Software Testing Assignment

Module – 2 (Manual Testing)

1) What is Exploratory Testing?

- Exploratory testing is an approach to software testing that is often described as simultaneous learning, test design, and execution.

- It focuses on discovery and relies on the guidance of the individual tester to uncover defects that are not easily covered in the scope of other tests.

- Exploratory testing allows you to think outside the box and come up with use cases that might not be covered in a test case. For example, you might perform one test and then ask yourself, “What if I tried this? What if I didn't do that?”

- Exploratory Testing is a style of software testing where there is less of a structure and a specified process. In this testing, the tester has more personal freedom and responsibility to utilize their skills and knowledge to optimize the quality of their work. The tester designs and executes the test simultaneously.

2) What is traceability matrix?

- A traceability matrix is a document that details the technical requirements for a given test scenario and its current state.

- It helps the testing team understand the level of testing that is done for a given product.

- The traceability process itself is used to review the test cases that were defined for any requirement.

- A traceability matrix in software testing — otherwise known as a test matrix — is used to prove that tests have been run. It documents test cases, test runs, and test results. Requirements and issues may also be used in a test matrix.

3) What is Boundary value testing?

- Boundary-value analysis is a software testing technique in which tests are designed to include representatives of boundary values in a range. The idea comes from the boundary.

- It is easier and faster to find defects using this technique. This is because the density of defects at boundaries is more. Instead of testing will all set of test data, we only pick the one at the boundaries. So, the overall test execution time reduces.

4) What is Equivalence partitioning testing?

- Equivalence partitioning or equivalence class partitioning (ECP) is a software testing technique that divides the input data of a software unit into partitions of equivalent data from which test cases can be derived. In principle, test cases are designed to cover each partition at least once.

- It is a technique where the input data is divided into partitions of valid and invalid values.

- In equivalence-partitioning technique we need to test only one condition from each partition. If one condition in a partition works, we assume all of the conditions in that partition will work.

5) What is Integration testing?

- Integration testing -- also known as integration and testing (I&T) -- is a type of software testing in which the different units, modules or components of a software application are tested as a combined entity. However, these modules may be coded by different programmers.

- Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.

6) What determines the level of risk?

- In Software Testing, risk analysis is the process of identifying the risks in applications or software that you built and prioritizing them to test.

- After that, the process of assigning the level of risk is done. The categorization of the risks takes place, hence, the impact of the risk is calculated.

7) What is Alpha testing?

- Alpha testing is the initial phase of validating whether a new product will perform as expected. Alpha tests are carried out early in the development process by internal staff.
- Alpha Testing is done within the organization.

- During Alpha Testing only functionality and usability are tested

8) What is beta testing?

- In software development, a beta test is the second phase of software testing in which a sampling of the intended audience tries the product out.

- Beta is the second letter of the Greek alphabet. Originally, the term alpha test meant the first phase of testing in a software development process.

- Beta Testing is done in the user's environment.

- Beta testing is also sometimes referred to as user acceptance testing (UAT) or end user testing.

- In this phase of software development, applications are subjected to real world testing by the intended audience for the software.

- The experiences of the early users are forwarded back to the developers who make final changes before releasing the software commercially.

9) What is component testing?

- Component testing, also known as program or module testing, is done after unit testing.

- In this type of testing those test objects can be tested independently as a component without integrating with other components e.g. modules, classes, objects, and programs. This testing is done by the development team.

- The primary objective of component testing is to validate the behaviour of the individual component, as specified in the requirements document. The tester isolates an individual from other parts to prevent external influence and ensure that efficient results are generated.

10) What is functional system testing?

- Functional testing of a system includes tests that evaluate the functions that the system must perform.

- Functional requirements may be described in work products (requirements, specification, business need, user story, use case) and in the functional specification.

- Functional testing is a type of testing that seeks to establish whether each application feature works as per the software requirements. Each function is compared to the corresponding requirement to ascertain whether its output is consistent with the end user's expectations.

11) What is Non-Functional Testing?

