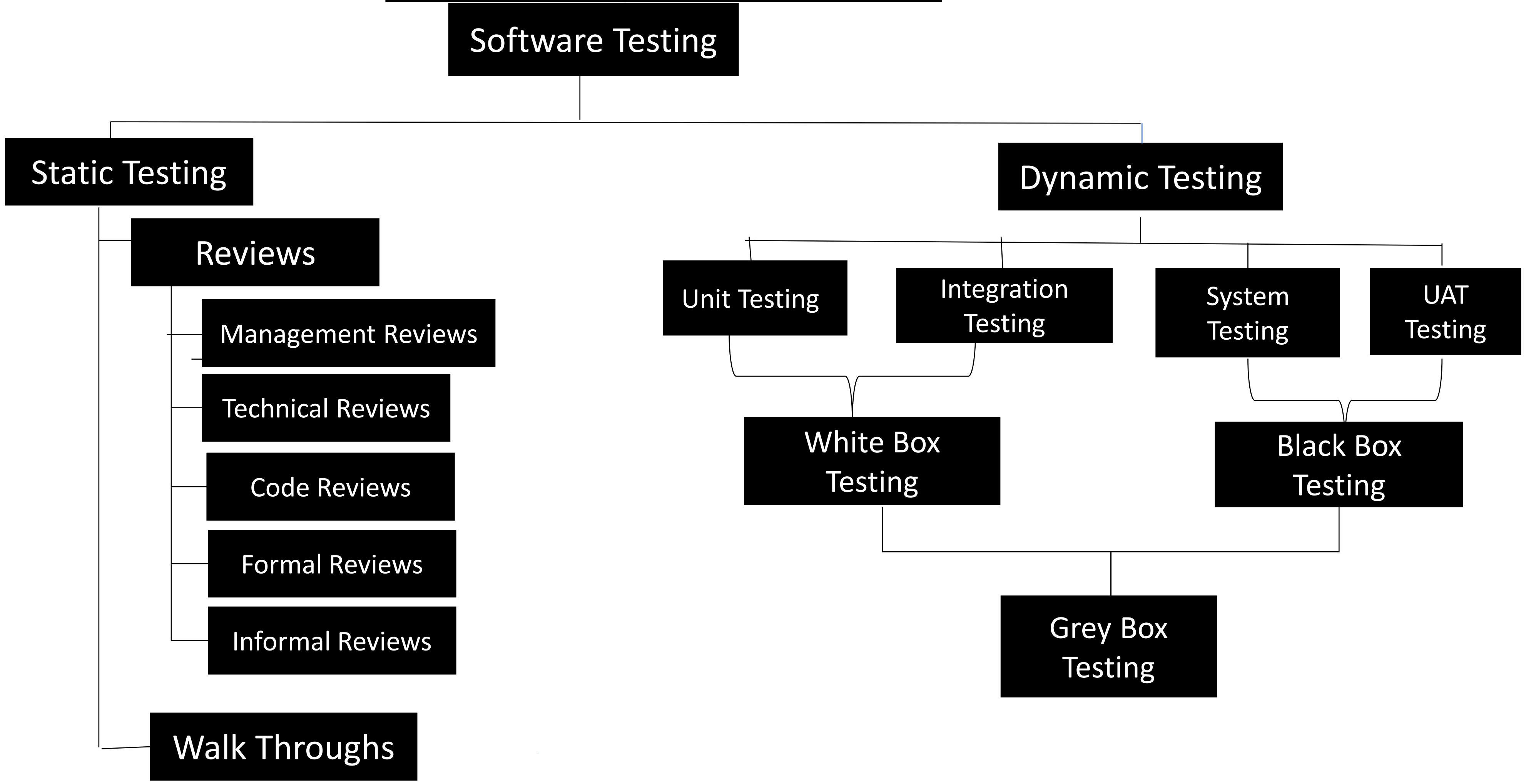
Dynamic testing finds the software failures.



What is the difference between static and dynamic testing?

Static Testing	Dynamic Testing
• Static testing examines software deliverable without execution • Static testing finds the cause of failures • Static testing technique is know as verification	• Dynamic testing verifies the software by executing it • Dynamic testing finds the software failures • Dynamic testing technique is known as validation

Software Testing Methods and Levels



Static Testing

Static Testing is carried out by Reviews and Walk Throughs.

1) **Review**:

Examining project related work called Review.

Example: Examining requirements, design, code etc..

Types of formal Reviews:

i. Management Review :

This review will be conducted by high level management team (Business Analyst, Project Manager) to monitor the project status.

ii. Technical Review :

This review will be conducted by technical team to decide the best approach of design implementation.

iii. Code Review:

This review will be conducted by programmer to decide the best approach of coding part.

iv. Formal Review :

If a review is carried out by following systematic procedures and proper documentations, then those reviews are called Formal review.

Inspections and Audits are best examples for formal review.

If a formal review conducted while executing task, then it is called Inspection.

If a formal review conducted after completion of task, then it called Audit.

v. Informal Review :

If a review is conducted without following any procedures and documentations, then those reviews are called Informal review.

<u>Objectives of Static Testing</u>: (Reviews)

i. To find defects in requirements.

ii. To find defects in design

iii. To identify the deviations in any process.

iv. To provide valuable suggestions to improve the process.

<u>Note</u>: To detect defects from requirements, plan, design, coding,testing we do static testing.

2) <u>Walk Throughs</u> :

A step by step code reviewed by experts called Walk Through.

- Review can be done by anytime and anyone, it is basically a informal review

<u>Dynamic Testing</u>

Dynamic Testing is carried out by 3 methods such as,

(1) WhiteBox Testing - developers

(2) BlackBox Testing - testers

(3) GreyBox Testing - testers

<u>WhiteBox Testing</u>

After coding, before start QA activities, testing conducted on source code by programmer to ensure the 100% code coverage or whether code is working as per client requirement or not is called WhiteBox Testing.

It is a collectively of 2 levels, such as (i) Unit Testing - developer

(ii) Integration Testing - developer

(i) <u>Unit Testing</u> :- (only developers)

A smallest testable part in the source code of the application (or) each and every part of the code working as per requirement or not called Unit Testing, also called as Module Testing (or) Component Testing.

- A unit is a single component or module of a software. So conducting testing on single component or module of a software is called Unit Testing.

While conducting Unit Testing, to ensure code coverages, programmer will follow whitebox testing techniques such as – (a) Condition coverage

(b) Loops coverage

(c) Path coverage

(d) Mutation coverage // they will check both valid inputs and invalid inputs to check the code coverage.

(ii) <u>Integration Testing</u> :- (developers and testers)

Once the unit testing is completed, developers will integrate all source code units and checks interactions in between all modules, which is called Integration Testing.

Types of Integration Testing:-

(I) Incremental Integration Testing.

(II) Non Incremental Integration Testing.

<u>Incremental Integration Testing</u> :

Incrementally adding the modules and testing the data flow between the modules one after the other, is called Incremental Integration Testing.

There are 3 approaches of Incremental Integration Testing.

