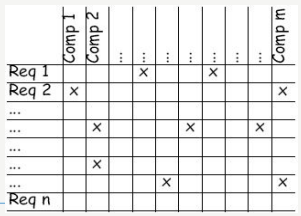
**Module – 2 (Manual Testing)**

* **What is Exploratory Testing ?**
* To understand the requirement completely, first we have to explore the application, hence it is called as exploratory.
* We have **to explore the application to understand the requirement completely**, then only we will start testing.
* Test engineer will do testing, when there is no requirement at all or the requirements are missing.
* **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.
* A software process should help you keeping the virtual table up-to-date.
* Simple technique may be quite valuable (naming convention)



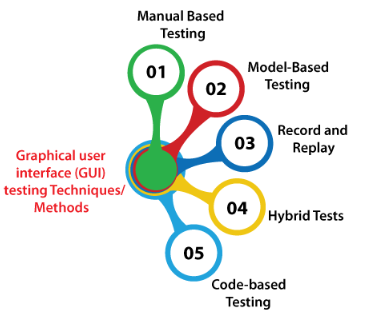
* Types of Matrix:

1. Forward Traceability – Mapping of Requirements to Test cases Backward Traceability
2. Mapping of Test Cases to Requirements
3. Bi-Directional Traceability - A Good Traceability matrix is the References from test cases to basis documentation and vice versa.

* Advantage:
* 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.
* Disadvantage:
* 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.
* Difficult project planning and tracking, misunderstandings between different teams over project dependencies, delays, etc
* **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.
* Boundary Value Analysis (BVA) uses the same analysis of partitions as EP and is usually used in conjunction with EP in test case design
* **What is Equivalence partitioning testing?**
* Equivalence partitioning is a technique of software testing in which **input data is divided into partitions of valid and invalid values, and it is mandatory that all partitions must exhibit the same behavior.**
* If a condition of **one partition is true**, then the condition of **another equal partition must also be true**, and if a condition of **one partition is false**, then the condition of **another equal partition must also be false.**
* The principle of equivalence partitioning is, test cases should be designed to cover each partition at least once. Each value of every equal partition must exhibit the same behavior as other.
* The equivalence partitions are derived from requirements and specifications of the software. The advantage of this approach is, it helps to reduce the time of testing due to a smaller number of test cases from infinite to finite. It is applicable at all levels of the testing process.
* **What is Integration testing?**
* Integration Testing - Testing performed to **expose defects in the interfaces and in the interactions between integrated components or systems** .
* Integration Testing is a level of the software testing process where individual units are combined and tested as a group.
* The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.
* There are **2 levels** of Integration Testing
* **Component Integration Testing**
* **System Integration Testing**
* **What determines the level of risk?**
* The level of risk is determined by several factors that collectively influence the likelihood and impact of potential negative events or outcomes. While the specific factors may vary depending on the context, here are some common elements that contribute to assessing and determining the level of risk.
* **What is Alpha testing?**
* It is always **performed by the developers at the software development site.**
* Alpha Testing is not open to the market and public
* It is always performed in Virtual Environment.
* It is always performed within the organization.
* It is the form of Acceptance Testing.
* It comes under the category of both White Box Testing and Black Box Testing.
* **What is beta testing?**
* It is always **performed by the customers at their own site.**
* Beta Testing is always open to the market and public.
* It is performed in Real Time Environment.
* It is always performed outside the organization.
* It is also the form of Acceptance Testing.
* It is only a kind of Black Box Testing.
* **What is component testing?**
* Component testing, also known as **module testing or unit testing**, is a software testing technique that **focuses on testing individual components or modules of a software system in isolation.**
* The purpose of component testing is to **verify the functionality, behavior, and correctness of each component independently** to ensure that they work as intended.
* Component testing is an integral part of the software testing process, complementing other testing techniques such as integration testing and system testing.
* It provides a solid foundation for building reliable and robust software systems by validating the functionality and correctness of individual components before they are combined.
* **What is functional system testing?**
* Functional system testing is a software testing technique that **focuses on verifying the functional requirements and behavior of a software system as a whole.**
* It involves testing the system's features, functions, and interactions to ensure that it meets the specified functional requirements and operates as intended.
* Functional system testing is an essential part of the software testing process, complementing other testing techniques such as unit testing, integration testing, and acceptance testing.
* It focuses on validating the system's functional requirements and behavior to ensure that the software system performs as expected and meets user expectations.
* **What is Non-Functional Testing?**
* Non-functional testing is a type of software testing that **focuses on evaluating the characteristics and attributes of a software system, rather than its specific functionalities.**
* It aims to assess the system's performance, reliability, usability, scalability, security, and other non-functional aspects that influence the overall quality and user experience. Non-functional testing helps ensure that the software system meets the desired standards and performance expectations.
* **7 Types** of Non-functional testing are:

