**Software Testing: - Assignment -2**

**1.What is Exploratory testing?**

* Exploratory Testing is a type of software testing where Test cases are not created in advance but testers check system on the fly. They may note down ideas about what to test before test execution.
* Under scripted testing, you design test cases first and later proceed with test execution. On the contrary, exploratory testing is a simultaneous process of test design and test execution all done at the same time.
* Scripted Test Execution is usually a non-thinking activity where testers execute the test steps and compare the actual results with expected results.  Such test execution activity can be automated does not require many cognitive skills.
* **Advantages**
* This testing is useful when requirement documents are not available or partially available
* It involves Investigation process which helps find more bugs than normal testing-
* Uncover bugs which are normally ignored by other testing techniques
* Helps to expand the imagination of testers by executing more and more test cases which finally improves productivity as well
* This testing drill down to the smallest part of an application and covers all the requirements
* This testing covers all the types of testing and it covers various scenarios and cases
* Encourages creativity and intuition
* Generation of new ideas during test execution
* **Disadvantages**
* This testing purely depends on the tester skills
* Limited by domain knowledge of the tester
* Not suitable for long execution time

2. What is traceability matrix?

* Test conditions should be able to be linked back to their sources in the test basis, this is known as traceability.
* To protect against changes, you should be able to trace back from every system component to the original requirement that caused its presence.
* Types of Traceability Matrices:
* **Forward Traceability**: Mapping of Requirements to Test cases
* **Backward Traceability**: Mapping of Test Cases to Requirements
* **Bi-Directional Traceability**: A Good Traceability matrix is the References from test cases to basis documentation and vice versa.

Components Used in Traceability Matrices:

1. **Requirement ID**: Unique identifier for each requirement.

2. **Requirement Description**: Detailed description of the requirement.

3. **Test Case ID(s)**: Identifier(s) for test case(s) that validate the requirement.

4. **Status**: Indicates the testing status (e.g., Passed, Failed, Not Tested).

5. **Remarks**: Any additional comments or notes.

* Pros of Traceability Matrix: -
* Make obvious to the client that the software is being developed as per the requirements.
* To make sure that all requirements included in the test cases.
* To make sure that developers are not creating features that no one has requested.
* Easy to identify the missing functionalities.
* If there is a change request for a requirement, then we can easily find out which test cases need to update.
* **Cons of Traceability Matrix: -**
* No traceability or Incomplete Traceability Results.
* Poor or unknown test coverage, more defects found in production.
* It will lead to miss some bugs in earlier test cycles which may arise in later test cycles. Then a lot of discussions arguments with other teams and managers before release.
* Difficult project planning and tracking, misunderstandings between different teams over project dependencies, delays, etc

**3. What is Boundary value testing?**

* Boundary value analysis is a methodology for designing test cases that concentrates software testing effort on cases near the limits of valid ranges.
* Boundary value analysis is a method which refines equivalence partitioning.
* Boundary value analysis generates test cases that highlight errors better than equivalence partitioning.
* The trick is to concentrate software testing efforts at the extreme ends of the equivalence classes.
* At those points when input values change from valid to invalid errors are most likely to occur.

4. What is Equivalence partitioning testing?

* Equivalence Partitioning Testing is testing technique that divides the input data of a system into different equivalence classes, where each class represents a group of inputs that are expected to behave similarly. Instead of testing every possible input, a single representative value from each class is tested, reducing the number of test cases while maintaining test effectiveness.
* Equivalence partitioning is a black-box testing technique that identifies input ranges or categories to group inputs with similar expected behaviour, ensuring efficient and sufficient testing.
* EP says that by testing just one value we have tested the partition (typically a mid-point value is used). It assumes that:
* If one value finds a bug, the others probably will too.
* If one doesn't find a bug, the others probably won't either.
* In EP we must identify Valid Equivalence partitions and Invalid Equivalence partitions where applicable.
* The Valid partition is bounded by the values 1 and 100.

