Name :- jayesh patil

Department :- software testing

Assignment topic:- module- 2 (manual testing)

Submitted To:- tanvi meer

(1) What is Software Testing?

Software testing is standard testing can be defined as a process of analysing a software item to defect the differences between existing and required conditions and to evaluate the features of software item.

The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability. It mainly aims at measuring the specification, functionality, and performance of a software program or application.

(2) What is Exploratory Testing?

Exploratory testing is type of software testing in which the tester is free to select any possible methodology to test the software. It is an unscripted approach for software testing. In exploratory testing, software developers use their personal learning, knowledge, skills, and abilities to test the software developed by themselves. Exploratory testing checks the functionality and operations of the software as well as it identifies the functional and technical faults in it. The aim of exploratory testing is to optimize and improve the software in every possible way. Exploratory testing technique combines the experience of tester along with a

structured approach for testing. It is often performed as a black box technique.

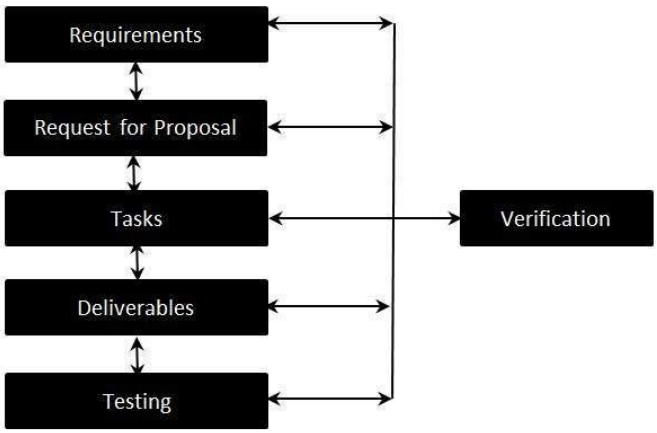
(3) What is Traceability Matrix?

Requirements tracing, a process of documenting the links between the requirements and the work products developed to implement and verify those requirements. The RTM captures all requirements and their traceability in a single document delivered at the conclusion of the life cycle.

RTM - Workflow:

The Matrix is created at the very beginning of a project as it forms the basis of the project's scope and deliverables that will be produced.

The Matrix is bi-directional, as it tracks the requirement forward by examining the output of the deliverables and backward by looking at the business requirement that was specified for a particular feature of the product.



Requirement traceability Matrix - Parameters:

- Requirement ID
- Risks
- Requirement Type
- Requirement Description
- Trace to Design Specification
- Unit Test Cases
- Integration Test Cases
- System Test Cases
- User Acceptance Test Cases
- Trace to Test Script

(4) What is Boundary Value Testing?

Boundary value analysis is a black box software testing technique where test

cases are designed using boundary values. BVA is based on the single fault assumption, also known as critical fault assumption which states that failures are rarely the product of two or more simultaneous faults. Hence while designing the test cases for BVA we keep all but one variable to take the extreme value.
Test case design for BVA:
while designing the test cases for BVA first we determine the number of inputs variables in the problem. For each input variable, we then determine the range of values it can take. Then we determine the extreme values and nominal value for each input variable.

(5) What is Equivalence partitioning testing?

Equivalence partitioning is also known as equivalence class partitioning (ECP). It is a software testing technique or black box testing that divides input domain into classes of data, and with the help of these classes of data, test cases can be derived. An ideal test case identifies class of error that might require many arbitrary test cases to be executed before general error is observed.
In equivalence partitioning, equivalence classes are evaluated for given input conditions. Whenever any input is given, then type of input condition is checked, then for this input conditions, equivalence class represents or describes set of valid or invalid states.

(6) What is Integration testing?

Upon completion of unit testing, the units or modules are to be integrated which gives raise to integration testing. The purpose of integration testing is to verify the functional, performance, and reliability between the modules that are integrated.

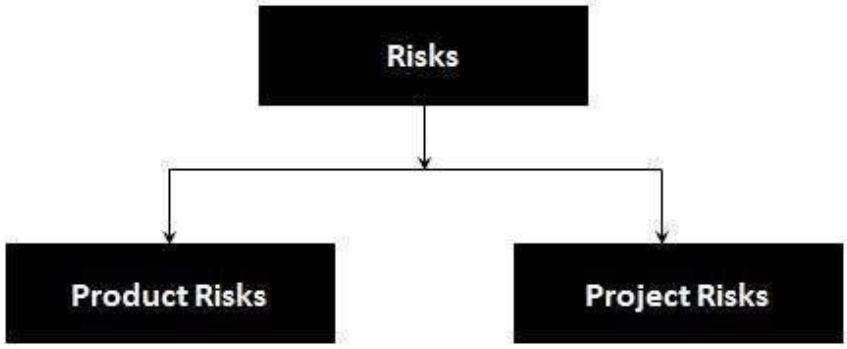
Integration Strategies:

- Big-Bang Integration
- Top Down Integration
- Bottom Up Integration
- Hybrid Integration

(7) What determines the level of risk?

Risk can be defined as the probability of an event, hazard, accident, threat or situation occurring and its undesirable consequences. It is a factor that could result in negative consequences and usually expressed as the product of impact and likelihood.

In software terminology, the risk is broadly divided into two main categories:



Project Risks:

- Supplier issues
- Organizational factors
- Technical issues

Product Risks:

Below are some of the product risks occurring in a LIVE environment?

- Defect Prone Software delivered
- The Critical defects in the product that could cause harm to an individual (injury or death) or company
- Poor software Features
- Inconsistent Software Features

(8) What is Alpha testing?

Alpha testing is a type of software testing performed to identify bugs before releasing the product to real users or to the public. Alpha testing is one of the user acceptance testing. This is referred to as alpha testing only because it is done early on, near the end of the development of the software. Alpha testing is commonly performed by homestead software engineers or quality assurance staff. It is the last testing stage before the software is released into the real world.

(9) What is Beta testing?

Beta testing is performed by real users of the software application in a real environment. Beta testing is one of the types of user acceptance testing. A beta version of the software, whose feedback is needed, is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing helps in minimization of product failure risks and it provides increased quality of the product through customer validation. It is the last test

before shipping a product to the customers. One of the major advantages of beta testing is direct feedback from customers.

