Module Topics

Let us take a quick look at the topics that we will cover in this module:

- Manual Testing.
- Automation Testing.
- Unit Testing.
- Integration Testing.
- Smoke-Sanity Testing.

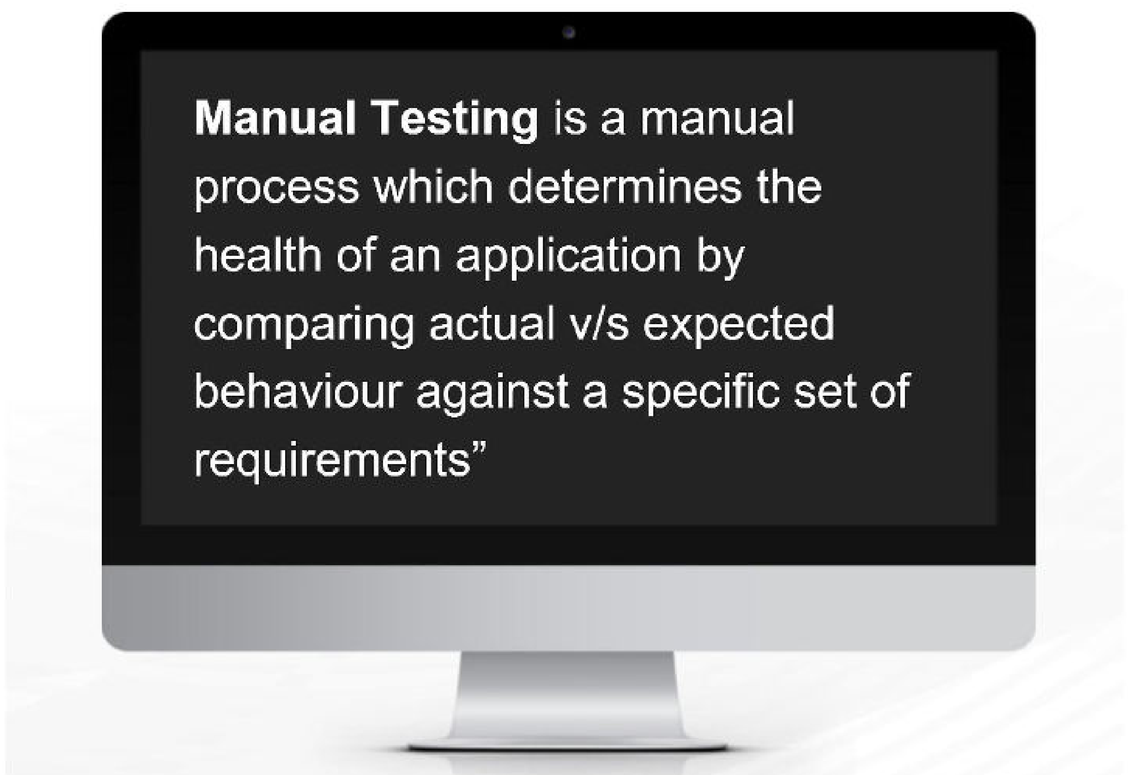
Facilitator Notes:

Explain the module topics to the participants.

You will learn about the following topics in this module:

- Manual Testing.
- Automation Testing.
- Unit Testing.
- Integration Testing.
- Smoke-Sanity Testing.

1.4.1 Manual Testing



Facilitator Notes:

Explain to the participants about ‘Manual’ in ‘Manual Testing’ approach

As the name suggests ‘Manual’ which means testers will be ‘Manual Testers’ and the testing they will perform is ‘Manual Testing’. Indirectly it means manual testers will have knowledge of the application and they will conduct the manual testing without the use or help of any Automation Testing tool.

Manual Testing has been the most primitive form of testing. Testing started with Manual Testing approach only. And any new application must undergo a thorough Manual Testing before checking the feasibility of Automation Testing

Manual Testing approach opened gates to other testing approaches. Like all other approaches this also means that the application should behave in accordance with the requirements and any deviation is reported.

Manual Testing includes a lot of Testing Artifacts which are used from the beginning till the end of the Testing Life Cycle. Sometimes they are hard to maintain, are boring and time consuming. But all these boring bits have their special place

Simply put ***"Manual Testing is a manual process which determines the health of an application by comparing actual v/s expected behaviour against a specific set of requirements"***

1.4.1 Manual Testing – How to Approach?

Listed below are the steps for approaching manual testing :

Facilitator Notes:

Explain to the participants briefly about a normal manual testing project.

Only following a Requirement Document is not enough for a good manual testing project. There has to be proper review and sign-off with the latest version of the Requirement Document to be read and followed by the testers. Proper communication channels should be opened so that testers can ask doubts or raise queries in case they find certain ambiguity in the requirement document. Asking questions and raising doubts should be highly encouraged. Testers should also be allowed to give suggestions to improve the document. Once the finalized document is with the testers, testing can begin.