5. What is Integration testing?

* Integration testing is a process that focuses on verifying the interactions and data exchange between different components or modules of a software application.
* The goal of integration testing is to identify any problems or bugs that arise when different components are combined and interact with each other.
* Integration testing is typically performed after unit testing and before system testing.
* It helps to identify and resolve integration issues early in the development cycle, reducing the risk of more severe and costly problems later on.
* Integration testing can be done by picking module by module. This can be done so that there should be a proper sequence to be followed. And also, if you don’t want to miss out on any integration scenarios then you have to follow the proper sequence.
* Exposing the defects is the major focus of the integration testing.
* Integration testing is important because it verifies that individual software modules or components work together correctly as a whole system.
* There are four types of integration testing approaches. Those approaches are the following:
* **Top-Down Integration Testing**:
* Starts testing from the top-level modules and gradually integrates lower-level modules.
* Stubs (dummy modules) simulate lower-level components.
* **Bottom-Up Integration Testing**:
* Begins testing from the lowest-level modules and progresses upward.
* Drivers (test harnesses) simulate higher-level modules.
* **Big-Bang Integration Testing**:
* All modules are integrated and tested simultaneously.
* Suitable for smaller systems but challenging for large, complex systems.
* **Incremental Integration Testing**:
* Combines and tests modules in small, logical groups incrementally.
* Easier to identify and fix defects compared to Big-Bang.

6. What determines the level of risk?

* Risk – ‘A factor that could result in future negative consequences; usually expressed as impact and likelihood’.
* When testing does find defects, the Quality of the software system increases when those defects are fixed.
* Risks are of two types: -
* Project Risks
* Product Risk
* Example of Project risk is Senior Team Member leaving the project abruptly.
* Every risk is assigned a likelihood i.e. chance of it occurring, typically on a scale of 1 to 10. Also, the impact of that risk is identified on a scale of 1- 10.
* But just identifying the risk is not enough. You need to identify mitigation. In this case mitigation could be Knowledge Transfer to other team members & having a buffer tester in place.
* Example of product risks would be Flight Reservation system not installing in test environment.
* Mitigation in this case would be conducting a smoke or sanity testing. Accordingly, you will make changes in your scope items to include sanity testing.

7. What is Alpha testing?

* Alpha testing is the first phase of user acceptance testing (UAT) conducted by the internal development team before the software is released to external testers or users. It aims to identify bugs or issues in the software and verify its functionality under real-world conditions.
* It is always performed by the developers at the software development site.
* Sometimes it is also performed by Independent Testing Team.
* Alpha Testing is not open to the market and public.
* It is conducted for the software application and project.
* It is always performed in Virtual Environment.
* It is always performed within the organization.
* It is the form of Acceptance Testing.
* Alpha Testing is definitely performed and carried out at the developing organization’s location with the involvement of developers.
* It comes under the category of both White Box Testing and Black Box Testing.

8. What is beta testing?

* Beta testing is a phase in software testing where a version of the product is released to a select group of external users (beta testers) outside the development team to test the product in real-world environments and provide feedback on bugs, performance, and user experience.
* Beta testing can be considered “pre-release” testing.
* Pilot Testing is testing to product on real world as well as collect data on the use of product in the classroom.
* It is always performed by the customers at their own site.
* It is not performed by Independent Testing Team.
* Beta Testing is always open to the market and public.
* It is usually conducted for software product.
* It is performed in Real Time Environment.
* It is always performed outside the organization.
* It is also the form of Acceptance Testing.
* Beta Testing (field testing) is performed and carried out by users or you can say people at their own locations and site using customer data.
* It is only a kind of Black Box Testing.

9. What is component testing?

* A minimal software item that can be tested in isolation. It means “A unit is the smallest testable part of software.”
* The testing of individual software components.
* Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended with debugging tool.
* component 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.
* component testing is performed by using the White Box Testing method.
* Unit testing in Extreme Programming involves the extensive use of testing frameworks.
* A unit test framework is used in order to create automated unit tests.
* Unit testing frameworks are not unique to extreme programming, but they are essential to it.
* Below we look at some of what extreme programming brings to the world of unit testing:
* Tests are written before the code.
* Rely heavily on testing frameworks.
* All classes in the applications are tested.
* Quick and easy integration is made possible.

10. What is functional system testing?

* A requirement that specifies a function that a system or system component must perform.
* Functional System Testing Functionality As below:
* Accuracy Provision of right or agreed results or effects.
* Interoperability Ability to interact with specified systems.
* Compliance Adhere to applicable standards, conventions, regulations or laws.
* Auditability Ability to provide adequate and accurate audit data.
* Suitability Presence and appropriateness of functions for specified tasks.
* There are two types of Test Approach: -
* Requirement Based Functional Testing
* Business Process Based Testing
* Requirement Based Functional Testing: -
* Testing against requirements and specifications
* Test procedures and cases derived from:

1. detailed user requirements.
2. system requirements functional specification.
3. User documentation/instructions.
4. high level System design.