1. **Usability Testing**
2. **Compatibility Testing**
3. **GUI Testing**
4. **Security Testing**
5. **Performance Testing**
6. **Stress Testing**
7. **Load Testing**

* **What is GUI Testing?**
* **Graphical User Interface (GUI) testing**
* 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.
* The Graphical user interface testing process implemented either manually or automatically and repeatedly executed by the third-parties organization instead of the developers or the end-users
* Graphical User Interface (GUI) Testing Techniques/Methods :



* **What is Adhoc testing?**
* Adhoc testing, also known as **exploratory testing** is an informal and unscripted software testing approach that involves testing without any predefined test cases or plans.
* It is a technique where testers use their domain knowledge, experience, and intuition to perform on-the-spot testing based on their understanding of the system under test.
* In adhoc testing, testers are not bound by any specific test scenarios or scripts. Instead, they interact with the software application in an unplanned and spontaneous manner, exploring various features, functionalities, and user interactions.
* The primary objective of adhoc testing is to uncover defects, bugs, or usability issues that might not be easily detected through formal test cases.
* **What is load testing?**
* Load testing is a software testing technique that **focuses on evaluating the performance and behavior of a system under expected and anticipated high load conditions.**
* It involves simulating realistic user loads and measuring the system's response to determine its ability to handle the anticipated workload.
* **The primary goal of load testing is to identify performance bottlenecks**, determine the system's capacity limits, and ensure that it can handle the expected number of concurrent users, transactions, or requests without degradation in performance. By subjecting the system to various load levels, testers can analyze its behavior, response times, resource utilization, and overall stability.
* Load testing is crucial for ensuring that a system can handle the expected user load without performance degradation or failures. By proactively identifying performance bottlenecks and optimizing the system, organizations can deliver a high-quality, reliable, and responsive software application to their users.
* **What is stress Testing?**
* Stress testing is a software testing technique that **aims to evaluate the stability, robustness, and reliability of a system under extreme and unfavorable conditions.**
* It involves subjecting the system to high loads, excessive data volumes, or resource limitations to identify its breaking point and assess its behaviour under stress.
* The **primary goal of stress testing is to determine how the system handles and recovers from adverse conditions,** such as high user loads, data spikes, insufficient resources, or network failures. By pushing the system to its limits, testers can uncover weaknesses, bottlenecks, and potential failure points that may impact its performance, stability, or data integrity.
* Stress testing **helps organizations identify and address system weaknesses before them impact real users or critical business operations.** By subjecting the system to extreme conditions, it ensures that the software application can handle unexpected stress factors and continues to perform reliably under adverse circumstances.
* **What is white box testing and list the types of white box testing?**
* White box testing, also **known as clear box testing or structural testing**, is a software testing technique that **focuses on examining the internal structure, design, and implementation details of a software application.**
* It involves testing the code, logic, and paths within the system to ensure that they function as intended and meet the specified requirements.
* White box testing **requires knowledge of the internal workings of the system and access to the source code.**
* Testers typically have a detailed understanding of the system's architecture, algorithms, and programming languages to effectively design and execute white box tests.
* **List of white box testing** :

**1. Statement Coverage Testing**

**2. Decision coverage Testing**

**3. Condition Coverage Testing**

* Advantages:

1. It creates additional test cases to increase coverage
2. It helps in finding areas of a program not exercised by a set of test cases

* Disadvantages:

1. code coverage measurement is that it measures coverage of what has been written, i.e. the code itself; it cannot say anything about the software that has not been written.
2. If a specified function has not been implemented or a function was omitted from the specification, then structure-based techniques cannot say anything about them it only looks at a structure which is already there.