- Non-functional testing assesses application properties that aren't critical to functionality but contribute to the end-user experience.

- Performance and reliability under load aren't functional components of a software system but can certainly make or break the user experience.

- Non-functional testing verifies the way software works — and how well it works.
- Non-functional testing checks things that aren’t covered in functional tests.

12) What is GUI Testing?

- GUI Testing is a software testing type that checks the Graphical User Interface of the Software.

- The purpose of Graphical User Interface (GUI) Testing is to ensure the functionalities of software application work as per specifications by checking screens and controls like menus, buttons, icons, etc.

- GUI is what the user sees. Say if you visit any website, what you will see say homepage it is the GUI (graphical user interface) of the site. A user does not see the source code. The interface is visible to the user. Especially the focus is on the design structure, images that they are working properly or not.

13) What is Ad hoc testing?

- Ad hoc Testing is an informal or unstructured software testing type that aims to break the testing process in order to find possible defects or errors at an early possible stage.

- Ad hoc testing is done randomly and it is usually an unplanned activity which does not follow any documentation and test design techniques to create test cases.

- Ad hoc Testing does not follow any structured way of testing and it is randomly done on any part of application.

- Main aim of this testing is to find defects by random checking. Ad hoc testing can be achieved with the Software testing technique called Error Guessing. Error guessing can be done by the people having enough experience on the system to “guess” the most likely source of errors.

- This testing requires no documentation/ planning /process to be followed.

- Since this testing aims at finding defects through random approach, without any documentation, defects will not be mapped to test cases. This means that, sometimes, it is very difficult to reproduce the defects as there are no test steps or requirements mapped to it.

14) What is Load testing?

- Load Testing is a non-functional software testing process in which the performance of software application is tested under a specific expected load. It determines how the software application behaves while being accessed by multiple users simultaneously.

- The goal of Load Testing is to improve performance bottlenecks and to ensure stability and smooth functioning of software application before deployment.

- This testing usually identifies :

    - The maximum operating capacity of an application
    - Determine whether the current infrastructure is sufficient to run the application
    - Sustainability of application with respect to peak user load
    - Number of concurrent users that an application can support, and scalability to allow more users to access it.
    - It is a type of non-functional testing. In Software Engineering, Load testing is commonly used for the Client/Server, Web-based applications – both Intranet and Internet.

15) What is stress Testing?

- Stress Testing is a type of software testing that verifies stability & reliability of software application.

- The goal of Stress testing is measuring software on its robustness and error handling capabilities under extremely heavy load conditions and ensuring that software doesn't crash under crunch situations.

- It even tests beyond normal operating points and evaluates how software works under extreme conditions.

- A most prominent use of stress testing is to determine the limit, at which the system or software or hardware breaks.

- It also checks whether the system demonstrates effective error management under extreme conditions.

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

- White Box Testing is a testing technique in which software's internal structure, design, and coding are tested to verify input-output flow and improve design, usability, and security.

- In white box testing, code is visible to testers, so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing, and Glass box testing.

- The testing can be done at system, integration, and unit levels of software development.

- One of the basic goals of white box testing is to verify a working flow for an application. It involves testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

- Types of White Box Testing

    1. Unit Testing
    2. Static Analysis
    3. Dynamic Analysis
    4. Statement Coverage
    5. Branch testing Coverage
    6. Security Testing
    7. Mutation Testing

17) What is black box testing? What are the different black box testing techniques?

- Black box testing involves testing a system with no prior knowledge of its internal workings.

- A tester provides an input, and observes the output generated by the system under test. This makes it possible to identify how the system responds to expected and unexpected user actions, its response time, usability issues and reliability issues.

- Black box testing is a powerful testing technique because it exercises a system end-to-end. Just like end-users "don't care" how a system is coded or architected, and expect to receive an appropriate response to their requests, a tester can simulate user activity and see if the system delivers on its promises.

- Along the way, a black box test evaluates all relevant subsystems, including UI/UX, web server or application server, database, dependencies, and integrated systems.