(1) Top-Down approach

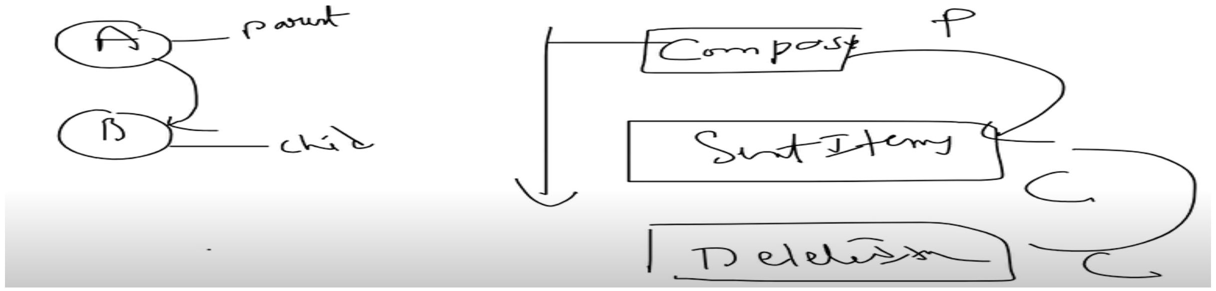
(2) Bottom-Up approach

(3) Hybrid/Sandwitch approach (4) Big bang approach

(1) <u>Top-Down Approach</u>:

Incrementally adding the modules and testing the data flow between the modules. And ensures module added is the child of previous module.

Eg: A module is there, and when we integrate with B module then that B module should be the child module of A.



- This approach is recommended if there are any incomplete programs at top level.

- In this approach, integration testing will be carried out from top to bottom.

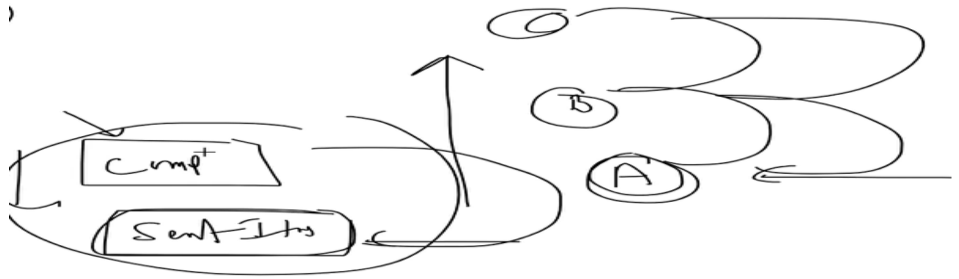
- The incomplete programme at top level will be replaced with **Stubs**.

Note:

Stub is a temporary code used to stop user actions when top module is incomplete.

2)**<u>Bottom up Approach</u>**:

Incrementally adding the modules and testing the data flow between the modules. And ensure the module added is the parent of previous module.



- This approach is recommended when there are incomplete programmes at the bottom level.

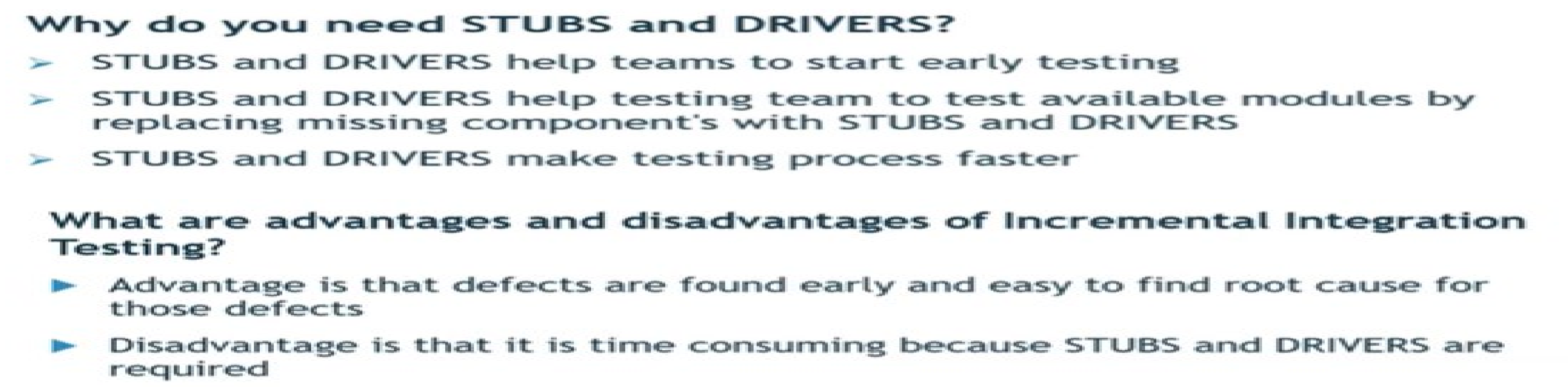
- In this, testing will be carried out from bottom to top.

- The incomplete programme at bottom level will be replaced with **Drivers.**.
Note:
Driver is a temporary code used to continue user actions when bottom module is incomplete.

3) **<u>Sandwich/Hybrid approach</u>**:

Combination of top-down and bottom-up approach is Sandwich/hybrid approach.

- In this, the middle level modules are integrated with both Stubs and Drivers



Why do you need STUBS and DRIVERS?

- STUBS and DRIVERS help teams to start early testing
- STUBS and DRIVERS help testing team to test available modules by replacing missing component's with STUBS and DRIVERS
- STUBS and DRIVERS make testing process faster

What are advantages and disadvantages of Incremental Integration Testing?

- Advantage is that defects are found early and easy to find root cause for those defects
- Disadvantage is that it is time consuming because STUBS and DRIVERS are required

Non incremental Integration Testing :

Here we integrate all the modules at one shot and test the data flow between the modules. (we don’t prefer most of the time this type).

Drawbacks:

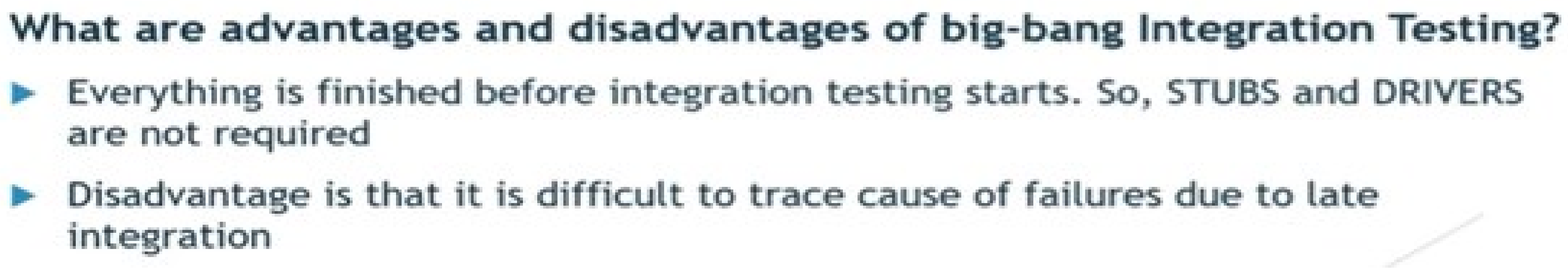
-> We might miss the data flow between some of the modules.

-> If we find any defect we can’t understand the root cause of defect.

Big bang approach :

This approach is recommended whenever 100% coding completed and all source code units are available, then conduct integration testing called as Big bang approach.

NOTE: All components or systems are integrated simultaneously, after which everything is testing as a whole.



What are advantages and disadvantages of big-bang Integration Testing?