* Starts by using the most appropriate black-box testing techniques.
* May support this with white-box techniques (e.g. menu structures, web page navigation).
* Risk based approach.
* Business Process Based Testing: -
* Test procedures and cases derived from:

1. Expected user profiles.
2. Business scenarios.
3. Use cases.

* Testing should reflect the business environment and processes in which the system will operate.
* Therefore, test cases should be based on real business processes.

11. What is Non-Functional Testing?

* Testing the attributes of a component or system that do not relate to functionality, e.g. reliability, efficiency, usability, interoperability, maintainability and portability.
* Non-functional testing includes, but is not limited to, performance testing, load testing, stress testing, usability testing, maintainability testing, reliability testing and portability testing.
* It is the testing of “how” the system works. Non-functional testing may be performed at all test levels.
* To address this issue, performance testing is carried out to check & fine tune system response times. The goal of performance testing is to reduce response time to an acceptable level.
* Hence load testing is carried out to check systems performance at different loads i.e. number of users accessing the system.
* Non - Functional Testing Examples: -

1. **Web Based Testing:**

* Identify the software processes that directly influence the overall performance of the system.
* In website number of user/customer will increase, how the website will handle to every customer/user.

1. **Desktop Based Testing:**

* Numerous other such GUI test cases, the desktop application tester must view.
* Guarantee that error messages are instructive and helpful for the client.
* Memory, and different other issues.

1. **Mobile Based Testing:**

* In mobile, automatically will switch off without any reason.
* To stop the application which is not in our hand.

1. **Game Based Testing:**

* Confirms workability and stability of the software.
* Validate whether the user interface of the app is as per the screen size of the device and ensure high quality.

**12. What is GUI Testing?**

* Graphical User Interface (GUI) testing is the process of testing the system’s GUI of the System under Test. GUI testing involves checking the screens with the controls like menus, buttons, icons, and all types of bars – tool bar, menu bar, dialog boxes and windows etc.
* Check all the GUI elements for size, position, width, length and acceptance of characters or numbers. For instance, you must be able to provide inputs to the input fields.
* Check you can execute the intended functionality of the application using the GUI.
* Check Error Messages are displayed correctly.
* Check for Clear demarcation of different sections on screen.
* Check Font used in application is readable.
* Check the alignment of the text is proper.
* Check the Color of the font and warning messages is aesthetically pleasing.
* Check that the images have good clarity.
* Check that the images are properly aligned.
* Check the positioning of GUI elements for different screen resolution.
* GUI Testing Examples: -
* Web Based Testing & Desktop Based Testing.
* Mobile Based Testing.
* Game Based Testing.

13. What is Adhoc testing?

* Adhoc testing is an informal testing type with an aim to break the system.
* It does not follow any test design techniques to create test cases.
* In fact, is does not create test cases altogether!
* This testing is primarily performed if the knowledge of testers in the system under test is very high.
* Main aim of this testing is to find defects by random checking.
* Adhoc testing can be achieved with the testing technique called Error Guessing.
* The Error guessing is a technique where the experienced and good testers are encouraged to think of situations in which the software may not be able to cope.
* Some people seem to be naturally good at testing and others are good testers because they have a lot of experience either as a tester or working with a particular system and so are able to find out its weaknesses.
* There are different types of Adhoc testing and they are listed as below:
* Buddy Testing: - Two buddies mutually work on identifying defects in the same module. Mostly one buddy will be from development team and another person will be from testing team
* Pair Testing: - Two testers are assigned modules, share ideas and work on the same
* machines to find defects.
* Monkey Testing: - Randomly test the product or application without test cases with a goal to break the system.