All the Testing Artifacts should be prepared. The Test Strategy document along with the Test Plan which defines the Scope is of highest importance. With Agile methodologies followed in almost every Software Company today, the good old practice of Test Strategy and Test Plan is getting sidelined but the same action is creating a lot of conflict and last moment defect creep-in for a lot of Projects. hence, it is always advisable to prepare a good solid Test

Strategy which defines the Scope precisely. The Requirement Traceability Matrix (RTM) and the TCER (Test Conditions Expected Results) and other documents should be formulated depending on organization approved template formats.

Test Cases writing is probably the heart of the Manual Testing process. A good, simple, full coverage of scenarios Test Case is very important. The linguistics should be simple and well understood, the intent should be clearly defined and it should cover all aspects of functionality. Coverage should not be compromised.

Review, Inspection and Walkthrough are generally skipped in almost all software companies because of time crunch and high paced deliverable environments and methodologies. But a review from a senior always helps in identifying problems beforehand which will save lot of time later in the Project. Junior Testers or less experienced testers should always get their work reviewed from senior testers.

Test Execution should be done in accordance to the test case prepared. Sometimes, manual testers think they know in and out of the application and they can test the application on their own and they do not need to execute test cases. This leads to certain defect leakage later. So test execution should be done precisely with respect to the test cases that have been designed.

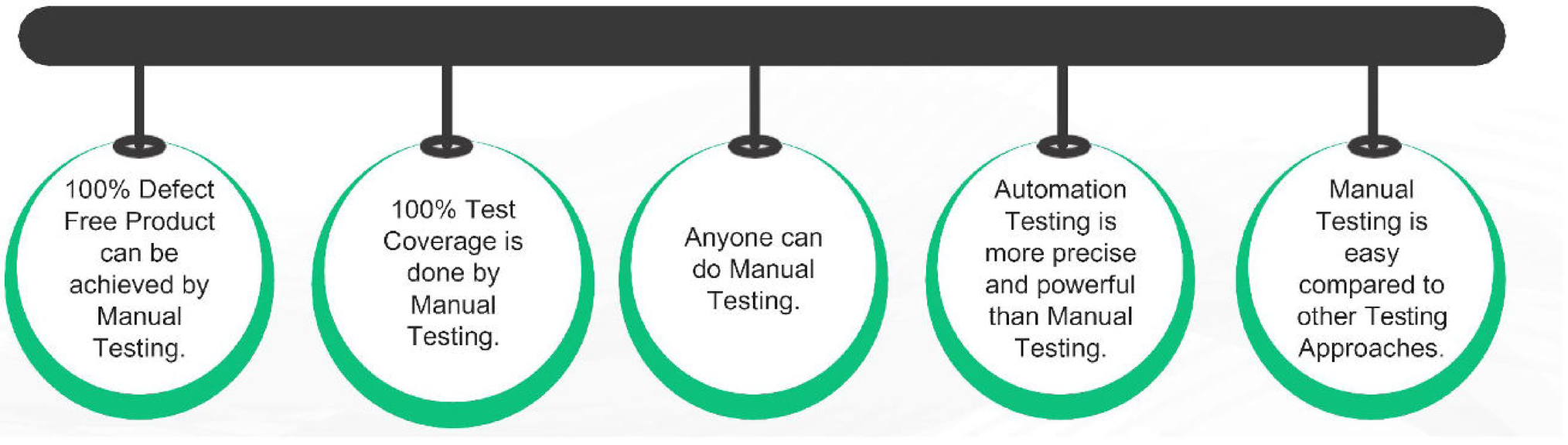
Defect/Bug reporting – this is another very important branch of manual testing. Finding a bug is not the only responsibility of a manual tester but reporting the same in correct channels is also very important and this should be done with precision as sometimes developers and other depending teams tend to deny responsibility if the bug reported is not understood or is not reproducible.

Following up with the status quo of the defect whether it is still in open state or it is fixed or it has some discrepancy is also important. Closer vigil needs to be kept. In case, the defect has been fixed and it has been re-assigned to the tester then the tester should test the defect in the requisite environment and report its current status immediately.

Test Summary – These reports are good practices to actually give a depiction of (how+what+when+by whom) things happened during the course of manual testing.

1.4.1 Manual Testing – Myth and fallacy

Manual testing is generally treated as most easy job. Let's look at some myths of manual testing.



Facilitator Notes:

Explain to the participants briefly about a manual testing project.