- Everything is finished before integration testing starts. So, STUBS and DRIVERS are not required
- Disadvantage is that it is difficult to trace cause of failures due to late integration

Black Box Testing

After 100% coding and WhiteBox testing, testing conducted on application by testing engineer (or) domain experts to ensure the requirement coverage (or) to check whether the application developed as per client requirement (or) not , called BlackBox Testing, also called as Closed Box Testing or Specification Based Testing.

It is collectively of 2 levels :

(i) System Testing - functional and non-functional

(ii) User Acceptance Testing – UAT

System Testing

System Testing is a process of validating both Functional requirements and Non Functional requirements.

- Validating client / business requirements are called Functional Testing. Behaviour of the application is called functional testing.
- Validating client / business expectations are called Non Functional Testing.

Once the project is scheduled for system testing, every testing engineer validate Functional requirements.

Whenever 100% functional requirements validate, then tester can go through non functional requirements.

Types of Functional Testing

1) Smoke Testing / Build Verification Testing
2) Sanity Testing.
3) Positive / Negative Testing or input domain testing
4) Re Testing
5) Regression Testing
6) Database Testing - basic sql CRUD operation
   - Data Validation - insertions
   - Data Integrity – updations
   - Data Volume Testing – database capacity
7) Exhaustive Testing
8) End-to-End Testing
9) Adhoc Testing
10) Exploratory Testing
11) Monkey Testing
12)Globalisation and Localisation

(1) Smoke Testing / Build Verification Testing :

Functional testing starts with Smoke Testing.

In this type of testing, once build release by developer, to check whether application is stable or not, by checking basic functionality called Smoke Testing, also called as Build Verification Testing.

we will check whether the build is stable or not.

(2) **Positive / Negative Testing** : or **Input Domain Testing**

Positive Testing :

In this type of testing, application check with valid inputs called Positive Testing.

Negative Testing:

In this type of testing, application check with invalid inputs called Negative Testing.

Note:

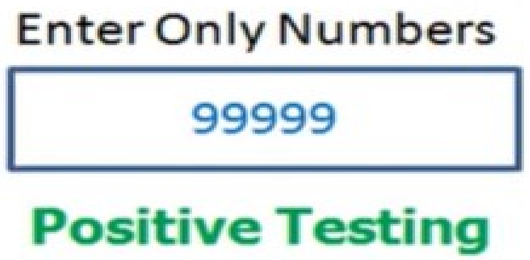
- The objective of positive testing is to confirm client requirements.
- The objective of negative testing is to find defects.

(3) **Re Testing** :

Once defect fixed by developer, to check its working as per customer requirements or not, checking or testing again and again that defected functionality called Re Testing.

Positive Testing

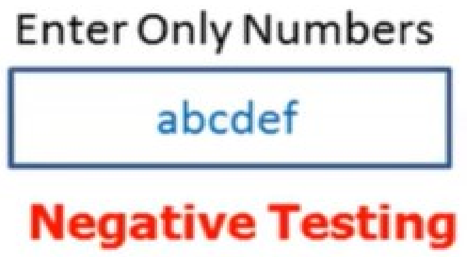
- Testing the application with valid inputs is called as Positive Testing.
- It checks whether an application behaves as expected with positive inputs.
- For example -



There is a text box in an application which can accept only numbers. Entering values up to **99999** will be acceptable by the system and any other values apart from this should not be acceptable.
To do positive testing, set the valid input values from 0 to 99999 and check whether the system is accepting the values.

Negative testing

- Testing the application with invalid inputs is called as Negative Testing.
- It checks whether an application behaves as expected with the negative inputs.
- For example -



Enter Only Numbers

abcdef

Negative Testing

Negative testing can be performed by entering characters A to Z or from a to z. Either software system should not accept the values or else it should throw an error message for these invalid data inputs.

Positive V/s Negative Test Cases

- **Requirement:**
  - For Example if a text box is listed as a feature and in FRS it is mentioned as Text box accepts 6 - 20 characters and only alphabets.
- **Positive Test Cases:**
  - Textbox accepts 6 characters.
  - Textbox accepts upto 20 chars length.
  - Textbox accepts any value in between 6-20 chars length.
  - Textbox accepts all alphabets.
- **Negative Test Cases:**
  - Textbox should not accept less than 6 chars.
  - Textbox should not accept chars more than 20 chars.
  - Textbox should not accept special characters.
  - Textbox should not accept numerical.

Re-Testing:

- Whenever the developer fixed a bug, tester will test the bug fix is called Re-Testing.
- Tester close the bug if it worked otherwise re-open and send to developer.
- To ensure that the defects which were found and posted in the earlier build were fixed or not in the current build.
- Example:

  - Build 1.0 was released. Test team found some defects (Defect Id 1.0.1 , 1.0.2) and posted.

  - Build 1.1 was released, now testing the defects 1.0.1 and 1.0.2 in this build is retesting.

NOTE:

======

- only changed feature testing is re-testing
- along with changed features checking overall functionalities is regression
- along with changed features checking main functionalities is sanity.

(4) **Regression Testing** :

Testing conducts on modified build to make sure there will not be impact on existing functionality because of changes like adding/deleting/modifying features.

- **Unit Regression Testing**:

  - Testing only the changes/modifications done by developer.

- **Regional Regression Testing**:

  - Testing the modified module along with the impacted modules.

  - Impact Analysis meeting conducts to identify the impacted modules with QA and Dev.

- **Full Regression**:

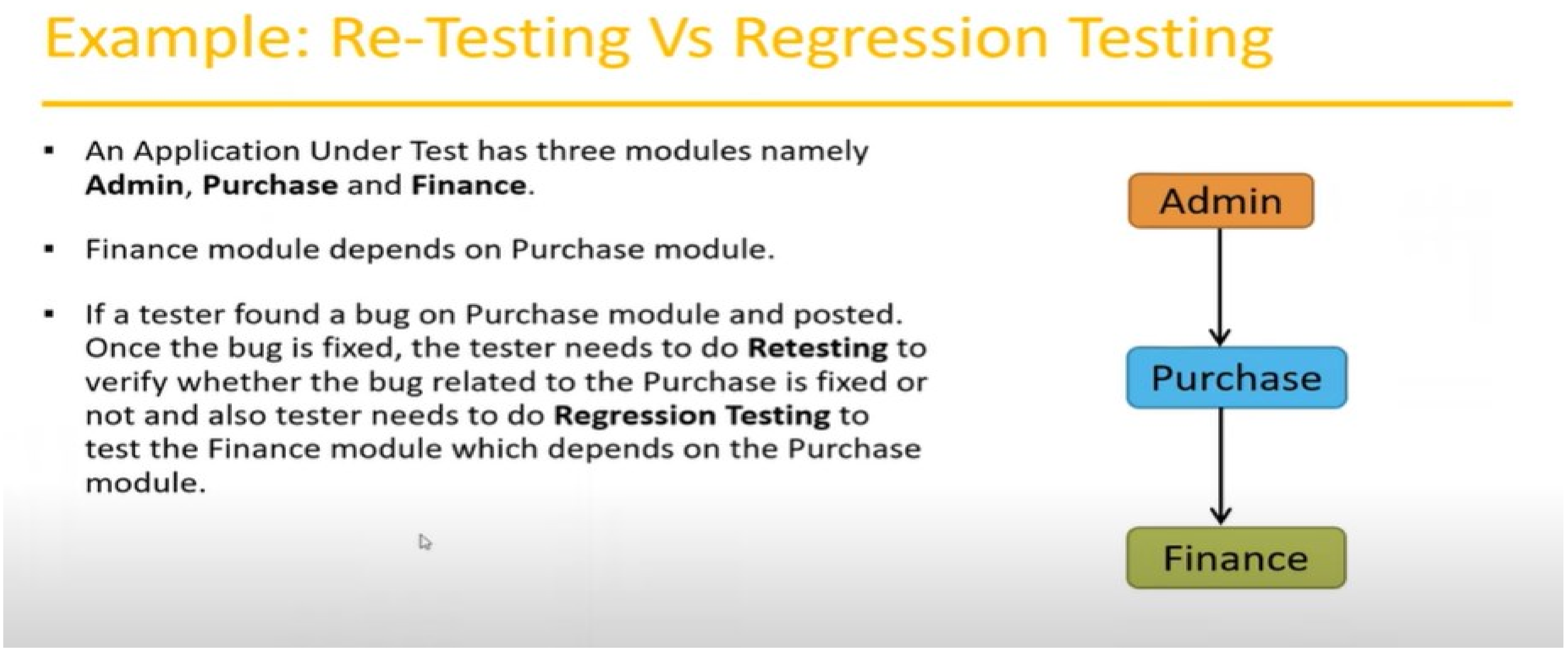
  - Testing the main features and remaining part of the application.