14. What is load testing?

* It’s a performance testing to check system behavior under load. Testing an application under heavy loads, such as testing of a web site under a range of loads to determine at what point the system’s response time degrades or fails.
* Load testing is a kind of performance testing which determines a system’s performance under real-life load conditions. This testing helps determine how the application behaves when multiple users access it simultaneously.
* Some extremely popular sites have suffered serious downtimes when they get massive traffic volumes. E-commerce websites invest heavily in advertising campaigns, but not in Load Testing to ensure optimal system performance, when that marketing brings in traffic.
* Load testing gives confidence in the system & its reliability and performance. Load Testing helps identify the bottlenecks in the system under heavy user stress scenarios before they happen in a production environment. Load testing gives excellent protection against poor performance and accommodates complementary strategies for performance management and monitoring of a production environment.
* Pros:
* Performance bottlenecks identification before production.
* Improves the scalability of the system.
* Minimize risk related to system down time.
* Reduced costs of failure.
* Increase customer satisfaction.
* Cons:
* Need programming knowledge to use load testing tools.
* Tools can be expensive as pricing depends on the number of virtual users supported.

15. What is stress Testing?

* System is stressed beyond its specifications to check how and when it fails. Performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous input to system or database load.
* It even tests beyond the normal operating point and evaluates how the system works under those extreme conditions.
* Stress Testing is done to make sure that the system would not crash under crunch situations.
* Stress testing is also known as endurance testing.
* Under Stress Testing, AUT is be stressed for a short period of time to know its withstanding capacity.
* Most prominent use of stress testing is to determine the limit, at which the system or software or hardware breaks.
* The goal of stress testing is to analyze the behavior of the system after failure. For stress testing to be successful, system should display appropriate error message while it is under extreme conditions.
* Types of Stress Testing: -
* Application Stress Testing
* Transactional Stress Testing
* Systemic Stress Testing
* Exploratory Stress Testing
* Stress Testing Tools:
* Stress Tester
* Neo Load
* App Perfect

16. 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.
* 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 tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.
* White Box Testing Examples: -
* Web Based Testing:
* Analyze the logic by reading the code.
* Code optimization Suggest if any optimization can be done in the code which is better than the existing one.
* Deploy the code using different web servers which could be a case study even if the product is not supporting other web servers.
* Desktop Based Testing:
* When we debug the code when we writing.
* Mobile Based Testing:
* The Android SDK and related plugin for Eclipse.
* Android devices enabled for development and debugging, with appropriate USB drivers as necessary.
* Game Based Testing:
* When we connect with remote device, so which device we connect will check in code.
* When some debug the code and play at that time game.

17. What is black box testing? What is the different black box testing techniques?

* Black-box testing: Testing, either functional or non-functional, without reference to the internal structure of the component or system.
* Specification-based testing technique is also known as ‘black-box’.
* The testers have no knowledge of how the system or component is structured inside the box. In black-box testing the tester is concentrating on what the software does, not how it does it.
* The technique of testing without having any knowledge of the interior workings of the application is Black Box testing.
* When performing a black box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.
* There are different black box testing techniques and they are:
* Equivalence partitioning
* Boundary value analysis
* Decision tables
* State transition testing
* Use-case Testing
* Other Black Box Testing
* Syntax or Pattern Testing

18. Mention what bigbang testing is?

* In Big Bang integration testing all components or modules is integrated simultaneously, after which everything is tested as a whole.
* Big Bang testing has the advantage that everything is finished before integration testing starts.
* The major disadvantage is that in general it is time consuming and difficult to trace the cause of failures because of this late integration. Here all component is integrated together at once, and then tested.
* Advantages:
* Convenient for small systems.
* Disadvantages:
* Fault Localization is difficult.
* Since the integration testing can commence only after “all” the modules are designed, 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.

19. What is the purpose of exit criteria?

* The Purpose of exit criteria are:
* **Quality Assurance**: Ensures that the software meets the required quality standards before moving to the next phase or releasing it to production.
* **Defect Tracking**: Confirms that critical defects have been identified, resolved, or mitigated to acceptable levels.
* **Risk Mitigation**: Minimizes the risk of software failures in later stages by ensuring thorough testing and issue resolution.
* **Project Management**: Provides measurable benchmarks for project progress and readiness to proceed.
* **Compliance**: Ensures compliance with standards, regulations, or contractual obligations.
* **Customer Satisfaction**: Guarantees that the product meets customer expectations and requirements.