(10) What is Component testing?

Also known as unit testing, module testing or program testing. A unit is the smallest testable part of software. Unit testing is a level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended with a debugging tool. Unit testing is a method by which individual units of source code are tested to determine if they are fit for use. A unit is the smallest testable part of an application like function/procedures, classes, interfaces. The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. Unit testing is performed by using the white box testing method. Quick and informal defect fixing.

(11) What is Functional system testing?

Functional testing is a type of software testing in which the system is tested against the functional requirements and specification. Functional testing ensures that the requirements or specification are properly satisfied by the application. This type of testing is particularly concerned with the result of processing. It focuses on simulation of actual system usage but does not develop any system structure assumptions. It is basically defined as a type of testing which verifies that each function of the software application works in conformance with the requirement and specification. The testing is not concerned about the source code of the application. Each functionality of the software application is tested by providing appropriate test input, expecting the output and comparing the actual output with the expected output.

(12) What is Non-functional testing?

Non-functional testing is a type of software testing that is performed to verify the non-functional requirements of the application. It verifies whether the behaviour of the system is as per the requirement or not. It tests all the aspects which are not tested in functional testing. Non-functional testing is defined as a type of software testing to check non-functional aspects of a software application. It is designed to test the readiness of a system as per non-functional parameters which are never addressed by functional testing. Non-functional testing is as important as functional testing.

(13) What is GUI testing?

Graphical User Interface Testing (GUI) is the process for ensuring proper functionality of the graphical user interface (GUI) for a specific application.

GUI testing generally evaluates a design of elements such as layout, colour and also fonts, font size, labels, textboxes, text formatting, captions, buttons, lists, icons, links, and content. GUI testing processes may be either manual or automatic and are often performed by third-party companies, rather than developers or and users.

(14) What is Adhoc testing?

Adhoc testing is a type of software testing which is performed informally and randomly after than formal testing is completed to find out any loophole in the system. For this reason, it is also known as random testing or monkey testing. Adhoc testing is not performed in an structured way so it is not based on any methodological approach. That's why adhoc testing is a type of unstructured software testing.

(15) What is white box testing and list the types of white box testing?

Testing based on an analysis of the internal structure of the component or system. Structure-based testing technique is also known as white box or glass box testing technique because here the testers require knowledge of how the software is implemented, how it works. Different test cases may be derived to exercise the loop once, twice, and many times. This may be done regardless of the functionality of the software. White box testing is the detailed investigation of internal logic and structure of the code. White box testing is also called glass testing or open box testing, in order to perform white box testing on an application, the tester needs to possess knowledge of the internal working of the code. The testers needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

Test/code coverage

- Test coverage measures the amount of testing performed by a set of test. Wherever we can count things and can tell whether or not each of those things has been tested by some test, then we can measure coverage and this is known as test coverage.
- The basic coverage measure is where the 'coverage item' is whatever we have been able to count and see whether a test has exercised or used this item.

$$\text{Coverage} = \frac{\text{Number of coverage item exercised}}{\text{Total number of coverage items}} \times 100\%$$

- There is danger in using a coverage measure. But, 100% coverage does not mean 100% tested. Coverage techniques measure only one dimension of a multi – dimensional concepts.

Types of coverage

- Statement/segment coverage:
  - The statement coverage is also known as line coverage or segment coverage.
  - The statement coverage covers only the true conditions.
  - Through statement coverage we can identify the statements executed and where the code is not executed because of blockage.
  - Aim is to display that all executable statements have been run at least once
  - The statement coverage can be calculated as shown below:

$$\text{Statement coverage} = \frac{\text{Number of statement exercised}}{\text{Total number of statements}} \times 100\%$$

- Decision/Branch coverage:
  - Decision coverage also known as branch coverage or all – edges coverage.
  - It covers both the true and false conditions unlikely the statement coverage.
  - Aim is to demonstrate that all decisions have been run at least once
  - With an if statement, the exit can either be true or false, depending on the value of the logical condition that comes after if.
  - The decision coverage can be calculated as shown below:

$$\text{Decision coverage} = \frac{\text{Number of decision outcomes exercised}}{\text{Total number of decision outcomes}} \times 100\%$$

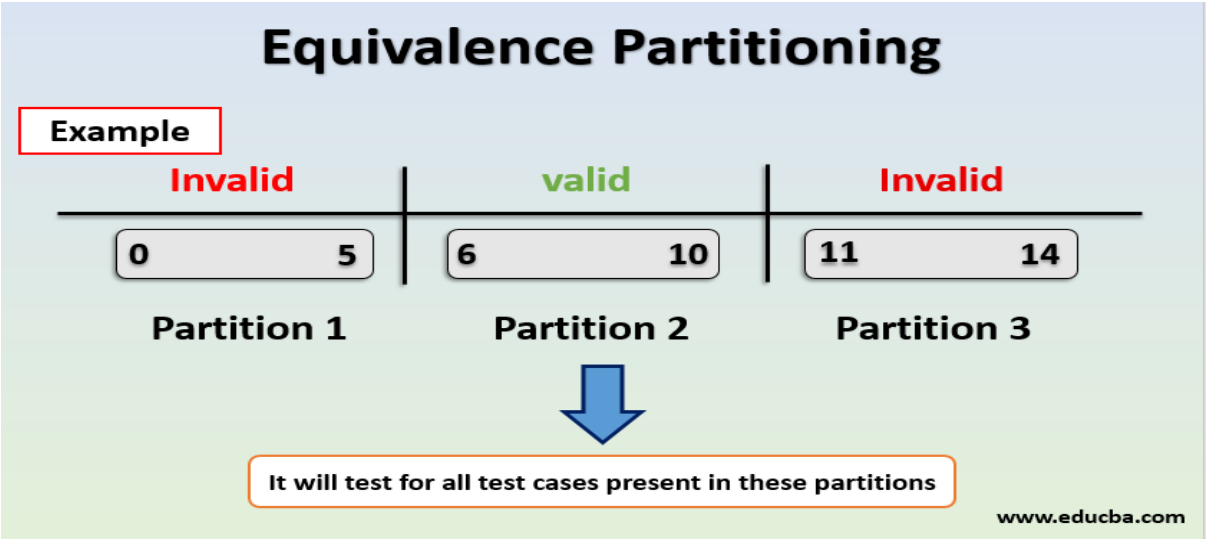
(16) What is black box testing? What are the different black box testing techniques?