- Types of Black Box Testing

    1. Equivalence partitioning
    2. Boundary value analysis
    3. Decision tables
    4. State transition testing

18) Mention what are the categories of defects?

1. Design Defects
2. Command Defects
3. Boundary Value Defects
4. Error Handling Defects
5. Multithreading Defects
6. Security Defect
7. Interface Defects
8. Priority of Defects

19) Mention what Big Bang testing is?

- Big Bang Testing is an Integration testing approach in which all the components or modules are integrated together at once and then tested as a unit.

- This combined set of components is considered as an entity while testing. If all of the components in the unit are not completed, the integration process will not execute.

Advantages:

- Convenient for small systems.

Disadvantages:

- Fault Localization is difficult.
- Given the sheer number of interfaces that need to be tested in this approach, some interfaces link to be tested could be missed easily.
- Since the Integration testing can commence only after "all" the modules are designed, the testing team will have less time for execution in the testing phase.
- Since all modules are tested at once, high-risk critical modules are not isolated and tested on priority. Peripheral modules which deal with user interfaces are also not isolated and tested on priority.

20) What is the purpose of exit criteria?

- Exit Criteria defines the items that must be completed before testing can be concluded.

- 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.

- The purpose of exit criteria is to prevent a task from being considered completed when there are still outstanding parts of the task which have not been finished.

- Exit criteria are used to report against and to plan when to stop testing.

21) When should "Regression Testing" be performed?

- Regression testing is a black box testing techniques.

- It is used to authenticate a code change in the software does not impact the existing functionality of the product. Regression testing is making sure that the product works fine with new functionality, bug fixes, or any change in the existing feature.

- Regression testing is a type of software testing. Test cases are re-executed to check the previous functionality of the application is working fine, and the new changes have not produced any bugs.

- Regression testing can be performed on a new build when there is a significant change in the original functionality. It ensures that the code still works even when the changes are occurring. Regression means Re-test those parts of the application, which are unchanged.

- Regression tests are also known as the Verification Method. Test cases are often automated. Test cases are required to execute many times and running the same test case again and again manually, is time-consuming and tedious too.

22) What is 7 key principles? Explain in detail

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

1) Testing shows a presence of defects

- Hence, testing principle states that – Testing talks about the presence of defects and don’t talk about the absence of defects. i.e. Software Testing reduces the probability of undiscovered defects remaining in the software but even if no defects are found, it is not a proof of correctness.

- But what if, you work extra hard, taking all precautions & make your software product 99% bug-free. And the software does not meet the needs & requirements of the clients.
- This leads us to our next principle, which states that- Absence of Error

2) Exhaustive testing is not possible

- Testing everything including all combinations of inputs and preconditions is not possible.
  So, instead of doing the exhaustive testing we can use risks and priorities to focus testing efforts.
- For example:
  - In an application in one screen there are 15 input
  - Fields, each having 5 possible values, then to test all the valid combinations you would need 30 517 578 125 (515) tests.
- This is very unlikely that the project timescales would allow for this number of tests.
- So, accessing and managing risk is one of the most important activities and reason for testing in any project.
- We have learned that we cannot test everything (i.e. all combinations of inputs and pre-conditions).
- That is we must prioritise our testing effort using a Risk Based Approach.

3) Early Testing

- Early Testing – Testing should start as early as possible in the Software Development Life Cycle. So that any defects in the requirements or design phase are captured in early stages. It is much cheaper to fix a Defect in the early stages of testing. But how early one should start testing? It is recommended that you start finding the bug the moment the requirements are defined. More on this principle in a later training tutorial.

4) Defect Clustering

- Defect clustering which states that a small number of modules contain most of the defects detected. This is the application of the Pareto Principle to software testing: approximately 80% of the problems are found in 20% of the modules.
- By experience, you can identify such risky modules. But this approach has its own problems
- If the same tests are repeated over and over again, eventually the same test cases will no longer find new bugs.