20. When should "Regression Testing" be performed?

* Regression Testing is performed whenever there are changes in the software to ensure that the modifications have not adversely affected the existing functionality.
* When to Perform Regression Testing: -
* **After Bug Fixes**:
* To verify that fixing one defect has not introduced new defects or disrupted related functionality.
* **After Feature Enhancements**:
* When new features are added, regression testing ensures they have not caused issues in existing features.
* **During Code Optimization**:
* To confirm that performance improvements or refactoring have not altered the behavior of the system.
* **After Integration of Modules**:
* To check that integrating new modules or components into the system does not affect the functioning of existing ones.
* **Before Major Releases**:
* Regression testing ensures the overall stability of the software before delivering it to users.
* **After Environment Changes**:
* When there are changes to the software’s environment, such as updates to libraries, operating systems, or hardware, regression testing ensures compatibility.
* **During Continuous Testing in CI/CD Pipelines**:
* In agile and DevOps practices, regression testing is performed frequently and automatically to maintain the stability of the application during continuous integration and deployment.

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

1. Testing shows presence of Defects

2. Exhaustive Testing is Impossible!

3. Early Testing

4. Defect Clustering

5. The Pesticide Paradox

6. Testing is Context Dependent

7. Absence of Errors Fallacy

1. Testing shows presence of Defects

* Testing can show that defects are present, but cannot prove that there are no defects.
* 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.
* As we find more defects, the probability of undiscovered defects remaining in a system reduces.
* However, Testing cannot prove that there are no defects present.

2. Exhaustive Testing is Impossible!

* 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.
* Exhaustive testing of complex software applications:
* requires enormous resources.
* is too expensive.
* takes too long.
* It is therefore impractical.
* Need an alternative that is pragmatic, affordable, timely and provides results.

3.Early Testing

* Testing activities should start as early as possible in the software or system development life cycle, and should be focused on defined objectives.
* Testing activities should start as early as possible in the development life cycle.
* These activities should be focused on defined objectives – outlined in the Test Strategy.

4.Defect Clustering

* A small number of modules contain most of the defects discovered during pre-release testing, or are responsible for the most operational failures.
* Defects are not evenly spread in a system.
* They are ‘clustered’.
* In other words, most defects found during testing are usually confined to a small number of modules.
* Similarly, most operational failures of a system are usually confined to a small number of modules.

5.The Pesticide Paradox

* If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new defects.
* To overcome this “pesticide paradox”, the test cases need to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects.
* Testing identifies bugs, and programmers respond to fix them.
* As bugs are eliminated by the programmers, the software improves.
* As software improves the effectiveness of previous tests erodes.
* Therefore, we must learn, create and use new tests based on new techniques to catch new bugs.

6. Testing is Context Dependent

* Testing is basically context dependent.
* Testing is done differently in different contexts.
* Different kinds of sites are tested differently.
* For example: -Safety – critical software is tested differently from an e-commerce site.

7. Absence of Errors Fallacy

* If the system built is unusable and does not fulfill the user’s needs and expectations then finding and fixing defects does not help.
* If we build a system and, in doing so, find and fix defects.
* It doesn’t make it a good system.
* Even after defects have been resolved it may still be unusable and/or does not fulfil the users’ needs and expectations.

22. Difference between Smoke and Sanity?

| **Smoke Testing** | **Sanity Testing** |
| --- | --- |
| [Smoke testing](https://www.geeksforgeeks.org/smoke-testing-software-testing/) is done to assure that the acute functionalities of program are working fine. | [Sanity testing](https://www.geeksforgeeks.org/sanity-testing-software-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 measure the stability of the system/product by performing testing. | Sanity testing is done to measure 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 functions 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. |
| It includes all the system’s essential basic functionality. | It includes only those modules where change in code is made. |
| Smoke Testing firstly performs on the initial build. Smoke testing is done first. | Sanity Testing is done on stable builds or for the introduced new features in the software. |
| Smoke testing can be carried out either way-manually or automatically. | Without using test cases or scripts sanity testing can be carried out. |
| There is end-to-end system verification done in smoke testing. | A specific component gets verified in sanity testing. |
| In the smoke testing process, the software build could be stable or unstable. | During sanity testing, the software build is comparatively stable. |
| For every new build release smoke testing is carried out. | Sanity testing is carried out when in-depth testing is not possible because of short time. |