  - Ex: Dev has done changes in many modules, instead of identifying impacted modules, we perform one round of full regression.

We can ensure 100% customer satisfaction only by Regression testing.

But manually its required huge resources to perform regression testing, but it can handle through automation tools like Selenium easily.

Example:



Example: Re-Testing Vs Regression Testing

- An Application Under Test has three modules namely **Admin**, **Purchase** and **Finance**.
- Finance module depends on Purchase module.
- If a tester found a bug on Purchase module and posted. Once the bug is fixed, the tester needs to do **Retesting** to verify whether the bug related to the Purchase is fixed or not and also tester needs to do **Regression Testing** to test the Finance module which depends on the Purchase module.

$$\begin{tikzcd} \text{Admin} \arrow[d] \\ \text{Purchase} \arrow[d] \\ \text{Finance} \end{tikzcd}$$

(5) **Database Testing** : CRUD operations

It is a collection of data backend application of software, every data will be insert into database as a table format.

Checking database operations with respect to user operations is called Database testing.

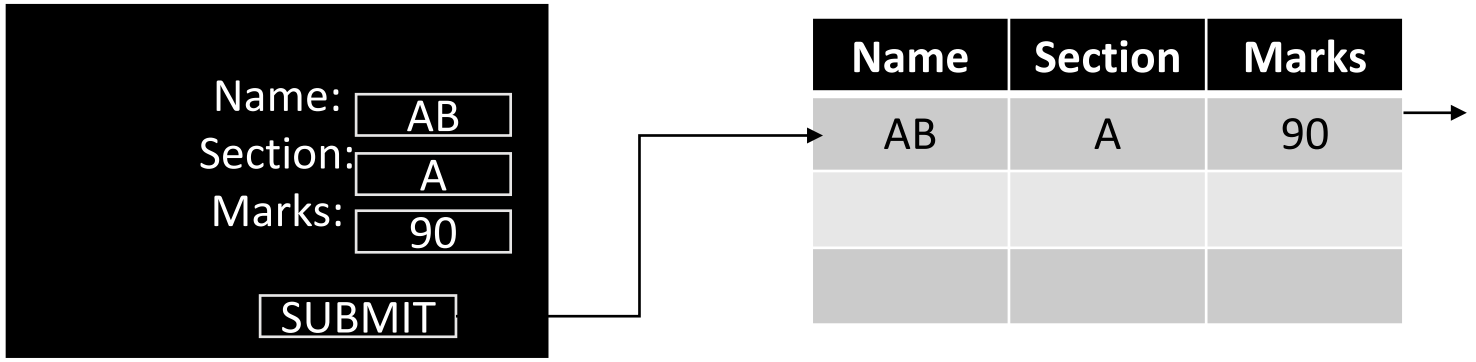
During database testing, we can perform below types of testing's.

(i) Data Validation :

In this type of testing, insert new data into the database and checking data will be correctly inserted or not, called Data Validation.

Student Form

Student Database

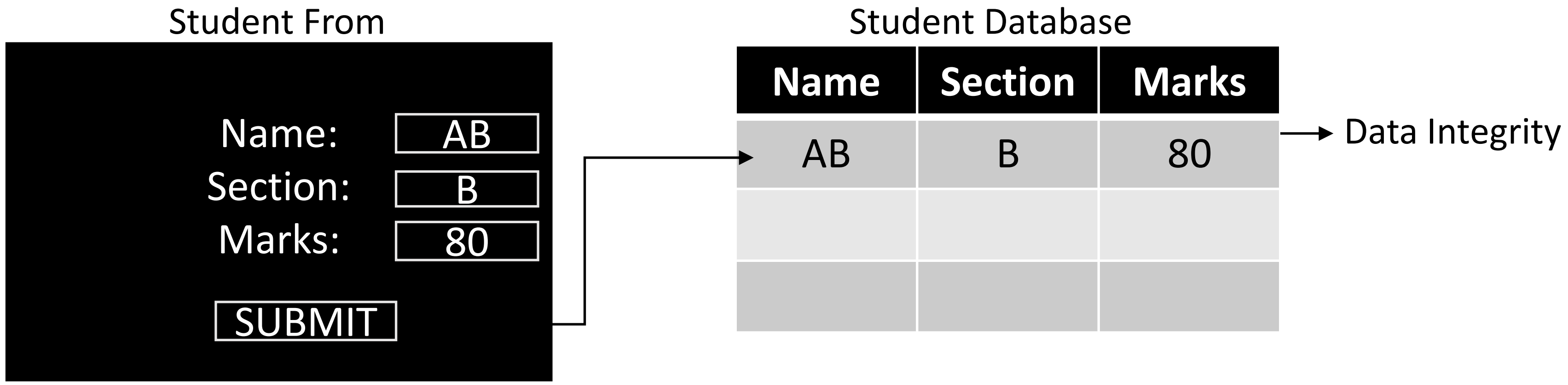


Name	Section	Marks
AB	A	90

Data Validation

(II) Data Integrity :

In this type of testing, update existing data and check correctly updated or not, called Data Integrity.



Name	Section	Marks
AB	B	80

(III) Data Volume Testing :

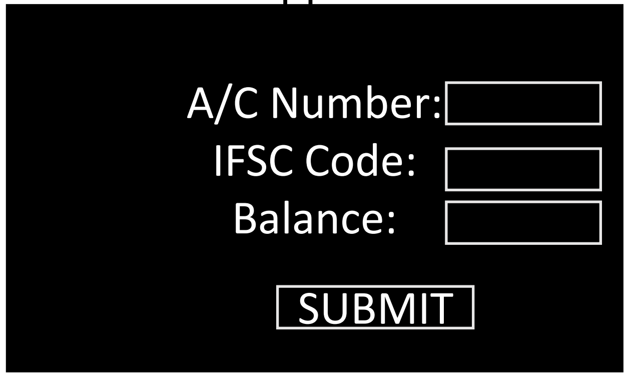
In this type of testing, checking capacity of database by entering sample data into the database, called Data Volume Testing or Data Memory Testing.