No application can be 100% defect free. Take for any product which we use in our day to day life. Even Google and Facebook and Amazon have defects. They are always evolving. Hence, no one can say that manual testing will cover and identify all the defects. It is a fallacy. Same is true about coverage. Try as hard as may, some or the other scenario will get skipped or will not come in terms of thought process of testing team. Manual Testers do tend to work hard and do their best to have maximum test coverage but somehow labeling that 100% test coverage is possible is a pure myth.

Anyone can do Manual Testing as if it is a thankless job. This is really not true. Manual Testing needs skills and not one but multiple skill sets which we shall discuss later.

Automation Testing is powerful – Agreed but it cannot be said that Automation Testing is more powerful than Manual Testing. As manual testing can uncover scenarios and defects which Automation Testing might not be able to because of coding constraints or some logic which will be hard to design.

Manual Testing is not easy. It requires skills, thorough understanding and certain functional and technical approaches to get the job done. And testers are more at risk compared to developers as they need to authenticate the product.

What did you learn so far ?

1. Manual Testing is outdated and soon to be deprecated ?
   A) True
   B) False

Answer: 1 : B

2. Manual Testing is recommended for testing negative scenarios than Automation Testing
   A) True
   B) False

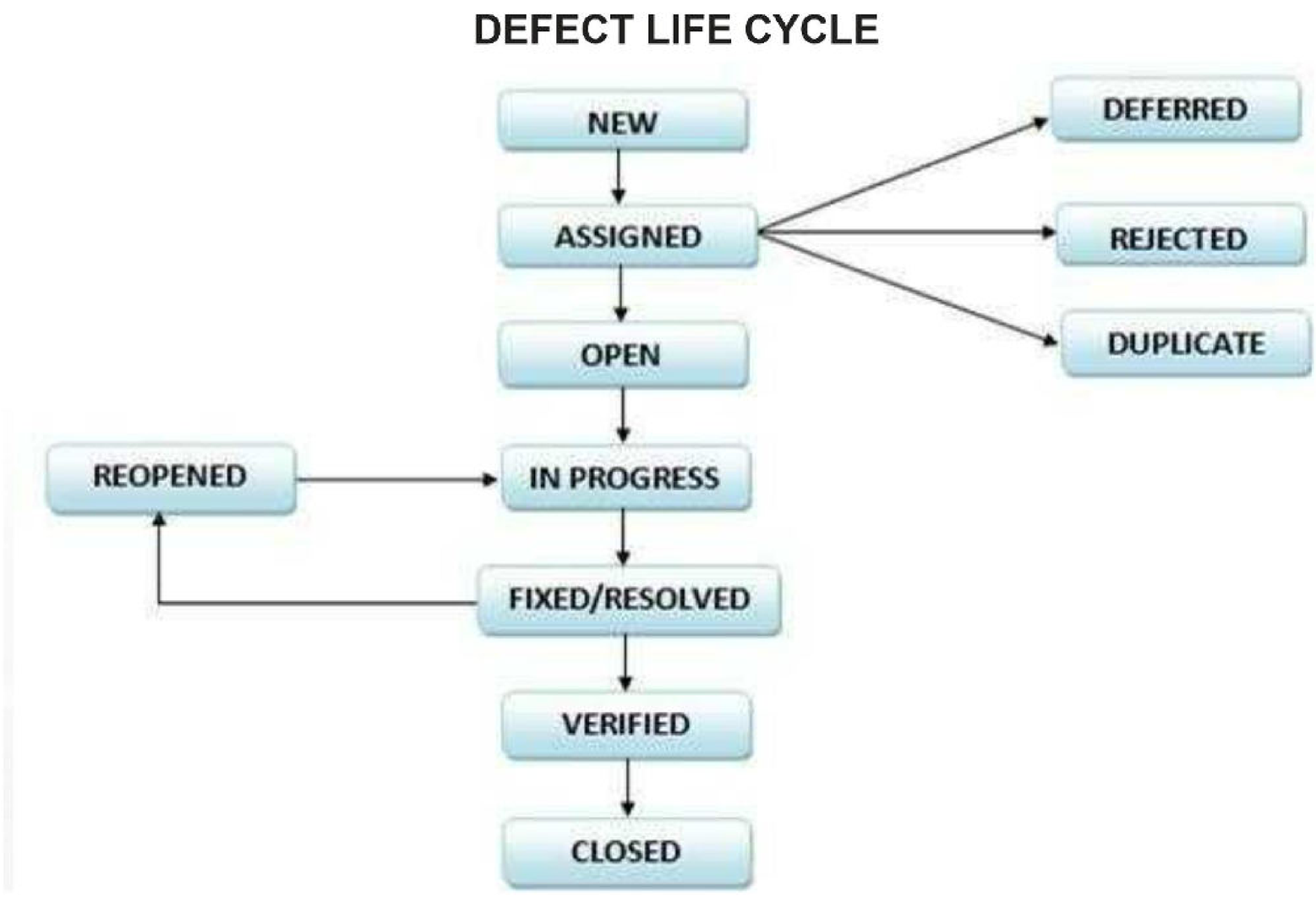
Answer: 2 : A

3. Smoke and Sanity testing are mandatory in Manual Testing but can be skipped in Automation ?

   A) True
   B) False

Answer: 3 : B

1.4.1 Manual Testing – Defect Life Cycle



Facilitator Notes:

Explain the participants briefly about a defect life cycle and its significance.