23. Difference between verification and Validation.

| **Factor** | **Verification** | **Validation** |
| --- | --- | --- |
| **Objective** | To check if the product meets specified requirements/designs. | To ensure the product meets user needs and expectations. |
| **Focus** | Process correctness and adherence to specifications. | Product effectiveness in real-world scenarios. |
| **Timing** | Conducted throughout the development process. | Generally conducted after verification, closer to product completion. |
| **Methodology** | Involves methods like code reviews, static analysis, and inspections. | Involves user acceptance testing, beta testing, and usability studies. |
| **Performed by** | Engineers and developers focus on technical aspects. | End-users, stakeholders, or QA teams focusing on user experience. |
| **Outcome** | Assurance that the product is built correctly according to the design. | Confidence that the product fulfills its intended use and satisfies user requirements. |
| **Feedback Loop** | Internal, focuses on correcting issues against specifications. | External, often lead to product adjustments based on user feedback. |
| **Documentation** | Specifications, design documents, and [test reports](https://www.testbytes.net/resource/10-tips-for-better-bug-reporting/). | User requirements, test scenarios, and feedback reports. |

24. Explain types of Performance testing.

* There 6 types of performance testing.

1.Load testing

2.Stress testing

3.Endurance testing

4.Spike testing

5.Volume testing

6.Scalability testing

* **Load testing**
* Load testing is a [type of testing](https://artoftesting.com/types-of-testing) which involves evaluating the performance of the system under the expected workload. A typical load test includes determining the response time, throughput, error rate, etc during the course of the load test.
* **Stress testing**
* Stress testing is a type of performance testing where we evaluate the application’s performance at a load much higher than the expected load. Another aspect of the stress testing is to determine the break-point of the application, the point at which the application fails to respond in the correct manner.
* **Endurance testing**
* Endurance testing is also known as ‘Soak Testing’. It is done to determine if the system can sustain the continuous expected load for a long duration. Issues like memory leakage are found with endurance testing.
* **Spike testing**
* In spike testing, we analyze the behavior of the system on suddenly increasing the number of users. It also involves checking if the application is able to recover after the sudden burst of users.
* **Volume testing**
* The volume testing is performed by feeding the application with a high volume of data. The application can be tested with a large amount of data inserted in the database or by providing a large file to the application for processing. Using volume testing, we can identify the bottleneck in the application with a high volume of data.

25. Difference between QA v/s QC v/s Tester

| **Quality Assurance** | Quality Control | Testing |
| --- | --- | --- |
| 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. |
| 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. |
| Process oriented activities. | Product oriented activities. Product oriented a | Product oriented activities. |
| Preventive activities | It is a corrective process. | It is a preventive process. |
| 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. |

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

* Error:- A mistake in coding is called error.
* Defect:- Error found by tester is called defect.
* Bug:- Defect accepted by development team then it is called bug.
* Failure:- The inability of a system or component to perform its required functions within specified performance requirements. See: bug, crash, exception, and fault.

27. 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 behavior of an application. |
| It is based on requirements of customer. | It is based on expectations of customer. |
| It helps to enhance the behavior 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. |
| **Examples:**  **1.** [Unit Testing](https://www.geeksforgeeks.org/unit-testing-software-testing/) **2.** [Smoke Testing](https://www.geeksforgeeks.org/smoke-testing-software-testing/) **3.** [Integration Testing](https://www.geeksforgeeks.org/software-engineering-integration-testing/) **4.** [Regression Testing](https://www.geeksforgeeks.org/software-engineering-regression-testing/) | **Examples:**  **1.** [Performance Testing](https://www.geeksforgeeks.org/performance-testing-software-testing/) **2.** [Load Testing](https://www.geeksforgeeks.org/software-testing-load-testing/) **3.** [Stress Testing](https://www.geeksforgeeks.org/stress-testing-software-testing/) **4.** [Scalability Testing](https://www.geeksforgeeks.org/software-testing-scalability-testing/) |

28. 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 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. |