Black box testing is a type of software testing in which the functionality of the software is not known. The testing is done without the internal knowledge of the products.

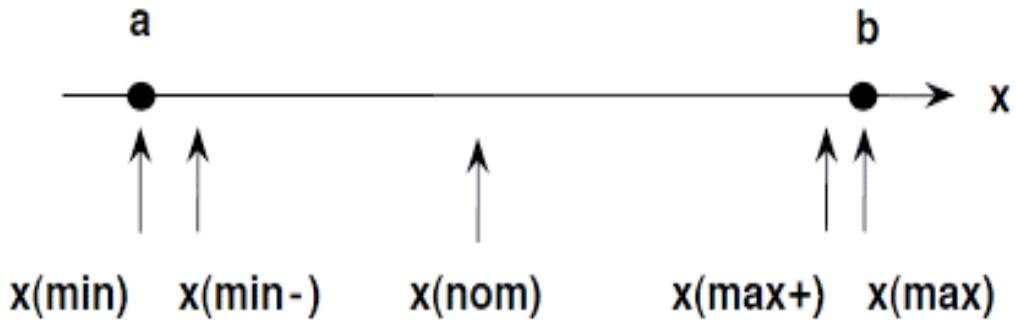
Black box testing techniques:

1. Equivalence partitioning: - Equivalence partitioning is also known as equivalence class partitioning (ECP). It is a software testing technique or black box testing that divides input domain into classes of data, and with the help of these classes of data, test cases can be derived. An ideal test case identifies class of error that might require many arbitrary test cases to be executed before general error is observed.
   In equivalence partitioning, equivalence classes are evaluated for given input conditions. Whenever any input is given, then type of input condition is

checked, then for this input conditions, equivalence class represents or describes set of valid or invalid states.



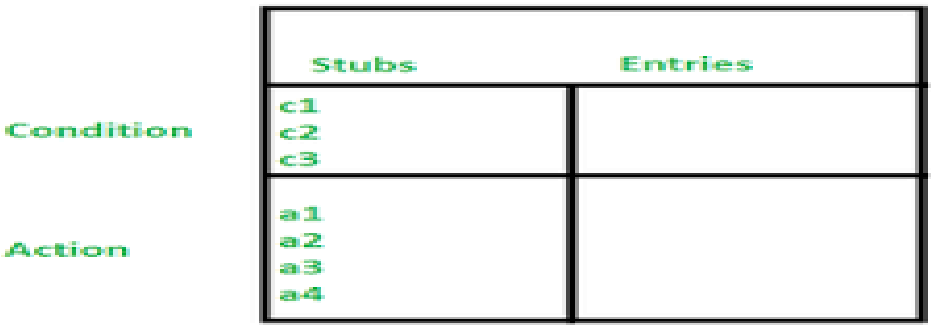
2. Boundary value analysis:- Boundary value analysis is a black box software testing technique where test cases are designed using boundary values. BVA is based on the single fault assumption, also known as critical fault assumption which states that failures are rarely the product of two or more simultaneous faults. Hence while designing the test cases for BVA we keep all but one variable to take the extreme value.
   Test case design for BVA:
   while designing the test cases for BVA first we determine the number of inputs variables in the problem. For each input variable, we then determine the range of values it can take. Then we determine the extreme values and nominal value for each input variable.



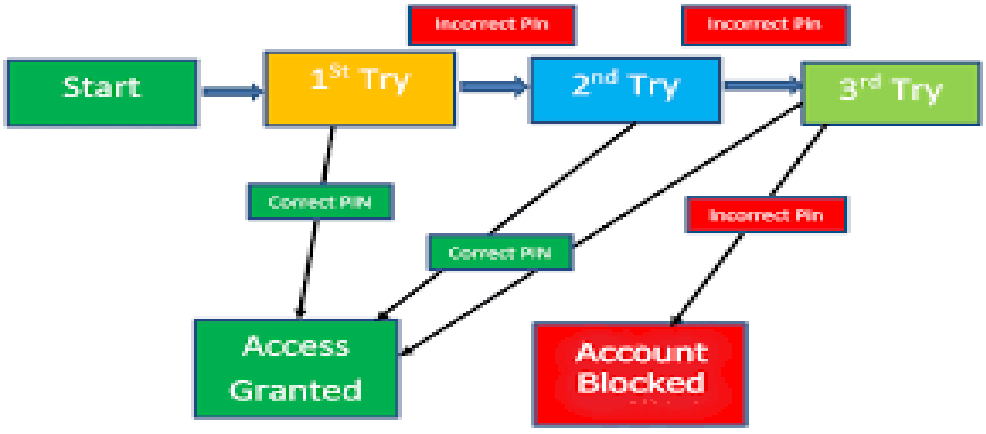
3. Decision table: - Decision table are used in various engineering fields to represent complex logical relationship. This testing is a very effective tool in testing the software and its requirements management. The output may be dependent on many input condition and decision tables give a tabular view of various combinations of input conditions and these conditions are in the form of True (T) and False (F). Also, it provides a set of conditions and its corresponding actions required in the testing.

Parts of decision table: In software testing the decision table has 4 parts which are divided into portions and are given below:

1. Condition stubs: - The conditions are listed in this first upper left part of the decision table that is used to determine a particular action or set of actions.
2. Action stubs: - All the possible actions are given in the first lower left portion (i.e., below condition stub) of the decision table.
3. Condition entries: - In the condition entry, the values are inputted in the upper right portion of the decision table. In the condition entries part of the table, there are multiple rows and columns which are known as rule.
4. Action entries: - In the action entry, every entry has some associated action or set of actions in the lower right portion of the decision table and these values are called outputs.