The above image shows the workflow.Ok let's say for example you are a tester and you found a defect. And this defect has never been found before.

NEW – The first state of the defect.

ASSIGNED – Now you will assign this to a certain developer or development team. So the status of the defect will become assigned from new.

OPEN – When the developer has acknowledged the defect, the status of the defect now becomes open.

INPROGRESS–Whenthedeveloperstartsworkingonthedefect.ThestatebecomesInProgress.
FIXED/RESOLVED – Developer marks this defect as fixed once he has done necessary modifications in the code and has fixed the bug.

PENDING RETEST - After fixing the defect, the developer assigns the defect to the tester for retesting the defect at their end and till the tester works on retesting the defect, the state of the defect remains in 'Pending Retest'.

RETEST - At this point, the tester starts the task of working on the retesting of the defect to verify if the defect is fixed accurately by the developer as per the requirements or not.

REOPENED - If any issue still persists in the defect then it will be assigned to the developer again for testing and the status of the defect gets changed to 'Reopen'.

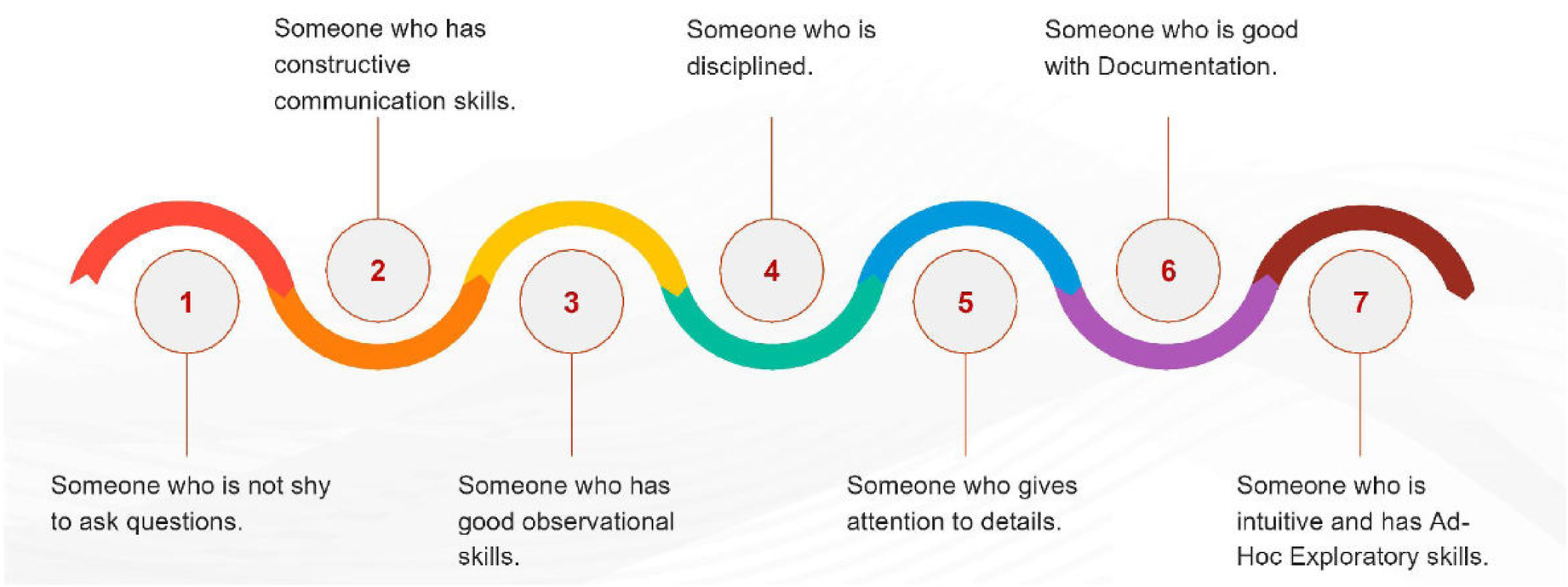
VERIFIED - If the tester does not find any issue in the defect after being assigned to the developer for retesting and the defect has been fixed accurately then the status of the defect gets assigned to 'Verified'.

CLOSED - When the defect does not exist any longer then the tester changes the status of the defect to 'Closed'.