(6) **Exhaustive Testing** :

If we test a functionality with all possible inputs called Exhaustive Testing.

We can perform Exhaustive testing on Account Number, IFSC code functionalities, but to perform exhaustive testing on maximum balance its required huge resources. So it requires Automation help.

Example: Bank Application

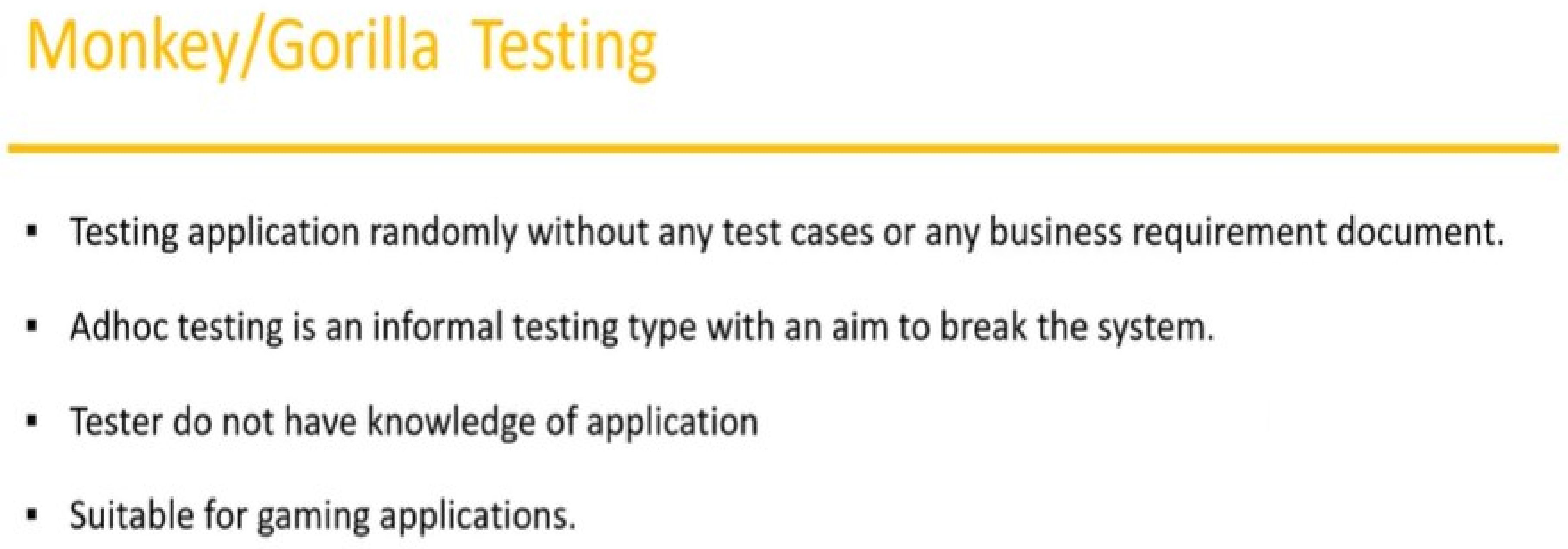


Exploratory Testing

- We have to explore the application ,understand completely and test it.
- Understand the application , identify all possible scenarios , document it then use it for testing.
- We do exploratory testing when the Application ready but there is no requirement.
- Test Engineer will do exploratory testing when there is no requirement.
- Drawbacks:
- You might misunderstand any feature as a bug (or) any bug as a feature since you do not have requirement.
- Time consuming
- If there is any bug in application , you will never know about it.

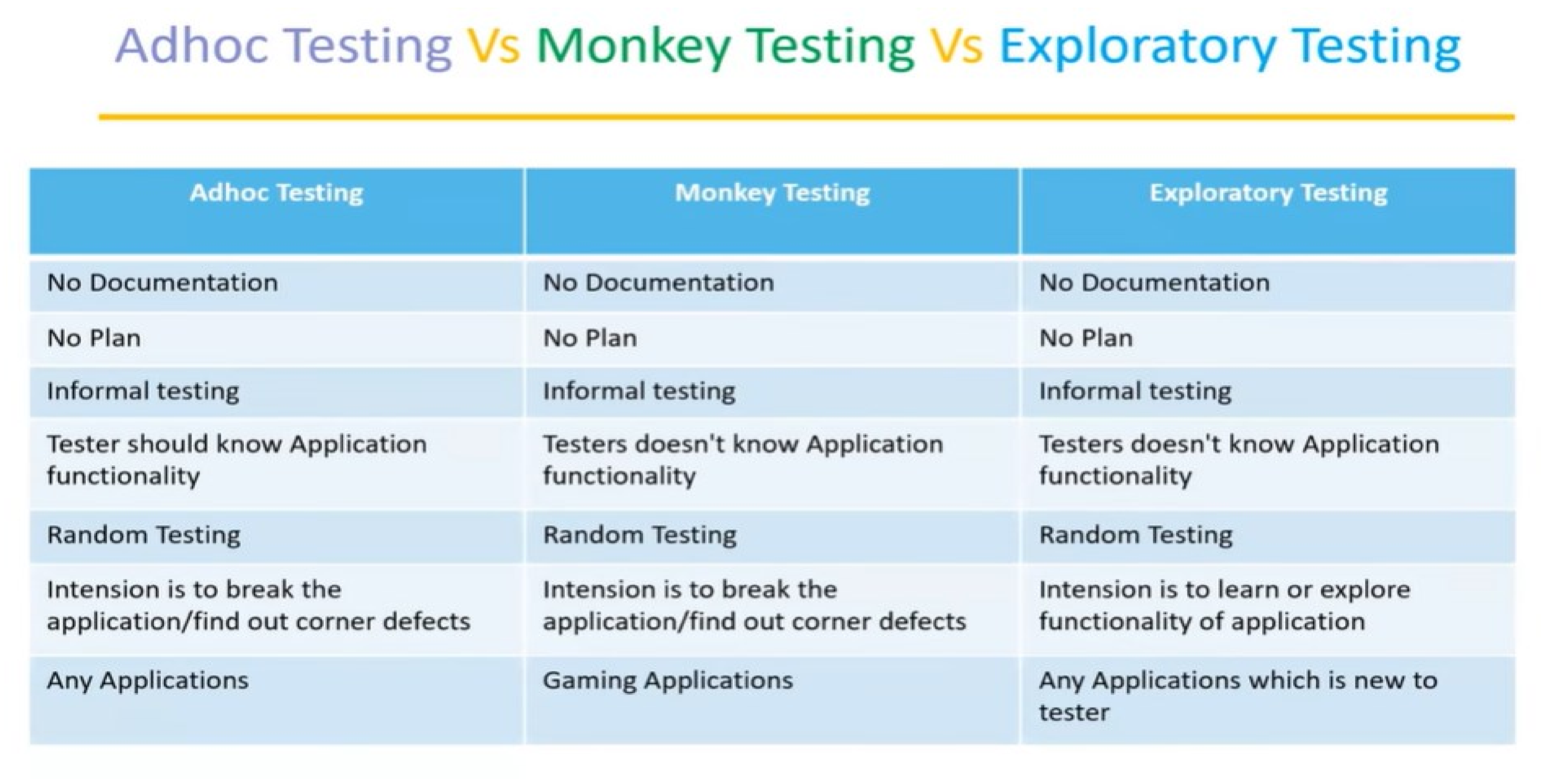
Adhoc Testing

- Testing application randomly without any test cases or any business requirement document.
- Adhoc testing is an informal testing type with an aim to break the system.
- Tester should have knowledge of application even thou he doesn't have requirements/test cases.
- This testing is usually an unplanned activity.