4. State transition testing: - State transition testing is a type of software testing which is performed to check the change in the state of the application under varying input. The condition of input passed is changed and the change in state is observed. State transition testing is basically a black box testing techniques that is carried out to observe the behaviour of the system or application for different input condition passed in a sequence. In this type of testing, both positive and negative input values are provided and the behaviour of the system is observed. State transition testing is basically used where different system transitions are needed to be tested.

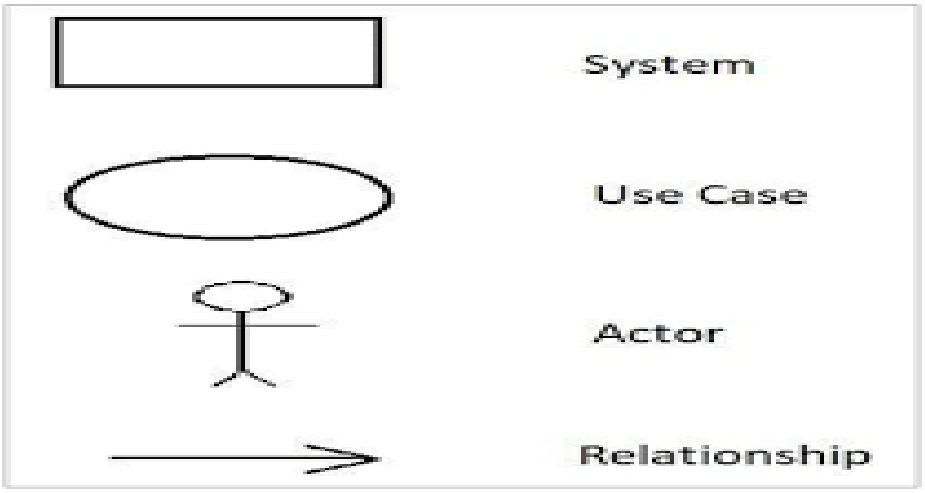


5. Use case testing: - A use case is as a tool for defining the required user interaction and if you are trying to create a new application ar make changes to an existing application, several discussions are made. Use case testing is

generally a part of a black box testing and that helps developers and testers to identify test scenarios that exercise that whole system on each transaction basis from start to finish. Business experts and developers must have a mutual understanding of the requirements, as it's very difficult to attain. Use case testing is a functional testing technique which helps in identifying and test scenario on the whole system or doing start to end transactions.

Feature of use case testing: - There is some feature of a use case testing, which is used to test the software project and provide s better response, these are given below:

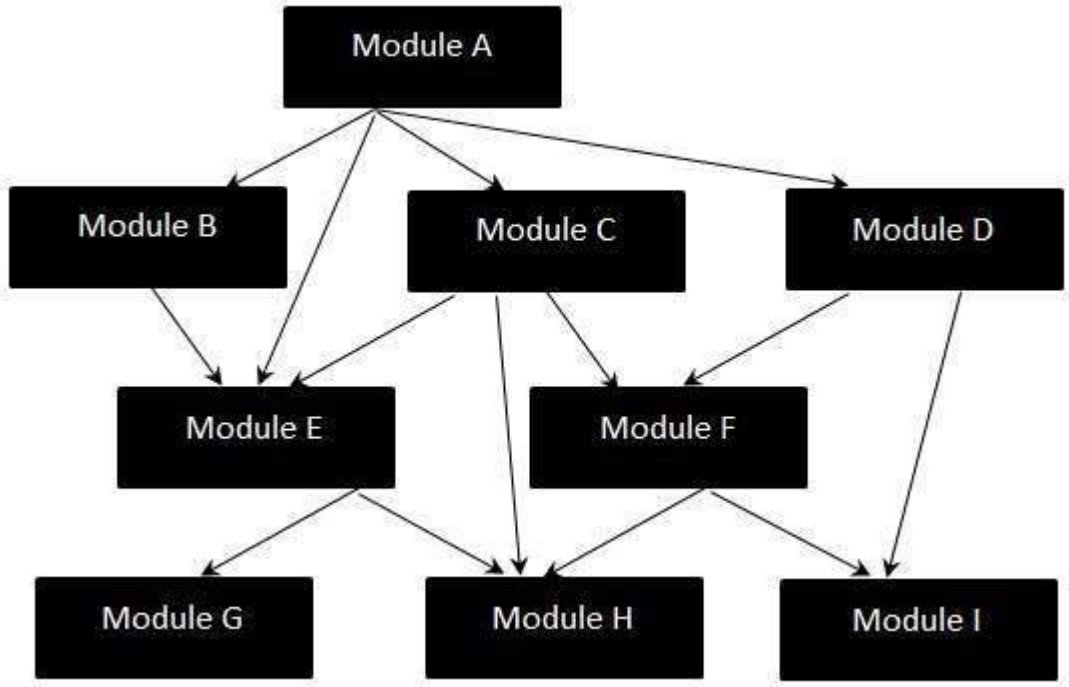
1. Use case testing is not testing that is performed to decide the quality of the software.
2. Although it is a type of end to end testing, it won't ensure the entire coverage of the user application.
3. Use case has generally captured the interactions between 'actors' and the 'system'.
4.'Actors' represents the user and their interactions that each user takes part in.
5. The use case will find out the defects in integration testing.



18. Mention what big bang testing is?

Big Bang Integration Testing is an integration testing strategy wherein all units are linked at once, resulting in a complete system. When this type of testing strategy is adopted, it is difficult to isolate any errors found, because attention is not paid to verifying the interfaces across individual units.

Big Bang Testing is represented by the following workflow diagram:



19. What is the purpose of exit criteria?

Exit criterion is used to determine whether a given test activity has been completed or NOT. Exit criteria can be defined for all of the test activities right from planning, specification and execution.

Exit criterion should be part of test plan and decided in the planning stage.

Examples of Exit Criteria:

- Verify if all tests planned have been run.
- Verify if the level of requirement coverage has been met.
- Verify if there are NO Critical or high severity defects that are left outstanding.
- Verify if all high risk areas are completely tested.
- Verify if software development activities are completed within the projected cost.
- Verify if software development activities are completed within the projected timelines.