There are chances that the developer might reject the defect if he finds it invalid.Sometimes some defects cannot be fixed in this iteration. It is kept under consideration to be fixed in the next release and is marked as **Deferred**.When the developers are not able to understand how to reproduce the defect and they mark it as **Could not Reproduce.**

1.4.1 Manual Testing – Qualities of a good Manual Tester

Some qualities of good manual testers:



Facilitator Notes:

Explain to the participants briefly about a good manual tester and an average manual tester

What separates a good manual tester vs an average manual tester is not only dependent on skill factor but it also depends on the thought process and personal skills to enhance one own self to become a better software tester. The one aspect that is common for anyone is asking questions. When you ask questions – that means you are keen and you want to learn and see how things are and are more interested.

Testers should be able to do Constructive Communication. For testers it just becomes even more important as an average communication leads to extra intrusion of other people who will try to cover up for the lack of communication or err in communication. It gives good language in test documents and would be well appreciated by clients if they are of a Foreign Origin.

Observation is very important whether you are manually testing a web application or a mobile app. Yes it needs little experience to actually bring the observational skill to use but that eye is definitely going to help a manual tester to actually catch hold of any defect or deviation which might get overlooked otherwise

Disciplined approach and attention to details are very important aspects for a Manual Tester as manual testing involves quite a lot of maintenance and creation of documents. Sometimes the quantity of the same could be overwhelming. So a better organized tester is a better tester.

As mentioned in the above point, an organized tester (in this case with documentation) is prone to find answers to questions with lesser hassle. A good manual tester definitely needs to possess this quality of out of the blue doing something to find certain deviation even though all sorts of approvals have been given for the release of that product. It could be anything. The approach could be negative too but anything within the scope of testing should always be encouraged.

1.4.1 Manual Testing vs Automation Testing

- Do not require coding skills
- More documents
- Takes longer time
- High Costs
- Repetitive in nature
- Application need not be stable to begin
- Done by Manual Testers

- Requires Coding skills
- Documents are less
- Takes shorter time once automation suite is ready
- Comparatively low in costs
- Repetitive but can be edited
- Application has to be stable
- Done by Automation Testers

Facilitator Notes:

Tell the basic difference between a manual tester and an automation tester

In 2020, we are moving towards the Robotic frame of life where in everything is going to get automated or at least the step towards it, so why Software Testing stay behind. Cometh the need of the hour, even the Manual Testing approach is slowly moving towards the automation route.

But is the transition going to be that easy. What happens to Manual Testing then ? Will it be deprecated ? There are lot of questions to be answered and some of them quite genuine in that regard. Let's understand few facts what led to all this.

Software Applications with time have become quite complicated, Robust and the vastness also has exponentially increased. To handle the vastness, manual testing is not going to be an effective solution any more. Yes Manual Testing will have a big and important role to

play in the first iteration but then the subsequent iterations will have to be assessed with something which will be time saving, more cost effective and less human intervention. Hence Automate it.

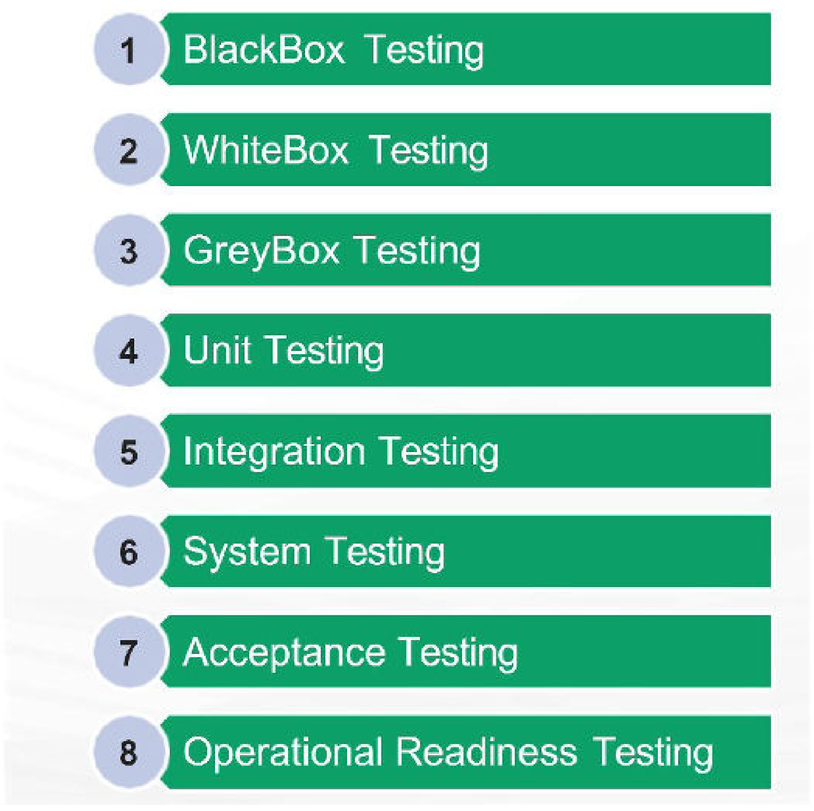
If we actually go by the book, yes Automation Testers need to hone their coding skills. Just looking into the Business Document and Functional Document and writing test cases will not suffice. Automation testers have to code in any of the language binding depending upon the automation testing tool which they are using