5) Pesticide Paradox

- Repetitive use of the same pesticide mix to eradicate insects during farming will over time lead to the insects developing resistance to the pesticide Thereby ineffective of

pesticides on insects. The same applies to software testing. If the same set of repetitive tests are conducted, the method will be useless for discovering new defects.

- To overcome this, the test cases need to be regularly reviewed & revised, adding new & different test cases to help find more defects.
- Testers cannot simply depend on existing test techniques. He must look out continually to improve the existing methods to make testing more effective. But even after all this sweat & hard work in testing, you can never claim your product is bug-free. To drive home this point, let's see this video of the public launch of Windows 98
- You think a company like MICROSOFT would not have tested their OS thoroughly & would risk their reputation just to see their OS crashing during its public launch!

6) Testing is context dependent

- Testing is context dependent which basically means that the way you test an e-commerce site will be different from the way you test a commercial off the shelf application. All the developed software's are not identical. You might use a different approach, methodologies, techniques, and types of testing depending upon the application type. For instance testing, any POS system at a retail store will be different than testing an ATM machine.

7) Absence of Error – fallacy

- It is possible that software which is 99% bug-free is still unusable. This can be the case if the system is tested thoroughly for the wrong requirement. Software testing is not mere finding defects, but also to check that software addresses the business needs. The absence of Error is a Fallacy i.e. finding and fixing defects does not help if the system build is unusable and does not fulfil the user's needs & requirements.
- To solve this problem, the next principle of testing states that early Testing

23) Difference between QA v/s QC v/s Testing

S.N.	Quality Assurance	Quality Control	Testing
1	Activities which ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements.	Activities which ensure the verification of developed software with respect to documented (or not in some cases) requirements.	Activities which ensure the identification of bugs/error/defects in the Software.
2	Focuses on processes and procedures rather than conducting actual testing on the system.	Focuses on actual testing by executing Software with intend to identify bug/defect through implementation of procedures and process.	Focuses on actual testing.
3	Process oriented activities.	Product oriented activities.	Product oriented activities.
4	Preventive activities.	It is a corrective process.	It is a preventive process.
5	It is a subset of Software Test Life Cycle (STLC).	QC can be considered as the subset of Quality Assurance.	Testing is the subset of Quality Control.

24) Difference between Smoke and Sanity?

Smoke Testing	Sanity Testing
Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine	Sanity Testing is done to check the new functionality/bugs have been fixed
The objective of this testing is to verify the “stability” of the system in order to proceed with more rigorous testing	The objective of the testing is to verify the “rationality” of the system in order to proceed with more rigorous testing
This testing is performed by the developers or testers	Sanity testing in software testing is usually performed by testers
Smoke testing is usually documented or scripted	Sanity testing is usually not documented and is unscripted
Smoke testing is a subset of Acceptance testing	Sanity testing is a subset of Regression Testing
Smoke testing exercises the entire system from end to end	Sanity testing exercises only the particular component of the entire system
Smoke testing is like General Health Check Up	Sanity Testing is like specialized health check up

Smoke Testing:

- Smoke Testing can be understood as a software testing process that determines whether the deployed software build is stable or not.

Some Pros:

- It helps to find issues in the early phase of testing.
- Improves the overall quality of the system.
- Very limited number of test cases is required to do the Smoke testing.

- **Some Cons:**
- Smoke testing does is not detailed.
- Smoke testing sometimes causes wastage of time if the software build is not stable.
- This type of test is more suitable if you can automate; otherwise, a lot of time is spent manually executing the test cases.

Sanity Testing:

- Sanity Testing is a type of software testing that is performed after receiving a software build. The goal is to determine that the proposed functionality works approximately as expected.
- **Some Pros:**
- It helps to find related and missing objects.
- It saves lots of time and effort because it is focused on one or few areas of functionality.
- It is the simplest way to assure the quality of the product before it is developed.

- **Some Cons:**
- Its scope is relatively narrow, compared to the other types of testing.
- Sanity testing focuses only on the commands and functions of the software.
- Most of the time Sanity testing is not at all scripted, which may be a problem for some testers.