Monkey/Gorilla Testing

- Testing application randomly without any test cases or any business requirement document.
- Adhoc testing is an informal testing type with an aim to break the system.
- Tester do not have knowledge of application
- Suitable for gaming applications.



Adhoc Testing Vs Monkey Testing Vs Exploratory Testing

Adhoc Testing	Monkey Testing	Exploratory Testing
No Documentation	No Documentation	No Documentation
No Plan	No Plan	No Plan
Informal testing	Informal testing	Informal testing
Tester should know Application functionality	Testers doesn't know Application functionality	Testers doesn't know Application functionality
Random Testing	Random Testing	Random Testing
Intension is to break the application/find out corner defects	Intension is to break the application/find out corner defects	Intension is to learn or explore functionality of application
Any Applications	Gaming Applications	Any Applications which is new to tester

(7) **End-To-End Testing** :

If we conduct testing on application from starting to ending by all requirements of application called End-To-End Testing.

We can perform this testing whenever we have sufficient time.

(8) **Adhoc Testing** :

Due to lack of time, instead of testing all modules, testing main modules of application randomly called as Adhoc Testing.

Due to lack of time, under Adhoc testing, tester's will perform below types of testing's.

(i) Buddy Testing

(ii) Pair Testing

(iii) Exploratory Testing

(iv) Monkey Testing

(i) Buddy Testing :

Due to lack of time, testing team can join with development team to continue development and testing parallelly to complete as soon as possible called Buddy Testing.

(ii) **Pair Testing** :

Due to lack of time, testing team will combine as a pair or group and perform testing on the project. While working they can share knowledge and ideas of project called Pair Testing.

(iii) **Exploratory Testing** :

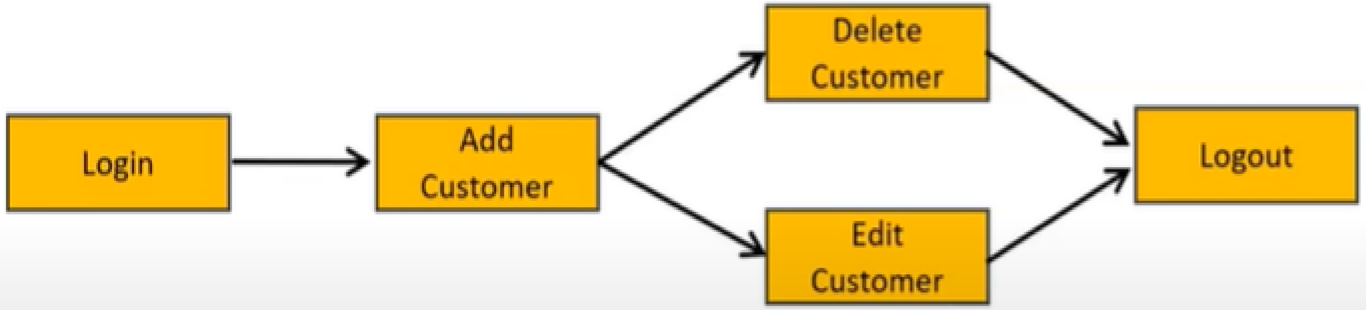
Due to lack of skills, explore knowledge on project by using search engine, interactive with seniors or verify project related documents, then work on project is called Exploratory Testing.

(iv) **Monkey Testing** :

Due to lack of time, without documents, process, standards work on the project randomly to break the system called Monkey Testing.

END-To-END Testing

- Testing the overall functionalities of the system including the data integration among all the modules is called end-to-end testing.



End-To-End Test

1) Login
2) ADD New Customer
3) Edit customer
4) Delete Customer
5) Logout

Globalization and Localization Testing

- **Globalization Testing:**
- Performed to ensure the system or software application can run in any cultural or local environment.
- Different aspects of the software application are tested to ensure that it supports every language and different attributes.
- It tests the different currency formats, mobile number formats and address formats are supported by the application.
- For example, Facebook.com supports many of the languages and it can be accessed by people of different countries. Hence it is a globalized product.

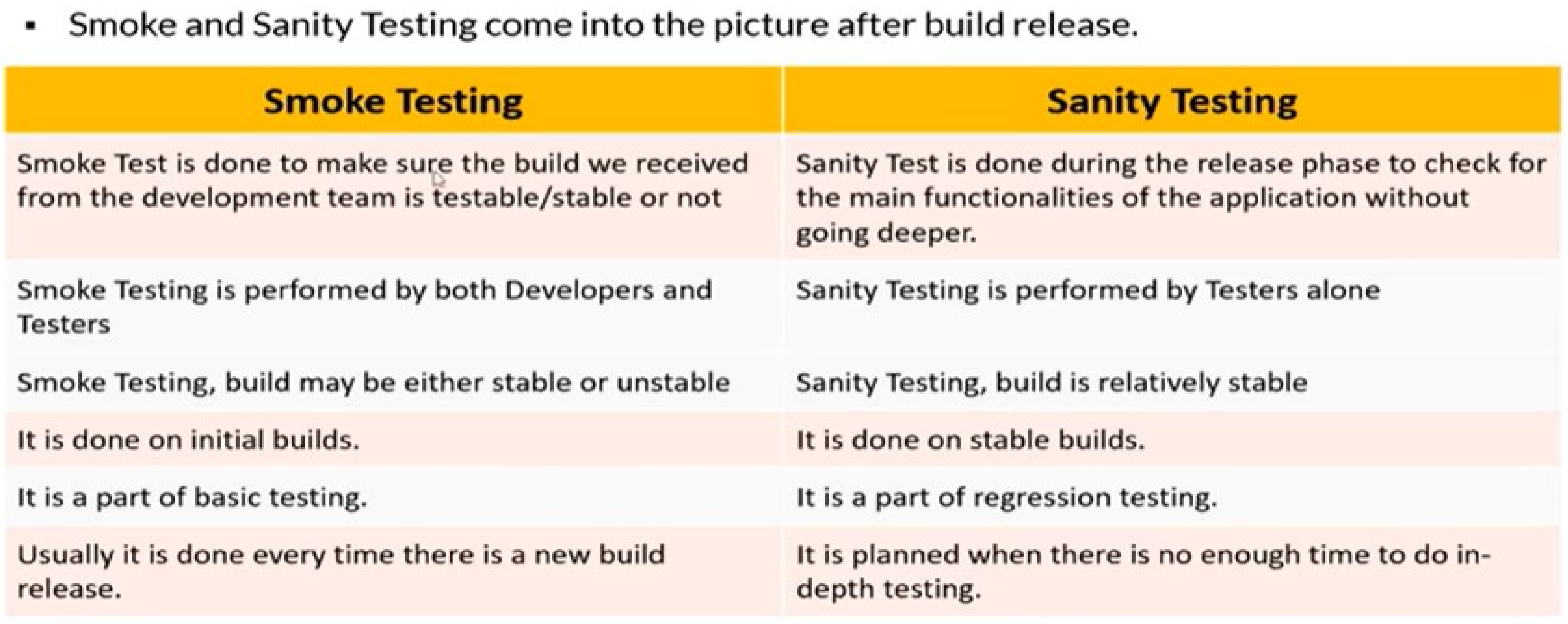
- **Localization Testing:**
- Performed to check system or software application for a specific geographical and cultural environment.
- Localized product only supports the specific kind of language and is usable only in specific region.
- It testes the specific currency format, mobile number format and address format is working properly or not.
- For example, Baidu.com supports only the Chinese language and can be accessed only by people of few countries. Hence it is a localized product.