Automation Testing does help cut down costs and also saves a lot of time. Plus integration with CI/CD tools like JENKINS gives quite a peace of mind to let the test cases run automatically with just a timer set up.

The only major drawback is that if the application is unstable, then Automation Testing might not be effective rather it will be a complete waste of time.

1.4.1 Manual Testing - Types

The most common types of testing are listed below:



Facilitator Notes:

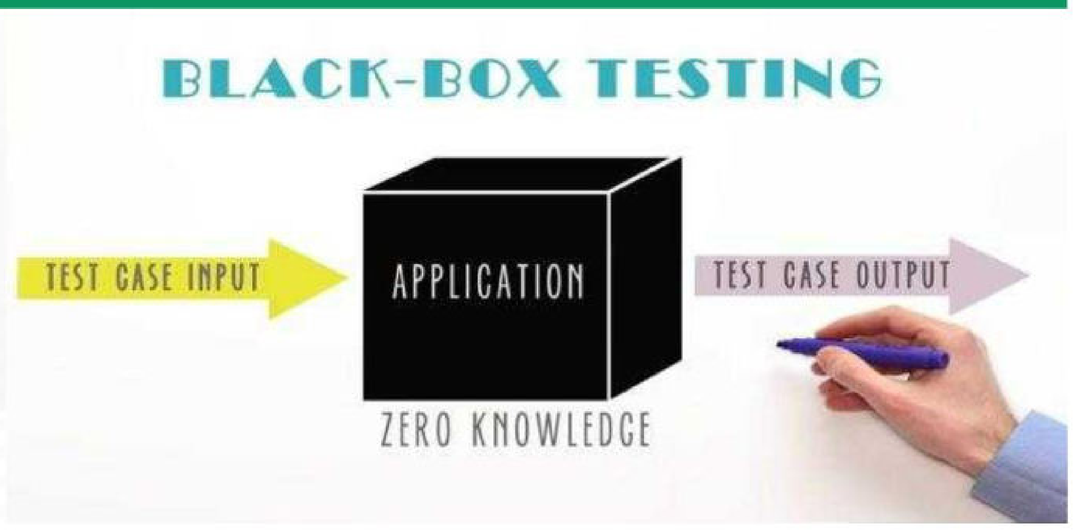
Tell the participants brief about all the testing types mentioned

The foremost thing to understand here is if the end goal is to determine defects, then why have so many forms and types. It is not that simple as just word makes it seem like. There is lot more to it and it is quite vast. Every stage of development in tandem with testing will have a certain way of looking into things. What Whitebox testing can determine, blackbox might not and what Acceptance testing can reveal a Blackbox testing might not. So every type has its own unique importance and that is what we will understand in further discussion

1.4.1 Manual Testing - Types

BlackBox Testing

This technique' focus is mostly on the external layer of the software such as technical specifications, design, and client's requirements to design test cases. The technique enables testers to develop test cases that provide full test coverage.



Facilitator Notes:

Tell the participants what does zero knowledge mean in the above slide.

Black Box testing technique does not need the tester to have in depth coding knowledge of the application. What matters most here is the knowledge of possible inputs, logical operation and expected output. Having these 3 in the kitty, the Tester or Black Box Tester is good to go.

Let's discuss little bit about the various Black Box Testing Techniques:

1. **Boundary Value Analysis:-**

   Like any application logic becomes vulnerable and susceptible at boundaries even in software applications. A Developer has more chances of introducing a faulty logic which might not be completely wrong but might be somewhat wrong at the border lines. Testing any application with the minimum value and maximum value parameter and transition or switch the next parameter should be done with this technique. There could be many real life examples like "***Checking February calendar for leap year, non leap year and checking the value in March", "checking for bulk items which might give a discount scheme from item 10th to 19th [so item no 19th, 20th and 21st will come under BVA] and then another discount scheme from item 20th to 29th [so item no 29th, 30th and 31st will come under BVA] "***

   Similarly there could be multiple exams. So basic understanding is the minimum value, just below minimum value, maximum value and just above exact value – these areas needs to be tested.