20. When should “Regression testing” be performed?

- Testing of a previously tested program following modification to ensure that defects have not been introduced or uncovered in unchanged areas of the software, as a result of the changes made. It is performed when the software or its environment is changed.
- You also need to ensure that the modifications have not caused unintended side – effects elsewhere and that the modified system still meets its requirements.
- Regression testing should be carried out:
    1. When the system is stable and the system or the environment.
    2. Changes when testing bug – fix releases as part of the maintenance phase.
- It should be considered complete when agreed completion criteria for regression testing have been met.
- Regression test suites evolve over time and given that they are run frequently are ideal candidates for automation.

21. What is 7 key principles? Explain in detail?

Software testing is the process of executing a program with the aim of finding the error. To make our software perform well it should be error- free. If testing is done successfully it will remove all the errors from the software.

There are seven principles in software testing:

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

1. Testing shows the presence of defects: The goal of software testing is to make the software fail. Software testing reduces the presence of defects. Software testing talks about the presence of defects and doesn’t talk about the absence of defects. Software testing can ensure that defects are present but it cannot prove that software is defect-free. Even multiple testing can never ensure that software is 100% bug-free. Testing can reduce the number of defects but not remove all defects.

2. Exhaustive testing is not possible: It is process of testing the functionality of the software in all possible inputs (valid or invalid) and pre-condition is known

as exhaustive testing. Exhaustive testing is impossible means the software can never test at every test case. It can test only some test cases and assume that the software is correct and it will produce the correct output in every test case. If the software will test every test case then it will take more cost, effort, etc. which is impractical.

3. Early testing: To find the defect in the software, early test activity shall be started. The defect detected in the early phase of SDLC will be very less expensive. For better performance of software, software testing will start at the initial phase i.e. testing will perform at the requirement analysis phase.

4. Defects clustering: In a project, a small number of modules can contain most of the defects. Pareto principle to software testing state that 80% of software defect comes from 20% of modules.

5. Pesticide paradox: Repeating the same test cases, again and again, will not find new bugs. So it is necessary to review the test cases and add or update test cases to find new bugs.

6. Testing is context-dependent: The testing approach depends on the context of the software developed. Different types of software need to perform different types of testing. For example, The testing of the e-commerce site is different from the testing of the android application.

7. Absence of errors fallacy: If a built software is 99% bug-free but it does not follow the user requirement then it is unusable. It is not only necessary that software is 99% bug-free but it is also mandatory to fulfil all the customer requirements.

22. Difference between QA v/s QC v/s Tester?

Quality Assurance(QA)	Quality Control(QC)	Tester
• It focuses on providing assurance that the quality requested will be achieved.	• It focuses on fulfilling the quality requested.	• It is responsible for evaluating individual software.
• It is the technique of managing quality.	• It is the technique to verify quality.	• It simply evaluates functionality of software application.
• It is involved during the	• It is not included during the	• Software tester generally test

development phase.	development phase.	whether or not code runs as we expected it to run.
• It does not include the execution of the program.	• It always includes the execution the program.	• They are responsible for quality of software development and deployment.
• It is a managerial tool.	• It is a corrective tool.	• Its aim is to find bugs and errors in software application if present.
• It is process oriented.	• It is product oriented.	• They report problem as soon as possible, helps in saving money, provide security, and guarantee quality of software.
• The aim of quality assurance is to prevent defects.	• The aim of quality control is to identify and improve the defects.	• They not only find bugs, but also find its root cause so that it can be resolved permanently.
• It is a preventive technique.	• It is a corrective technique.	• Tester should have deep knowledge of system that is being developed, good communication skills, critical thinking, etc.
• It is a proactive measure.	• It is a reactive measure.	• They generally work with new software so that errors can be reduced or repaired if present.
• It is responsible for the entire software development life cycle.	• It is responsible for the software testing life cycle.	• They mainly focus on behaviour of end user while testing software application.

• It pays main focus is on the intermediate process.	• Its primary focus is on final products.	•
• All team members of the project are involved.	• Generally, the testing team of the project is involved.	•
• Example: Verification	• Example: Validation.	•

23. Difference between Smoke and Sanity?

Smoke Testing	**Sanity Testing**
• Smoke testing is done to assure that the acute functionalities of program is working fine.	• Sanity testing is done to check the bugs have been fixed after the build.
• Smoke testing is also called subset of acceptance testing.	• Sanity testing is also called subset of regression testing.
• Smoke testing is documented.	• Sanity testing isn't documented.
• Smoke testing is performed by either developers or testers.	• Sanity testing is normally performed by testers.
• Smoke testing may be stable or unstable.	• Sanity testing is stable.
• Smoke testing is scripted.	• Sanity testing is usually not scripted.
• Smoke testing is done to measures the stability of the system/product by performing testing.	• Sanity testing is done to measures the rationality of the system/product by performing testing.
• Smoke testing is used to test all over function of the system/product.	• Sanity testing is used in the case of only modified or defect function of system/products.
• Smoke testing can be performed either manually or by using automation tools.	• Sanity testing is commonly executed manually, not by using any automation approach.
• Smoke testing is performed when new product is built.	• Sanity testing is conducted after the completion of regression testing.

24. Difference between Verification and validation?

Verification	Validation
• It includes checking documents, design, codes and programs.	• It includes testing and validating the actual product.
• Verification is the static testing.	• Validation is the dynamic testing.
• It does not include the execution of the code.	• It includes the execution of the code.
• Methods used in verification are reviews, walkthroughs, inspections and desk-checking.	• Methods used in validation are black box testing, white box testing and non-functional testing
• It checks whether the software conforms to specification or not.	• It checks whether the software meets the requirements and expectation of a customer or not.
• It can find the bugs in the early stage of the development.	• It can only find the bugs that could not be found by the verification process.
• The goal of verification is application and software architecture and specification.	• The goal of validation is an actual product.
• Quality assurance team does verification.	• Validation is executed on software code with the help of testing team.
• It comes before validation.	• It comes after verification
• It consists of checking of documents/files and is performed by human.	• It consists of execution of program and is performed by computer.