* **What is black box testing? What are the different black box testing techniques?**
* Black box testing is a software testing technique that **focuses on evaluating the functionality and behaviour of a software application without examining its internal structure or implementation details.**
* Testers treat the system as a "black box" and test it based on its specified inputs, expected outputs, and the system's external behaviour, without knowledge of its internal workings.
* Black box testing is primarily concerned with validating whether the software application meets the specified requirements and performs as expected from the end-user's perspective. Testers do not have access to the source code and rely solely on the system's inputs and outputs to design and execute tests.
* There are four specification-based or black-box techniques:

1. Equivalence partitioning
2. Boundary value analysis
3. Decision tables
4. State transition testing
5. Use-case Testing
6. Other Black Box Testing
7. Syntax or Pattern Testing

* Advantages

1. Well suited and efficient for large code segments.
2. Code Access not required.

* Disadvantages:

1. Limited Coverage since only a selected number of test scenarios are actually performed.
2. Inefficient testing, due to the fact that the tester only has limited knowledge about an application.
3. The test cases are difficult to design.

* **Mention what are the categories of defects?**
* There are 5 types of defects which are hear under:-

1. Data Quality/Database Defects: Deals with improper handling of data in the database.
2. Critical Functionality Defects: The occurrence of these bugs hampers the crucial functionality of the application.
3. Functionality Defects: These defects affect the functionality of the application.
4. Security Defects: Application security defects generally involve improper handling of data sent from the user to the application. These defects are the most severe and given highest priority for a fix.
5. User Interface Defects: As the name suggests, the bugs deal with problems related to UI are usually considered less severe.

* **Mention what bigbang testing is?**
* Big Bang testing is a software testing **approach where all the individual components or modules of a software system are tested together as a whole.**
* It is a type of integration testing method that involves combining all the modules at once and executing tests on the integrated system.
* This approach is in contrast to incremental testing, where modules are tested individually and then gradually integrated and tested in a step-by-step manner.
* In the Big Bang testing approach, the focus is on verifying the interactions and dependencies among different modules when they are integrated.
* The purpose is to identify any issues or defects that may arise due to the integration of modules and to ensure that the system functions correctly as a whole.
* Overall, Big Bang testing is considered a high-risk approach and is typically used in situations where the system's complexity and dependencies are relatively low, and time constraints are tight.
* **What is the purpose of exit criteria?**
* Exit criteria, also known as **test completion criteria**, are predefined conditions or metrics that must be met in order to determine when to stop testing activities for a particular phase, level, or project.
* The purpose of exit criteria is to provide a set of guidelines or benchmarks to evaluate whether testing has been performed adequately and whether the software system is ready to progress to the next phase or be released.
* Here are the main 5 purposes of exit criteria in software testing:

1. **Evaluation of Test Completion:** Exit criteria **help in determining when testing activities for a specific phase or level are considered complete**. By defining specific conditions that need to be met, such as achieving a certain level of test coverage or executing a specified number of test cases, exit criteria enable testers and stakeholders to assess the progress and completeness of testing efforts.
2. **Quality Assessment:** Exit criteria **serve as quality gates for evaluating the overall quality of the software system**. They help in measuring the level of defects, severity of issues, and other quality indicators. Meeting the exit criteria demonstrates that the software system meets the predefined quality standards and is ready for the next phase or release.
3. **Risk Management:** Exit criteria **aid in managing risks associated with software testing.** They help identify and assess the risks that need to be mitigated before proceeding further. Exit criteria may include requirements such as achieving a certain level of test coverage in critical areas or resolving high-priority defects that pose significant risks to the system.
4. **Decision Making:** Exit criteria **provide a basis for making informed decisions about software release or progression to the next phase**. By establishing clear and measurable criteria, stakeholders can make decisions on whether the system is stable enough, meets the required quality standards, and is ready for the next set of activities, such as user acceptance testing or deployment.
5. **Resource Allocation:** Exit criteria assist in **optimizing resource allocation by ensuring that testing activities are not prolonged indefinitely**. Once the exit criteria are met, testing resources can be efficiently redirected to other critical tasks, maximizing the efficiency of the testing process.