2. **Equivalence Partitioning**:-

   In this technique, the entire range of input data is divided into different partitions. All possible test cases are considered and divided into logical set of data. One test value is picked during each execution. So we divide the total input data into sets of data and then test them to find vulnerabilities. The only de-merit of this technique is that "it is not too helpful when fewer test cases should cover maximum requirements".

3. Decision Table Testing:-

In this technique, test cases are designed on the basis of the decision tables that are formulated using different combinations of inputs and their corresponding outputs based on various conditions and scenarios adhering to different business rules. It is used in both testing and requirements analysis. It manages complex business logics in simplified tabular format which gives a simplistic explanation. One advantage of using decision tables is that they make it possible to detect combinations of conditions that would otherwise not have been found and therefore, not tested or developed. The requirements become much clearer and you often realize that some requirements are illogical, something that is hard to see when the requirements are only expressed in text.

A disadvantage of the technique is that a decision table is not equivalent to complete test cases containing step-by-step instructions of what to do in what order. When this level of detail is required, the decision table has to be further detailed into test cases.

4. State Transition Testing:-

It is a Black Box Testing technique. The tester will give certain inputs [keep in mind the inputs can differ from positive to negative]. The tester determines the behavior of the application under certain permissible finite limits of inputs. And notices the system's response the moment the finite number of attempts gets exhausted.

Let's understand this via an example. You have an ATM Debit Card and you go to the ATM Machine.

Negative Attempts:-

1st Attempt – wrong pin and system denies access and warns you 2 more attempts left

2nd Attempt – wrong pin and system denies access and warns you 1 more attempt left

3rd Attempt – wrong pin and system denies access, blocks card and gives message "card has been blocked due to suspicious activity"

Positive Attempt:-

1st Attempt – right PIN and system acknowledges access

So here is that the state of the system changes with every attempt. This form of testing in real life scenario is called State Transition Testing

5. Use Case Testing:-

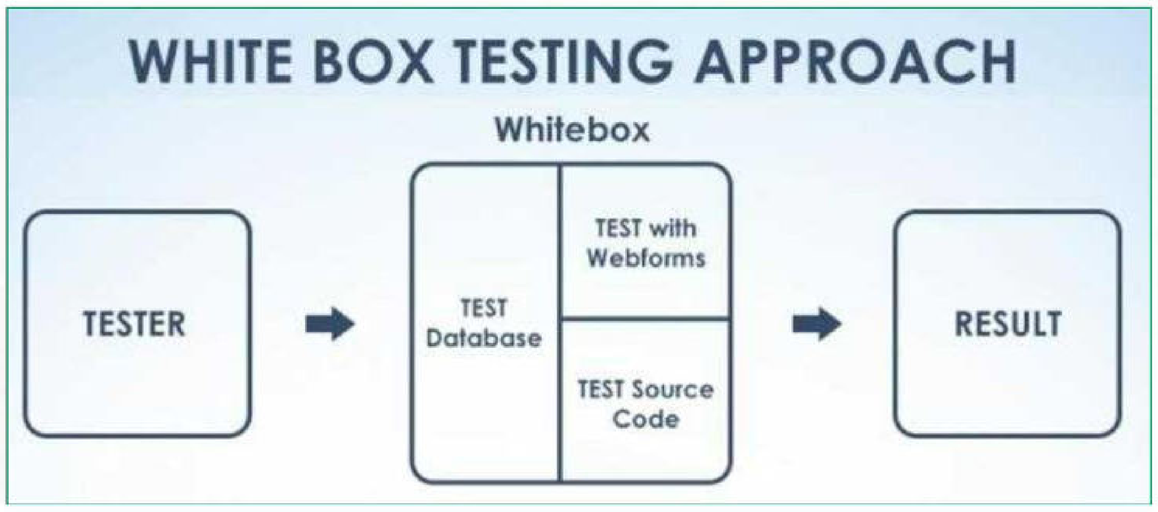
It is a descriptive form to address the functionality of the system at a user level. Specific negative and positive scenarios are not segregated. Just the intent of testing is directly mentioned in one line

For example : Login into the application using valid credentials and validating system response

Enter invalid credentials and attempting login and validating system response.

WhiteBox Testing

The structure-based or white-box technique design test cases based on the internal structure of the software. This technique exhaustively tests the developed code. Developers who have complete information of the software code, its internal structure, and design help to design the test cases.



Facilitator Notes:

Tell the participants the difference between the knowledge about the business logic in whitebox testing which is not there in blackbox testing

White Box testing technique does need the tester to have in depth coding knowledge of the application. So generally developers act as Testers here. Unit Testing is an Example of White Box Testing which is done by developers

Let's discuss little bit about the various White Box Testing Techniques

- **Statement Testing & Coverage**

  This technique involves execution of all the executable statements in the source code at least once. The percentage of the executable statements is calculated as per the given requirement. This is the least preferred metric for checking test coverage.