Sanity Testing :

The Sanity testing is used to determine that the changes or the functionality are working as expected. If the sanity testing fails, software product is rejected by the testing team to save time and money.

It is performed only after the software product has passed the smoke test and QA team has accepted for further testing.

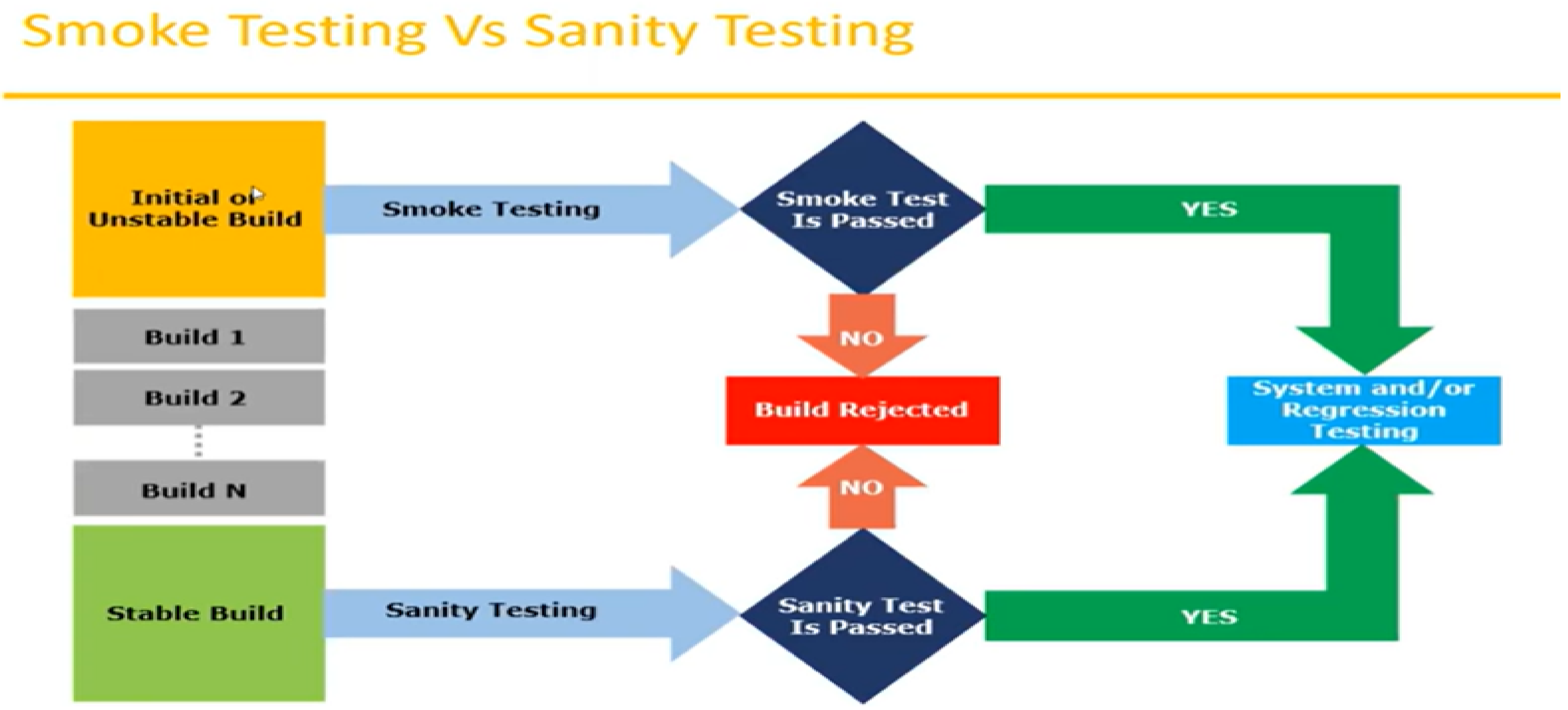
Smoke Vs Sanity Testing



- Smoke and Sanity Testing come into the picture after build release.

Smoke Testing	Sanity Testing
Smoke Test is done to make sure the build we received from the development team is testable/stable or not	Sanity Test is done during the release phase to check for the main functionalities of the application without going deeper.
Smoke Testing is performed by both Developers and Testers	Sanity Testing is performed by Testers alone
Smoke Testing, build may be either stable or unstable	Sanity Testing, build is relatively stable
It is done on initial builds.	It is done on stable builds.
It is a part of basic testing.	It is a part of regression testing.
Usually it is done every time there is a new build release.	It is planned when there is no enough time to do in-depth testing.

Smoke Testing Vs Sanity Testing



Non-Functional Testing

Once the project is validated by all functional requirements to ensure customer expectations, testing engineer can perform below types of non-functional testing’s.

(1) GUI Testing - imp

(2) Usability Testing - imp

(3) Compatibility Testing - imp

(4) Security Testing – imp

(5) Installation Testing

(6) Recovery Testing

(7) Standard Testing

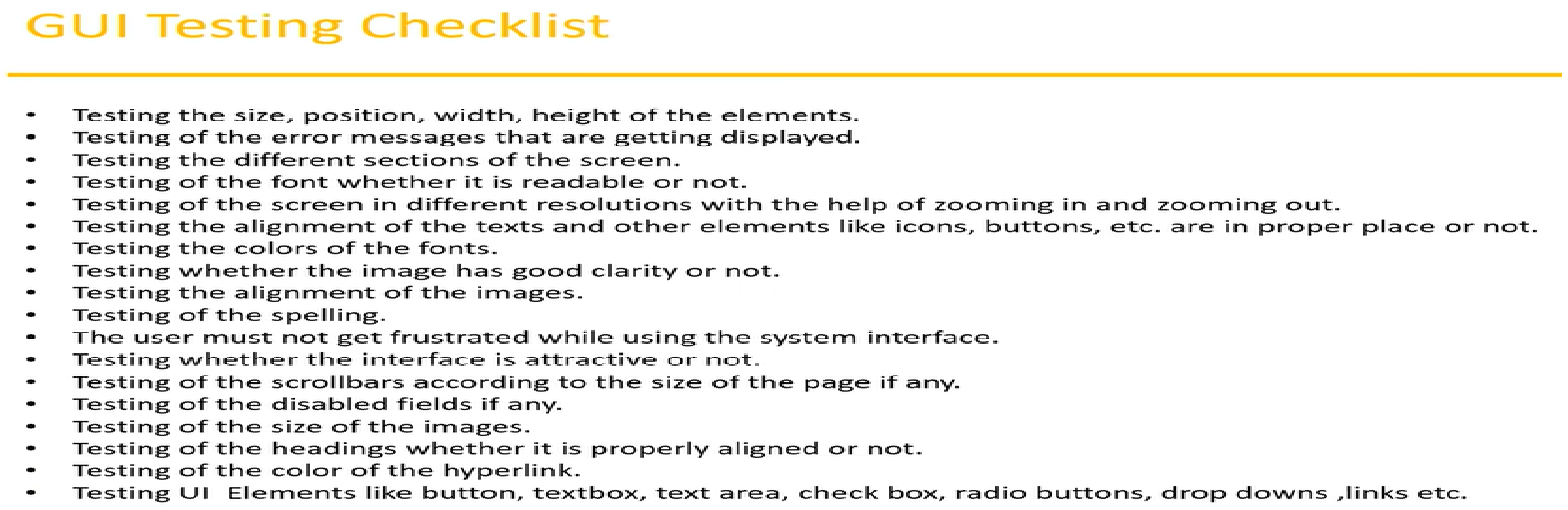
(8) Performance Testing - vimp

(1) **GUI Testing** :

Graphical User Interface Testing or GUI testing is a process of testing the user interface(UI) of an application.