25. Explain types of performance testing?

Performance Testing is a type of software testing that ensures software applications to perform properly under their expected workload. It is a testing technique carried out to determine system performance in terms of sensitivity, reactivity and stability under a particular workload.
Performance Testing is the process of analysing the quality and capability of a product. It is a testing method performed to determine the system performance in terms of speed, reliability and stability under varying workload. Performance testing is also known as **Perf Testing**.

Performance Testing Attributes:

- **Speed:**
  It determines whether the software product responds rapidly.

- **Scalability:**
  It determines amount of load the software product can handle at a time.
- **Stability:**
  It determines whether the software product is stable in case of varying workloads.
- **Reliability:**
  It determines whether the software product is secure or not.

Objective of Performance Testing:

1. The objective of performance testing is to eliminate performance congestion.
2. It uncovers what is needed to be improved before the product is launched in market.
3. The objective of performance testing is to make software rapid.
4. The objective of performance testing is to make software stable and reliable.

Types of Performance Testing:

1. **Load testing:**
   It checks the product's ability to perform under anticipated user loads. The objective is to identify performance congestion before the software product is launched in market.
2. **Stress testing:**
   It involves testing a product under extreme workloads to see whether it handles high traffic or not. The objective is to identify the breaking point of a software product.
3. **Endurance testing:**
   It is performed to ensure the software can handle the expected load over a long period of time.
4. **Spike testing:**
   It tests the product's reaction to sudden large spikes in the load generated by users.
5. **Volume testing:**
   In volume testing large number of data is saved in a database and the overall software system's behaviour is observed. The objective is to check product's performance under varying database volumes.
6. **Scalability testing:**
   In scalability testing, software application's effectiveness is determined in scaling up to support an increase in user load. It helps in planning capacity addition to your software system.

26. What is Error, Defects, Bug, and failure?

1. Error:

A mistake in coding is called an error. A discrepancy between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition. This can be a misunderstanding of the internal state of the software, an oversight in terms of memory management, confusion about the proper way to calculate a value, etc.

2. Defects:

The bugs introduced by programmer inside the code is called is called as defects. Defect is defined as the deviation from the actual and expected result or application or software or in other words, defects, are defined as any deviation or irregularity from the specification document. Defect is also solved by the developer in development phase or stage. Reasons for defects:

- Any deviation from the customer requirements is called as defect.
- By giving wrong input may lead to defect.
- Any error in logic code may lead to defect.

3. Bugs:

Sometimes most people are confused between defect and bug, they say that bug is the informal name of defect. Actually bugs are faults in system or application which impact on software functionality and performance. Usually bugs are found in unit testing by tester. There are different types of bugs, some of them are given below.

- Functional errors
- Compilation errors
- Missing commands
- Run time errors
- Logical errors
- Inappropriate error handling

Above given these errors lead to bug.

4. Failure:

When a defect reaches the end customer, it is called as failure. Once the product is completed and it is delivered to the customers and if the customer fined any issues in product or software then it is the condition of failure of product. In other words, if an end user finds an issue I product then that particular issue is called as failure. Causes of failure:

- Human errors or mistakes may lead to failure.
- Environmental conditions
- The way in which system is used.

27. Difference between priority and severity?

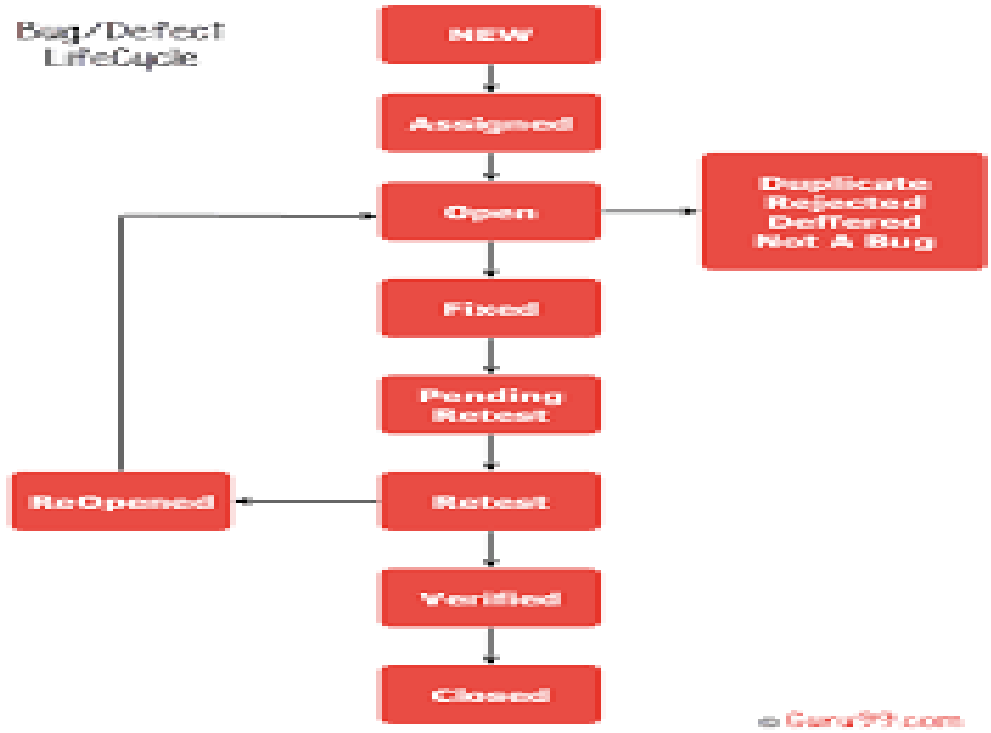
Priority	**Severity**
• Priority is a parameter to decide the order in which defects should be fixed.	• Severity is a parameter to denote the impact of a particular defect on the software.

• Priority means how fast defect has to be fixed.	• Severity means how severe defect is affecting the functionality.
• Priority is related to scheduling to resolve the problem.	• Severity is related to the quality standard.
• Product manager decides the Priorities of defects.	• Testing engineer decides the severity level of the defect.
• Its value is subjective.	• Its value is objective
• Its value changes from time to time.	• Its value doesn't change from time to time
• Priority is of 3 types: Low, Medium, and High.	• Severity is of 5 types: Critical, Major, Moderate, Minor, and Cosmetic.