* **When should "Regression Testing" be performed?**

1. Change in requirements and code is modified according to the requirement
2. Defect fixing
3. Performance issue fix
4. New feature is added to the software

* **What is 7 key principles ? Explain in detail ?**

1. Testing Shows the Presence of Defects:

* This principle highlights that testing is not aimed at proving the absence of defects but rather at demonstrating their existence. Testing helps in identifying defects, inconsistencies, and deviations from expected behaviour. By executing tests, defects are uncovered, enabling their identification, analysis, and subsequent correction.

2. Exhaustive Testing is Impossible:

* It is practically impossible to test a software system under all possible conditions and scenarios. The principle acknowledges that complete or exhaustive testing, which covers every possible input combination and usage scenario, is unattainable. Instead, testing efforts should be focused on achieving optimal risk coverage within the available resources and time constraints.

3. Early Testing:

* The principle emphasizes the importance of initiating testing activities as early as possible in the software development life cycle. Testing should not be limited to the final stages of development but should start early, ideally in parallel with requirements gathering, design, and coding. Early testing enables the early detection and resolution of defects, reducing the cost and impact of rework.

4. Defect Clustering:

* This principle states that defects tend to cluster or concentrate in specific areas or components of a software system. Not all parts of the system are equally error-prone. By analyzing defect patterns and trends, testers can focus their testing efforts on high-risk areas, modules, or functionalities, thereby optimizing test coverage and defect detection.

5. Pesticide Paradox:

* The pesticide paradox principle suggests that if the same set of tests is repeatedly executed, over time, they become less effective at finding new defects. This is analogous to the idea that pests can become resistant to a particular pesticide with continuous exposure. To overcome this paradox, testing needs to evolve and adapt by periodically reviewing and refreshing test cases to ensure their effectiveness.

6. Testing is Context Dependent:

* Testing effectiveness is influenced by various contextual factors, including the nature of the software, project constraints, stakeholder expectations, available resources, and time limitations. The principle recognizes that testing approaches, techniques, and priorities should be adapted and tailored based on the specific context of the project.

7. Absence-of-Errors Fallacy:

* The principle cautions against assuming that the absence of detected defects implies the absence of all defects. Testing can never guarantee that a software system is completely defect-free. It is important to understand that testing helps in increasing confidence in the system's behaviour, identifying known defects, and reducing risks. However, it does not eliminate the possibility of unknown or unforeseen defects.
* **Difference between QA v/s QC v/s Tester :**

|  |  |  |  |
| --- | --- | --- | --- |
| **SN.** | **Quality Assurance** | **Quality Control** | **Tester** |
| 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. |

* **Difference between Smoke and Sanity?**

|  |  |  |
| --- | --- | --- |
| **SN.** | **Smoke Testing** | **Sanity Testing** |
| 1 | Smoke Testing Verifies the critical functionality | Sanity Testing Verifies the New-functionality |
| 2 | The objective of this testing is to verify "stability" of the system in order to with more rigorous testing | The objective of the testing is to verify the "rationality" of the system in order proceed to proceed with more rigorous testing |
| 3 | This testing is performed by the developers or testers. | Sanity testing is usually performed by testers. |
| 4 | Smoke testing is usually documented or scripted. | Sanity testing is usually not documented and is unscripted. |
| 5 | Smoke testing is a subset of Regression testing. | Sanity testing is a subset of Acceptance testing. |
| 6 | Smoke testing exercises the entire system from end to end. | Sanity testing exercises only the particular component of the entire system. |
| 7 | Smoke testing is like General Health Check-up. | Sanity Testing is like specialized health check-up. |

* **Difference between verification and Validation.**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Verification** | **Validation** |
| **Definition** | The process of evaluating work-products (not the actual final product) of a development phase to determine whether they meet the specified requirements for that phase. | The process of evaluating software during or at the end of the development process to determine whether it satisfies specified business requirements. |
| **Objective** | To ensure that the product is being built according to the requirements and design specifications. In other words, to ensure that work products meet their specified requirements | To ensure that the product actually meets the user’s needs, and that the specifications were correct in the first place. In other words, to demonstrate that the product fulfils its intended use when placed in its intended environment. |
| **Question** | Are we building the product right? | Are we building the right product? |
| **Evaluation Items** | Plans, Requirement Specs, Design Specs, Code, Test Cases | The actual product/software. |
| **Activities** | Reviews Walkthroughs Inspections | Testing |