A GUI includes all the elements such as menus, checkbox, buttons, colours, fonts, sizes, icons, content and images.

GUI testing checklist:



GUI Testing Checklist

- Testing the size, position, width, height of the elements.
- Testing of the error messages that are getting displayed.
- Testing the different sections of the screen.
- Testing of the font whether it is readable or not.
- Testing of the screen in different resolutions with the help of zooming in and zooming out.
- Testing the alignment of the texts and other elements like icons, buttons, etc. are in proper place or not.
- Testing the colors of the fonts.
- Testing whether the image has good clarity or not.
- Testing the alignment of the images.
- Testing of the spelling.
- The user must not get frustrated while using the system interface.
- Testing whether the interface is attractive or not.
- Testing of the scrollbars according to the size of the page if any.
- Testing of the disabled fields if any.
- Testing of the size of the images.
- Testing of the headings whether it is properly aligned or not.
- Testing of the color of the hyperlink.
- Testing UI Elements like button, textbox, text area, check box, radio buttons, drop downs ,links etc.

(2) **Usability Testing : easiness of the application**

In this type of testing in customer point of view, application is user friendly or not means user is able to understand and operate the application easily or not.

During this testing, validates application provided context sensitive help or not to the user., so that non-technical people also should able to operate the developed application easily.

(3) Compatibility Testing : Configuration Testing:

In this type of testing, to check the application compatibilities, validate both software compatibilities and hardware compatibilities.

Software Compatibility :

In this testing application validate by various operating systems, various browsers supporting or not called Software Compatibility.

Hardware Compatibility :

In this testing application validate by various hardware devices configuration supporting or not called Hardware Compatibility.

(4) **Security Testing : security team will be separate.**

To check whether application is securable or not, security tester will perform below types of testing.

Authentication Testing:

In this type of testing, we need to check whether the valid user is accepted or not and invalid user is rejected or not, called Authentication Testing.

Authorization Testing (Access Control) :

In this testing, whenever the login as a level of user into the application, check whether he crossing any beyond user limits or not, called Authorization Testing or User Permission Testing.

Encryption Testing :

In this testing, to check whether the securable data default converted into encrypted format or not, called Encryption Testing.

(5) **Installation Testing :**

In this testing, we need to follow installation guidelines which are defined under installation document.

During this test, we have to verify below steps:

- While installation, application is user friendly or not.
- After installation, check memory.
- Verify uninstallation process.

(6) **Recovery Testing** : backenddatabase.

In this type of testing, checks how the system handles unexpected events such as power failure, application crash etc.

checking the system change from abnormal to normal.

(7) **Standard Testing :**

In this type of testing, we need to check whether application developed by following company standards or not such as ISO [International Standard Organization] etc.

(8) **Performance Testing :**

Speed in processing to check performance of the application, testers will follow below types of testing tools Jmeter, loadrunner etc…

(i) Load Testing :

The execution of software under customer expected configuration and customer expected load , to estimate the speed in processing and is there any chances of breaking the application, called Load Testing

Note: gradually (slowly) increase the load on the application and check the speed of the application.

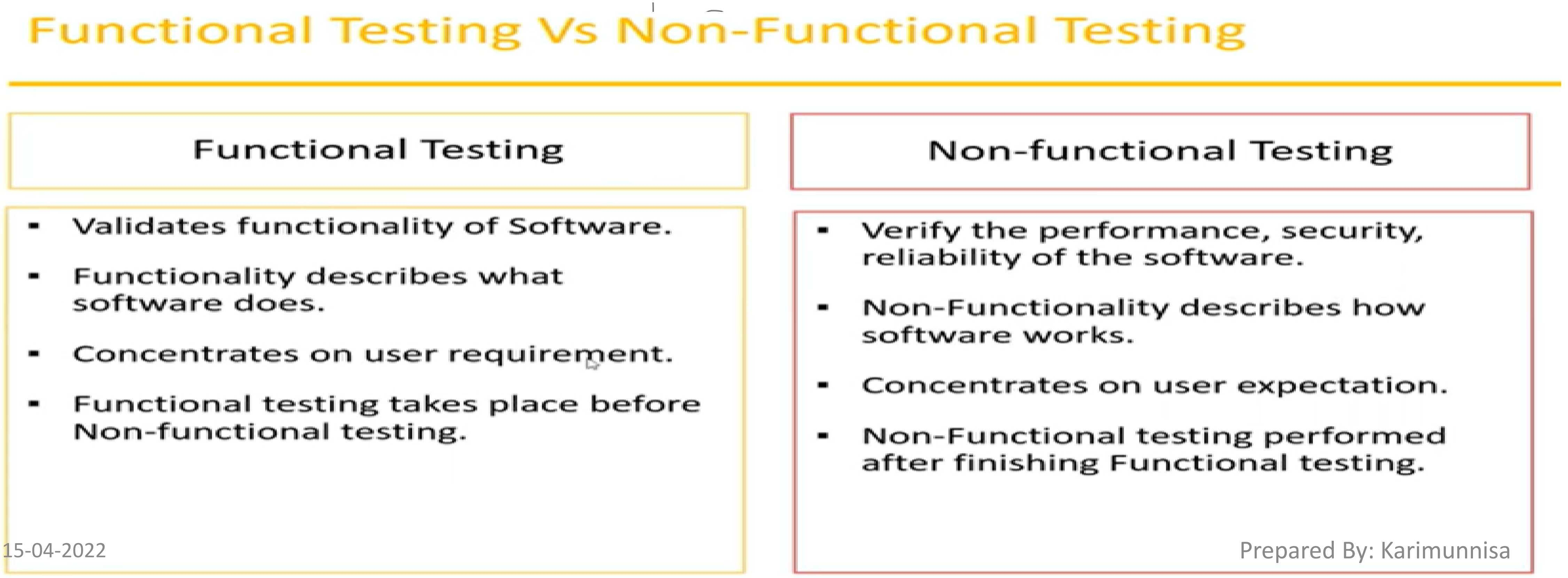
(ii) <u>Stress Testing</u> :

The execution of software under customer expected configuration but more than customer expected load, called Stress Testing.

Note: suddenly increase/decrease the load and checks the speed and stability of the application.

(iii) **<u>Volume Testing</u> : backend**

Volume means size, testing how much data the application can handle.



Functional Testing Vs Non-Functional Testing

Functional Testing	Non-functional Testing
Validates functionality of Software.	Verify the performance, security, reliability of the software.
Functionality describes what software does.	Non-Functionality describes how software works.
Concentrates on user requirement.	Concentrates on user expectation.
Functional testing takes place before Non-functional testing.	Non-Functional testing performed after finishing Functional testing.