28. What is Bug life cycle?

Workflow of Defect/Bug Life Cycle

The below diagram illustrates actual workflow of Defect Life Cycle.



The above diagram shows different states of Defect in Defect Life Cycle and these are as follows:

1. NEW -

When any new defect is identified by tester, it falls in 'New' state. It is first state of Bug Life Cycle. The tester provides a proper Defect document to Development team so that development team can refer to Defect Document and can fix bug accordingly.

2. ASSIGNED -

Defects which are in status of 'New' will be approved and that newly identified defect is assigned to the development team for working on defect and to resolve that. When the defect is assigned to developer team then status of bug changes to 'Assigned' state.

3. OPEN -

In this 'Open' state the defect is being addressed by developer team and developer team works on the defect for fixing the bug. Based on some specific reason if developer team feels that defect is not appropriate then it is transferred to either 'Rejected' or 'Deferred' state.

4. FIXED -

After necessary changes of codes or after fixing identified bug developer team marks state as 'Fixed'.

5. PENDING RETEST –

During the fixing of defect is completed, developer team passes new code to testing team for retest. And the code/application is pending for retesting at Tester side so status is assigned as 'Pending Retest'.

6. RETEST -

At this stage, tester starts work of retesting defect to check whether defect is fixed by developer or not, and the status is marked as 'Retesting'.

7. REOPEN -

After 'Retesting' if tester team found that bug continues like previously even after developer team has fixed the bug, then status of bug is again changed to 'Reopened'. Once again bug goes to 'Open' state and goes through life cycle again. This means it goes for Re-fixing by the developer team.

8. VERIFIED -

The tester re-tests bug after it got fixed by developer team and if tester does not find any kind of defect/bug then bug is fixed and status assigned is 'Verified'.

9. CLOSED -

It is the final state of Defect Cycle, after fixing defect by developer team when testing found that the bug has been resolved and it does not persist then they mark defect as a 'Closed' state.

Few More States that also comes under this Defect Life Cycle -

1. REJECTED -

If the developer team rejects defect if they feel that defect is not considered as a genuine defect, and then they mark status as ‘Rejected’. The cause of rejection may be any of these three i.e Duplicate Defect, NOT a Defect, Non-Reproducible.

2. DEFERRED -

All defects have their bad impact on developed software and also they have a level based on his impact on software. If the developer team feels that defect that is identified is not a prime priority and it can get fixed in further updates or releases then developer team can mark status as 'Deferred'. Means from current defect life cycle it will be terminated.

3. DUPLICATE -

Sometimes it may happen that defect is repeated twice or defect is same as any other defect then it is marked as 'Duplicate' state and then defect is 'Rejected'.

4. NOT A DEFECT -

If the defect has no impact or effect on other functions of the software then it is marked as 'NOT A DEFECT' state and 'Rejected'.

5. NON-REPRODUCIBLE -

If the defect is not reproduced due to platform mismatch, data mismatch, build mismatch, or any other reason than developer marks defect as in a 'Non-Reproducible' state.

6. CAN'T BE FIXED -

If the developer team fails to fix defect due to Technology support, Cost of fixing bug is more, lack of required skill or due to any other reasons then developer team marks defect as in 'Can't be fixed' state.

7. NEED MORE INFORMATION -

This state is very close to 'Non-reproducible' state. But it is different from that. When the developer team fails to reproduce defect due to steps/document provided by tester is insufficient or Defect Document is not so clear to reproduce defect then developer team can change status as “need more information’. When the Tester team provides a good defect document then developer team proceeds to fix the bug

29. Explain the difference between functional testing and non-functional testing?

Functional testing	Non-Functional testing
• It verifies the operations and actions of an application.	• It verifies the behaviour of an application.
• It is based on requirements of customer.	• It is based on expectations of customer.
• It helps to enhance the behaviour of the application.	• It helps to improve the performance of the application.
• Functional testing is easy to execute manually.	• It is hard to execute non-functional testing manually.
• It tests what the product does.	• It describes how the product does.
• Functional testing is based on the business requirement.	• Non-Functional testing is based on the performance requirement.
• Example: 1.Unit testing 2.Smoke testing 3.Integration testing 4.Regression testing	• Example: 1.performance testing 2.Load testing 3.Stress testing 4.Scalability testing

(30) To create HLR & test case of: 1.Instagram, Facebook only for first page

(31) what is difference between the STLC (software testing life cycle) and SDLC (software development life cycle)?

SDLC	STLC
• SDLC is mainly related to software development.	• STLC is mainly related to software testing.
• Besides development other phases like testing is also included.	• It focuses only on testing the software.
• SDLC involves total six phases or steps.	• STLC involves only five phases or steps.
• In SDLC, more number of members (developers) are required for the whole process.	• In STLC, less number of members (tester) are needed.
• In SDLC, development team make the plans and designs based on the requirements.	• In STLC, testing team (test lead or test architect) makes the plans and designs.
• Goal of SDLC is to complete successful development of software.	• Goal of STLC is to complete successful testing of software.

• It helps in developing good quality software.	• It helps making the software defects free.
• SDLC phases are completed before the STLC phases.	• STLC phases are performed after SDLC phases.
• Post deployment support, enhancement, and update are to be included if necessary.	• Regression tests are run by QA team to check deployed maintenance code and maintains test cases and automated scripts.
• Creation of reusable software systems is the end result of SDLC.	• A tested software system is the end result of STLC.

(32) What is difference between test scenario, test cases, and test script?

Test scenario	**Test cases**	**Test script**
• The test scenario is just a document that is detailed and provides details about the assessment method, testing process, precondition, and anticipated output.	• Test cases is a step by step procedure to test any functionality of the software application/product.	• Test script is set of instruction or a short program to test any functionality of software application/product.
• The test scenarios are the ones based on the use situation and give one-line information one what to check.	• Test cases is a manual approach of software testing.	• Test script is an automatic approach of software testing.
• Test scenarios are one-liner statement, however, it is linked to a few test instances.	• It is a set up that is used by the tester to test any specific function of the software product.	• It is a program developed by the tester, intended to test any specific function of the software product.
• These are high-level actions.	• Point by point test case configuration encourages tester to test viably.	• Automatic testing approach is beneficial for constant execution.