29. What is the difference between test scenarios, test cases, and test script?

| Test scenarios | Test cases | Test script |
| --- | --- | --- |
| A test scenario is a high-level document that describes end-to-end functionality to be tested. | Test Case is a step-by-step procedure to test any functionality of the software application/product. | Test Script is set of instructions or a short program to test any functionality of software application/product. |
| Test scenarios are focused on what to test. | Test Case is a manual approach of software testing. | Test Script is an automatic approach of software testing. |
| These are high-level actions. | 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. |
| Writing the test scenario’s primary objective is an address end to get rid of the functionality of a software program. | Point by point test case configuration encourages tester to test viably. | Automatic testing approach is beneficial for constant execution. |
| It will take less time as compared to test cases. | 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 scenarios are really easy to maintain due to their high-level design. | Test case is developed in form of templates. | Test script is developed in form of scripting. |
| Test scenario helps in agile way of testing of the application | Test Cases are written manually. | Test Scripting is done by scripting format. |
| Fewer resources are sufficient to write test scenarios. | Test Case is used in manual testing environment. | Test Script is used in automatic testing environment. |
| Test scenarios are usually derived from documents like SRS, BRS, etc. | Test Cases are classified as delegated, positive, reusable, negative and UI test cases. | Test Script are characterized as manual test script and automation test scripts. |
| Test scenarios can be ambiguous as these are one-liners. | Test Case comprises of many test qualities like conditions, 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. |
| Test scenarios are high-level actions. | Requires more resources and time. | Requires less time for testing scripts. |
| It helps in an agile way of testing the end to end functionality | Suitable for initial phases of testing when software requirements are still changing. | Suitable for stable software where regression testing is frequent. |

30. Explain what Test Plan is? What is the information that should be covered.

* Test Planning in STLC is a phase in which a Senior QA manager determines the test plan strategy along with efforts and cost estimates for the project.
* Moreover, the resources, test environment, test limitations and the testing schedule are also determined.
* The Test Plan gets prepared and finalized in the same phase.

⚫ Activities in Requirement Phase Testing:-

* Preparation of test plan/strategy document for various types of testing.
* Test tool selection.
* Test effort estimation.
* Resource planning and determining roles and responsibilities.
* Training requirement

⚫ Deliverables of Requirement Phase Testing:-

* Test plan /strategy document.
* Effort estimation document.

**31.What are the different Methodologies in Agile Development Model?**

* Agile development encompasses a variety of methodologies that adhere to the Agile Manifesto principles, focusing on iterative development, collaboration, and adaptability.
* Here are the primary Agile methodologies:

1. **Scrum:-**

* SCRUM is an agile development method which concentrates particularly on how to manage tasks within a team-based development environment.
* Basically, Scrum is derived from activity that occurs during rugby match. Scrum believes in empowering the development team and advocates working in small teams.
* It consists of three roles and their responsibilities are explained as follows:

⚫ Scrum Master: Master is responsible for setting up the team, sprint meeting and removes obstacles to progress.

⚫ Product owner: The Product Owner creates product backlog, prioritizes the backlog and is responsible for the delivery of the functionality at each iteration.

⚫ Scrum Team: Team manages its own work and organizes the work to complete the sprint or cycle.

**2.Kanban: -**

* Kanban is a very popular framework for development in the agile software development methodology.
* It provides a transparent way of visualizing the tasks and work capacity of a team.
* It mainly uses physical and digital boards to allow the team members to visualize the current state of the project they are working on.

**32. Explain the difference between Authorization and Authentication in Web testing.What are the common problems faced in Web testing?**

|  |  |
| --- | --- |
| **Authentication** | **Authorization** |
| Process of verifying the identity of a user or system. | Process of determining what actions or resources a user has access to. |
| |  | | --- | |  |  |  | | --- | | Confirms "Who are you**?"** | | Confirms **"**What are you allowed to do?" |
| Ensures that the user is genuine and valid. | Ensures that the authenticated user has appropriate permissions. |
| Occurs before authorization. | Occurs after authentication. |
| |  | | --- | |  |  |  | | --- | | Incorrect username or password results in access denial. | | A valid user trying to access admin functionality without proper permissions. |
| Handled by login mechanisms (e.g., password managers, OAuth). | Managed by access control mechanisms (e.g., ACLs, RBAC). |
| Logging in with a username and password, biometric scan, or OTP. | Granting access to admin-only pages or specific functionalities like editing or deleting data. |
| Implements login credentials, multi-factor authentication (MFA), or Single Sign-On (SSO). | Implements role-based access control (RBAC), policies, or permissions. |
| |  | | --- | |  |  |  | | --- | | Provides the user access to the system. | | Restricts or grants access to specific actions or resources within the system. |