25) 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 specifications or not.	• It checks whether the software meets the requirements and expectations 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.

26) Explain types of Performance testing.

1) Stress Testing

- This test pushes an application beyond normal load conditions to determine which components fail first. Stress testing attempts to find the breaking point of the application and is used to evaluate the robustness of the application's data processing capabilities and response to high volumes of traffic.

2) Spike Testing

- This testing evaluates the ability of the application to handle sudden volume increases. It is done by suddenly increasing the load generated by a very large number of users. The goal is to determine whether performance will suffer, the system will fail, or it will be able to handle dramatic changes in load. This testing is critical for applications that experience large increases in number of users; for example, utility customers reporting power outages during storms. This can be considered a component of stress testing

3) Load Testing

- The purpose of load testing is to evaluate the application's performance under increasingly high numbers of users. Load, or increasing numbers of users are applied to the application under https://qualitestgroup.com/initiatives/load-and-performance-testing-services/test and the results are measured to validate the requirements are met. This load can be the expected concurrent number of users on the application performing a specific number of transactions within the set duration. This test will give out the response times of all the important business critical transactions. If the database, application server, etc. are also monitored, then this simple test can itself point towards bottlenecks in the application software.

4) Endurance Testing

- Endurance testing evaluates the performance of the system under load over time. It is executed by applying varying loads to the application under test for an extended period of time to validate that the performance

requirements related to production loads and durations of those loads are met. Endurance testing can be considered a component of load testing and is also known as soak testing.

5) Volume Testing

- Also known as flood testing, this testing is used to evaluate the application's ability to handle large volumes of data. The impact on response time and the behaviour of the application are analysed. This testing can be used to identify bottlenecks and to determine the capacity of the system. This type of performance testing is important for applications that deal with big data.

6) Scalability Testing

- This testing is used to determine your application's ability to handle increasing amounts of load and processing. It involves measuring attributes including response time, throughput, hits and requests per second, transaction processing speed, CPU usage, Network usage and more. Results of this testing can be used in the planning and design phases of development which reduces costs and mitigates the potential for performance issues.

27) What is Error, Defect, Bug and failure?

- We can say that a mistake made by a programmer during coding is called an error, an error found during the unit testing in the development phase is called a defect, an error found during the testing phase is called a bug and when an error is found at an end user's end is called as the failure.

DEFECT:

- It can be simply defined as a variance between expected and actual. The defect is an error found AFTER the application goes into production. It commonly refers to several troubles with the software products, with their external behaviour or with its internal features. In other words, a Defect is a difference between expected and actual results in the context of testing. It is the deviation of the customer requirement.

ERROR:

- An error is a mistake, misconception, or misunderstanding on the part of a software developer. In the category of the developer, we include software engineers, programmers, analysts, and testers. For example, a developer may misunderstand a de-sign notation, or a programmer might type a variable name incorrectly – leads to an Error. It is the one that is generated because of the wrong login, loop or syntax. The error normally arises in software; it leads to a change in the functionality of the program.

BUG:

- A bug is the result of a coding error. An Error found in the development environment before the product is shipped to the customer. A programming error that causes a program to work poorly, produce incorrect results or crash. An error in software or hardware that causes a program to malfunction. A bug is the terminology of Tester.

FAILURE:

- A failure is the inability of a software system or component to perform its required functions within specified performance requirements. When a defect reaches the end customer it is called a Failure. During development, Failures are usually observed by testers.

28) Difference between Priority and Severity.