• Writing the test scenario's primary objective is an address end to get rid of functionality of a software program.	• Test cases are written by manually.	• Test scripting is done by scripting format.
• It will take less time as compared to test cases.	• Test case is developed in form of templates.	• Test script is developed in form of scripting.
• The test scenario will help us in a way that is nimble of through the functionality.	• If the tester does not have a good understanding of how the program is used or about the recent risks to the program, then it will be difficult to use the test cases properly.	• Active software projects frequently change.so testers have to make a continuous effort to update the scripts to match the changes of the new product.
• Test scenario are really easy to maintain due to their high-level design.	• Test case is used in manual testing environment.	• Test script is used in automatic testing environment.
• The test scenarios tend to be work on the essential to "things to be tested".	• Test cases are classified as delegated, positive, reusable, negative and UI test cases.	• Test script are characterized as manual test script and automatic test scripts.
• A lot fewer sources are sufficient to publish test circumstances in comparison with the test instances.	• Test case comprises of many test qualities like condition, test data, environment, test suite id, name and others.	• In test script different commands can be used to develop scripts which is used as a part of performing repeatable and regression testing.
• Requires fewer resources and less time.	• Requires more resources and time.	• Requires less time for testing scripts.

(33) Explain what test plan is? What is the information that should be covered?

- A test plan is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverable, and resources required to perform testing for a software product.
- As per ISTQB definition: “test plan is a document describing the scope, approach, resources, and schedule of intended test activities.”
- The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager.
- A test plan will include the following.
    1) Introduction to the test plan document
    2) Assumptions when testing application
    3) List of test cases included in testing the application
    4) List of features to be tested
    5) What sort of approach to use when testing the software
    6) List of deliverables that need to be tested
    7) They resources allocated for testing the application
    8) Any risks involved during the testing process
    9) A schedule of tasks and milestones as testing is stared
    - Master test plan: A test plan that typically addresses multiple test levels.
    - Phase test plan: A test plan that typically addresses one test phase.
    - Specific test plan: In this type of test plan, it is designed for specific types of testing especially non – functional testing.

(34) What are the different methodologies in agile development model?

- There are various methodologies present in agile testing and those are listed below:
    1) Scrum
    2) eXtreme programming
- Below listed methodologies are used less frequently
- **Dynamic system development method (DSDM):** This is an iterative and incremental approach that emphasizes on the continuous user involvement.
- **Total driven development (TDD):** This is a technique which has short iterations where new test cases covering the desired improvement or new functionality are written first.
- **Feature driven development:** This is an iterative and incremental software development process and this can aim depends on the features.
- **XBreed:** Agile enterprise previously known as Xbreed. It is agile way of managing, architecting and monitoring the enterprise
- **Crystal:** Crystal is an adaptive techniques mainly used for software development methodologies.

(35) Explain the difference between authorization and authentication in web testing?

Authentication	Authorization
• In the authentication process, the identity of users are checked for providing the access to the system.	• While in authorization process, a the person's or user's authorities are checked for accessing the resources.
• In the authentication process, users or persons are verified.	• While in this process, users or persons are validated.
• It is done before the authorization process.	• While this process is done after the authentication process.
• It needs usually the user's login details.	• While it needs the user's privilege or security levels.
• Authentication determines whether the person is user or not.	• While it determines **What permission does the user have?**
• **Example**: Employees in a company are required to authenticate through the network before accessing their company email.	• **Example:** After an employee successfully authenticates, the system determines what information the employees are allowed to access.
• The user authentication is visible at user end.	• The user authorization is not visible at the user end.
• The user authentication is identified with username, password, face recognition, retina scan, fingerprints, etc.	• The user authorization is carried out through the access rights to resources by using roles that have been pre-defined.
• The authentication credentials can be changed in part as and when required by the user.	• The authorization permissions cannot be changed by user as these are granted by the owner of the system and only he/she has the access to change it.

(36) What are the common problems faced in web testing?

1) Functionality testing
2) Usability testing
3) Security testing
4) Compatibility testing
5) Interface testing
6) Performance testing

- Functionality testing:
  - Links: Objectives is to check for all the links in the website.
  - Forms: Test for the integrity of submission of all forms.
  - Cookies: Check for the cookies that has to be enabled and how it has to be expired.
  - Web indexing: Depending on how the site is designed using meta tags, frames, HTML, syntax, dynamically created pages, passwords or different languages, our site will be searchable in different ways.
  - Database: Data integrity –missing or wrong data in table. output error – errors in writing, editing or reading operations in the tables.
- Usability:

- Navigation: Application navigation is proper through tab, mouse, and main features accessible from the main/home page. Any hotkeys, control keys to access menus.
- Content: Spellings and grammars and updated information
- General appearance: Page appearance, colour, font, and size, frames, consistent design

➢ Interfaces:
- Server interface: Verify that communication is done correctly web server – application server, application server – database server and vice versa.
- External interface: If any

➢ Security:
- Valid and invalid login.
- Limit defined for the number of tries.
- Verify encryption is done correctly if SSL is used.
- No access to edit scripts on the server without authorization.
- Verify log files are maintained to store the information for tracebility.

➢ Client side Compatibility:
- Platform: Check for the compatibility of various operating system.
- Printing: Verify that pages are printable with considerations on: text and image alignment, colour of text, foreground and background, scalability to fit paper size, tables and borders.
- Browsers: Browser setting, frames and cascade style sheets, applets, activex controls, DHTML, client side scripting, HTML, specifications, graphics: loading of image, graphics, etc.

➢ Performance:
- Connection speed: Try with connection speed: 14.4, 28.8, 33.6, 56.6, ISDN, cable, DSL, T1, T3
- Load: Large amount of data from users, can your site handle a large amount of users requesting a certain page?
- Stress: Stress tests are designed to push and test system limitations and determines whether the system recovers gracefully from crashes.
- Continuous use: Verify that the application is able to meet the requirements and does not run out of memory or disk space.

(37) To create HLR & test case of web based (whatsapp web, instagram)?

(38) To create HLR & test case on this link: https://artoftesting .com/?