- **Decision Testing Coverage**

  This technique is also known as branch coverage is a testing method in which each one of the possible branches from each decision point is executed at least once to ensure all reachable code is executed. This helps to validate all the branches in the code. This helps to ensure that no branch leads to unexpected behavior of the application.

- **Condition Testing**

  Condition testing also is known as Predicate coverage testing, each Boolean expression is predicted as TRUE or FALSE. All the testing outcomes are at least tested once. This type of testing involves 100% coverage of the code. The test cases are designed as such that the condition outcomes are easily executed.

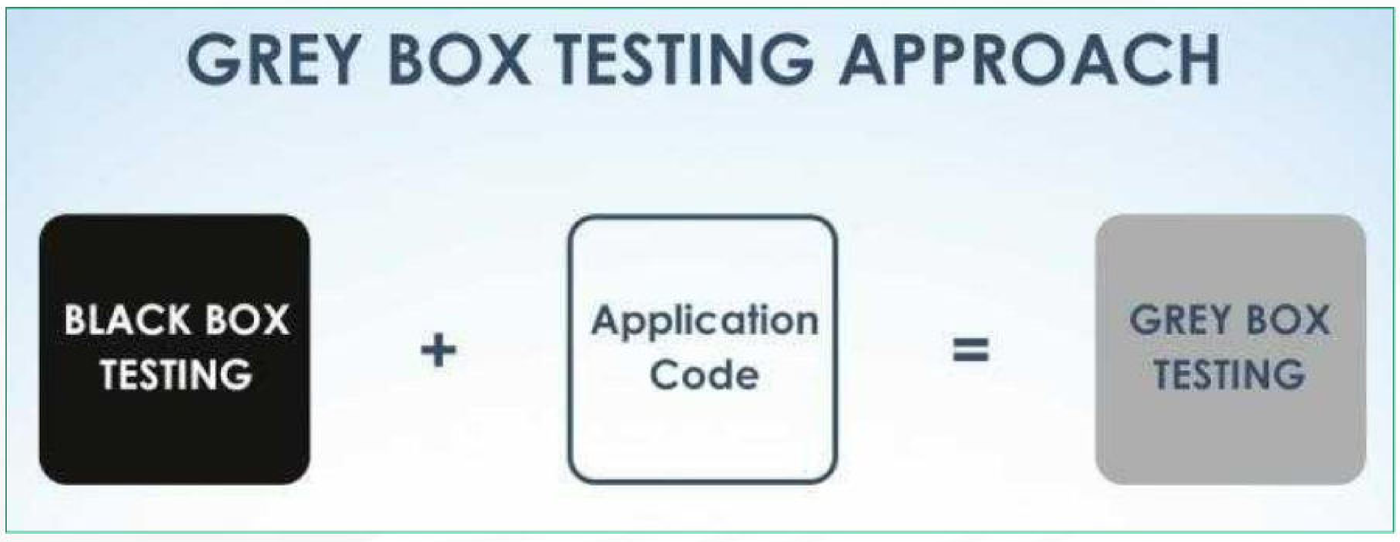
- **Multiple Condition Testing**

  The purpose of Multiple condition testing is to test the different combination of conditions to get 100% coverage. To ensure complete coverage, two or more test scripts are required which requires more efforts.

- **All Path Testing**

  In this technique, the source code of a program is leveraged to find every executable path. This helps to determine all the faults within a particular code.

GreyBox Testing
Unlike blackbox testing where the knowledge of business logic is not required and also complete knowledge in case of whitebox testing, greybox testing stands somewhere in the middle of both the approaches.



Facilitator Notes:

Tell the participants that greybox testing needs certain knowledge about the business logic of the application if not full knowledge.

It is also done by Developers and experienced testers. It increases testing coverage by testing the layers of a system which is complex in nature. The most advantageous aspect of greybox testing can sometimes become quite a mess.

Anything with testing has a very basic aspect and that aspect in 'input'. In the case of greybox testing, the developers will have their inputs and the testers will have their inputs and sometimes there is a clash as developers try to emphasize as their inputs being more accurate than testers.

But there are quite few advantages too. It reduces the overhead of long process of testing functional and non-functional types. It gives enough free time for a developer to fix defects.

Some of the techniques which are used to perform Greybox testing are:-

- **Matrix Testing:** This testing technique involves defining all the variables that exist in their programs.
- **Regression Testing:** Testing to ensure the older version is working fine when the newer version has been introduced in the software.
- **Orthogonal Array Testing or OAT:** 100% code coverage is not possible and anyways developers tend to keep too busy to even ensure it happens or not. So in such scenarios developers write certain test cases which ensure maximum code coverage in a very smart way. This is OAT.