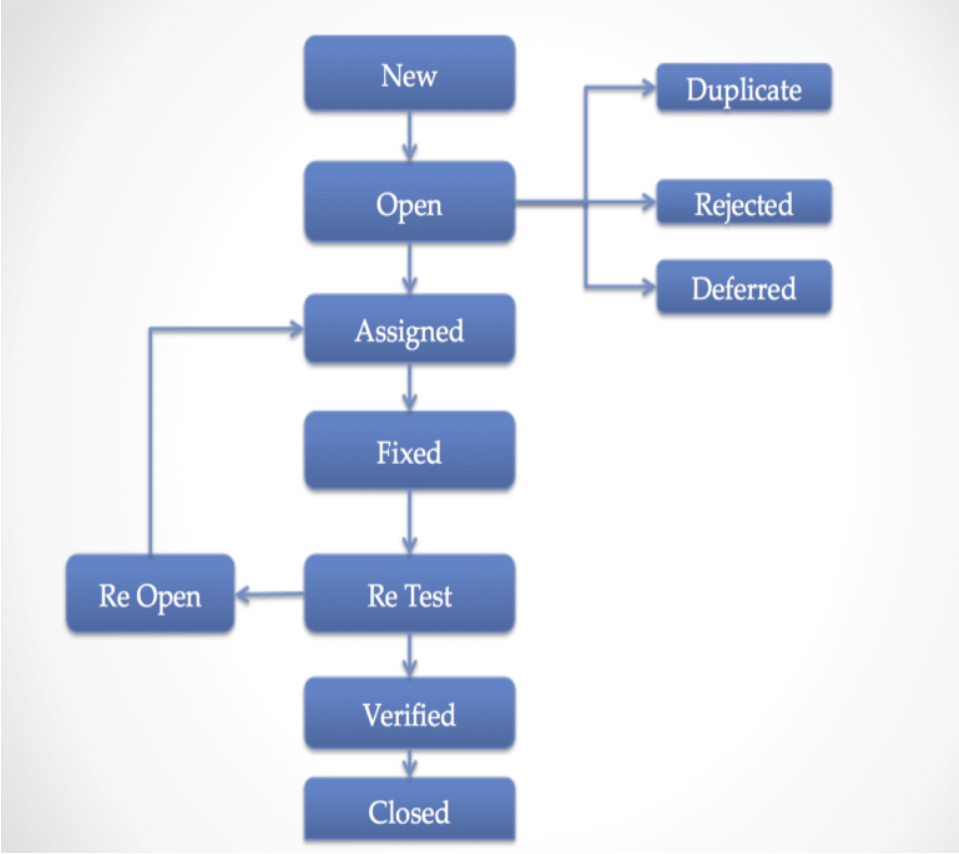
Parameters	**Severity in Testing**	**Priority in Testing**
Definition	Severity is a term that denotes how severely a defect can affect the functionality of the software.	Priority is a term that defines how fast we need to fix a defect.
Parameter	Severity is basically a parameter that denotes the total impact of a given defect on any software.	Priority is basically a parameter that decides the order in which we should fix the defects.
Relation	Severity relates to the standards of quality.	Priority relates to the scheduling of defects to resolve them in software.
Value	The value of severity is objective.	The value of priority is subjective.

Change of Value	The value of Severity changes continually from time to time.	The value of Priority changes from time to time.
Who Decides the Defect	The testing engineer basically decides a defect's severity level.	The product manager basically decides a defect's priority level.
Types	There are 5 types of Severities: Cosmetic, Minor, Moderate, Major, and Critical.	There are 3 types of Priorities: High, Medium, and Low.

29) What is Bug Life Cycle?

- The Defect Life Cycle, also known as the Bug Life Cycle, is a cycle of defects from which it goes through covering the different states in its entire life.

- This starts as soon as any new defect is found by a tester and comes to an end when a tester closes that defect assuring that it won't get reproduced again.



30) Explain the difference between Functional testing and Non Functional testing.

Functional Testing	Non Functional Testing
It tests 'What' the product does. It checks the operations and actions of an Application.	It checks the behaviour of an Application.
Functional testing is done based on the business requirement. It tests whether the actual result is working according to the expected result.	Non- functional testing is done based on the customer expectation and Performance requirement. It checks the response time, and speed of the software under specific conditions.
It is carried out manually. Example: Black box testing method.	It is more feasible to test using automated tools. Example: Loadrunner.
It tests as per the customer requirements. Customer feedback helps in reducing the risk factors of the product.	It tests as per customer expectations. Customer feedback is more valuable for non- functional testing as it helps to improve and lets the tester to know the expectation Of the customer.
It is testing the functionality of the software.	It is testing the performance of the functionality of the software.
Functional testing has the following types: •Unit testing •Integration testing •System Testing •Acceptance Testing	Non-functional testing includes: •Performance testing •Load Testing •Stress testing •Volume testing •Security testing •Installation testing •Recovery testing

31) What is the 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	STLC
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 (testers) are needed.
In SDLC, development team makes 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 in 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 the difference between test scenarios, test cases, and test script ?

Test Scenario	Test Case	Test Script
Is any functionality that can be tested.	Is a set of actions executed to verify particular features or functionality.	Is a set of instructions to test an app automatically.
Is derived from test artifacts like Business Requirement Specification (BRS) and Software Requirement Specification (SRS).	Is mostly derived from test scenarios.	Is mostly derived from test cases.
Helps test the end-to-end functionality in an Agile way.	Helps in exhaustive testing of an app.	Helps to test specific things repeatedly.
Is more focused on what to test.	Is focused on what to test and how to test.	Is focused on the expected result.
Takes less time and fewer resources to create.	Requires more resources and time.	Requires less time for testing but more resources for scripts creating and updating.
Includes an end-to-end functionality to be tested.	Includes test steps, data, expected results for testing.	Includes different commands to develop a script.
The main task is to check the full functionality of a software application.	The main task is to verify compliance with the applicable standards, guidelines, and customer requirements.	The main task is to verify that nothing is skipped, and the results are true as the desired testing plan.
Allows quickly assessing the testing scope.	Allows detecting errors and defects.	Allows carrying out an automatic execution of test cases.

33) Explain what Test Plan is? What is the information that should be covered ?

- A Test Plan is a detailed document that catalogs the test strategies, objectives, schedule, estimations, deadlines, and resources required to complete that project. Think of it as a blueprint for running the tests needed to ensure the software is working correctly – controlled by test managers.

As per IEEE 829 standards, the components of a Test Plan document should be :

- Test Plan identifier
- References
- Introduction
- Test Items
- Risks
- Items to be tested
- Features excluded for testing
- Testing Approach
- Test Pass and Fail Criteria
- Resumption/Suspension Criteria
- Test deliverables
- Test Environment Set Up
- Training and Staffing
- Team Member Responsibilities
- Testing Schedule
- Planning for Risks and Contingency Plans
- Approvals

34) What is Priority ?

- Priority is defined as the order in which the defects should be resolved. The priority status is usually set by the testing team while raising the defect against the dev team mentioning the timeframe to fix the defect.

35) What is Severity?

- One can define Severity as the extent to which any given defect can affect/ impact a particular software. Severity is basically a parameter that denotes the impact of any defect and its implication on a software's functionality. In other words, Severity defines the overall impact that any defect can have on a system.

36) Bug categories are…

- Functional errors
- Syntax errors
- Logic errors
- Calculation errors
- Unit-level bugs
- System-level integration bugs
- Out of bounds bugs

37) Advantages of Bugzilla.

- Open source, free bug tracking tool.
- Automatic Duplicate Bug Detection.
- Search option with advanced features.

- File/Modify Bugs By Email.
- Move Bugs Between Installs.
- Multiple Authentication Methods (LDAP, Apache server).
- Time Tracking.
- Automated bug reporting; has an API to interact with system.

38) Difference between Priority and Severity.

Parameters	**Severity in Testing**	**Priority in Testing**
Definition	Severity is a term that denotes how severely a defect can affect the functionality of the software.	Priority is a term that defines how fast we need to fix a defect.
Parameter	Severity is basically a parameter that denotes the total impact of a given defect on any software.	Priority is basically a parameter that decides the order in which we should fix the defects.
Relation	Severity relates to the standards of quality.	Priority relates to the scheduling of defects to resolve them in software.
Value	The value of severity is objective.	The value of priority is subjective.
Change of Value	The value of Severity changes continually from time to time.	The value of Priority changes from time to time.
Who Decides the Defect	The testing engineer basically decides a defect's severity level.	The product manager basically decides a defect's priority level.
Types	There are 5 types of Severities: Cosmetic, Minor, Moderate, Major, and Critical.	There are 3 types of Priorities: High, Medium, and Low.

39) What are the different Methodologies in Agile Development Model ?

- Kanban
- Scrum
- Extreme Programming (XP)
- Feature-driven development (FDD)
- Dynamic Systems Development Method (DSDM)
- Crystal
- Lean

40) Explain the difference between Authorization and Authentication in Web testing. What are the common problems faced 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.

Authentication	Authorization
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?**
Generally, transmit information through an ID Token.	Generally, transmit information through an Access Token.
The OpenID Connect (OIDC) protocol is an authentication protocol that is generally in charge of user authentication process.	The OAuth 2.0 protocol governs the overall system of user authorization process.
Popular Authentication Techniques- • Password-Based Authentication • Passwordless Authentication • 2FA/MFA (Two-Factor Authentication / Multi-Factor Authentication) • Single sign-on (SSO) • Social authentication	Popular Authorization Techniques- • Role-Based Access Controls (RBAC) • JSON web token (JWT) Authorization • SAML Authorization • OpenID Authorization • OAuth 2.0 Authorization
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.
The user authentication is visible at user end.	The user authorization is not visible at the user end.

Authentication	Authorization
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.
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 Common problems faced in Web Testing are…

- Integration. Integration testing exposes problems with interfaces among different program components before deployment. ...
- Interoperability. ...
- Security. ...
- Performance. ...
- Usability. ...
- Quality Testing, Exceptional Services.

- Verify that the user can create a group by adding multiple people from his contact list.
- Verify that the user can send and receive the message in group chats.
- Verify that users can send and receive images, audio, video, and emoticons in the chat with individuals.

41) When to used Usability Testing?

- If possible, usability testing can and should be conducted on the current iteration of a product before beginning any new design work, after you've begun the strategy work around a brand new site or app.

42)What is the procedure for GUI Testing?

- It extensively checks the user-interface of the application under test.

- Testing the size, position, height, width of the visual elements
- Verifying and testing the error messages are displayed or not
- Testing different sections of the display screen
- Verifying the usability of carousel arrows
- Checking the navigation elements at the top of the page
- Checking the message displayed, frequency and content
- Verifying the functionality of proper filters and ability to retrieve results.
- Checking alignment of radio buttons, drop downs
- Verifying the title of each section and their correctness
- Cross-checking the colors and its synchronization with the theme

43) Write agile Manifesto Principles.

- **Customer satisfaction through early and continuous software delivery** –
  Customers are happier when they receive working software at regular intervals, rather than waiting extended periods of time between releases.
- **Accommodate changing requirements throughout the development process** – The ability to avoid delays when a requirement or feature request changes.
- **Frequent delivery of working software** – Scrum accommodates this principle since the team operates in software sprints or iterations that ensure regular delivery of working software.
- **Collaboration between the business stakeholders and developers throughout the project** – Better decisions are made when the business and technical team are aligned.
- **Support, trust, and motivate the people involved** – Motivated teams are more likely to deliver their best work than unhappy teams.
- **Enable face-to-face interactions** – Communication is more successful when development teams are co-located.

- **Working software is the primary measure of progress** – Delivering functional software to the customer is the ultimate factor that measures progress.

- **Agile processes to support a consistent development pace** – Teams establish a repeatable and maintainable speed at which they can deliver working software, and they repeat it with each release.
- **Attention to technical detail and design enhances agility** – The right skills and good design ensures the team can maintain the pace, constantly improve the product, and sustain change.
- **Simplicity** – Develop just enough to get the job done for right now.
- **Self-organizing teams encourage great architectures, requirements, and designs** –Skilled and motivated team members who have decision-making power, take ownership, communicate regularly with other team members, and share ideas that deliver quality products.
- **Regular reflections on how to become more effective** – Self improvement, process improvement, advancing skills, and techniques help team members work more efficiently.