* **Explain types of Performance testing.**
* Performance testing is a type of software testing that **focuses on evaluating the performance characteristics of a system under specific workloads** or conditions.
* It helps identify performance bottlenecks, measure response times, throughput, and scalability, and ensure that the system meets the required performance criteria:

1. Load Testing:

* Load testing involves simulating realistic user loads on the system to measure its behaviour and performance under normal and peak usage conditions. It helps determine how the system handles concurrent user requests, assess response times, and identify performance degradation or bottlenecks. Load testing ensures that the system can handle the expected user load without significant performance issues.

2. Stress Testing:

* Stress testing involves testing the system beyond its normal operational capacity to evaluate its behaviour and performance under extreme or unusual conditions. It aims to identify the system's breaking point and understand how it recovers from failure. Stress testing typically involves increasing the load, data volume, or transaction complexity to push the system's limits. It helps assess the system's stability, robustness, and resilience.

3. Soak Testing:

* Soak testing, also known as endurance testing, involves subjecting the system to a sustained workload for an extended duration to evaluate its performance and stability over time. It aims to identify any performance degradation or issues that may occur due to long-duration usage, resource leaks, memory leaks, or other factors. Soak testing helps ensure that the system remains stable and performs well over extended periods, such as continuous usage or overnight processing.

4. Spike Testing:

* Spike testing involves testing the system's performance when it experiences sudden and significant increases or spikes in user load or transaction volume. It helps assess how the system handles sudden surges and if it can scale up to meet the increased demand. Spike testing can reveal issues related to resource allocation, concurrency, and response times during high-traffic situations.

5. Scalability Testing:

* Scalability testing is performed to assess the system's ability to handle increased workload or user load by adding more resources such as servers, CPUs, or memory. It helps determine how the system scales and maintains performance as the workload grows. Scalability testing is essential for systems that are expected to handle increasing user demand or for applications that need to support future growth.

6. Volume Testing:

* Volume testing involves testing the system's performance with a large amount of data to assess its behaviour and response times under such conditions. It helps identify how the system handles data storage, retrieval, and processing. Volume testing can uncover performance issues related to database performance, disk space, caching, and data handling algorithms.

7. Configuration Testing:

* Configuration testing focuses on testing the system's performance under different hardware or software configurations. It aims to determine how different configurations impact the system's performance and to identify any configuration-specific performance issues. Configuration testing helps ensure that the system can perform optimally across a range of hardware, operating systems, browsers, and network configurations.
* **What is Error, Defect, Bug and failure?**

1. Error

* An error, also known as a **mistake or a fault**, is a human action or a misconception that produces an incorrect or unintended result. Errors are made during the software development process and can occur at any stage, including requirements gathering, design, coding, or documentation. Errors can lead to defects in the software.

2. Defect

* **A defect, also known as a fault or a bug**, is a flaw or deviation in the software that causes it to behave in an unintended or incorrect way. **Defects are the result of errors made during software development or maintenance.** They can manifest **as coding mistakes, logic errors, missing or inaccurate functionality, or any other deviation from the expected behaviour.** Defects can exist in various forms, such as syntax errors, logic errors, usability issues, performance issues, or security vulnerabilities.

3. Bug

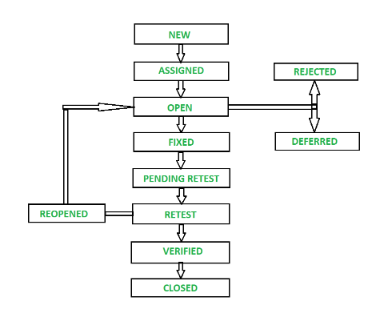
* The term **"bug" is often used interchangeably with "defect**." Historically, the term "bug" was first coined by Grace Hopper when she found an actual moth causing a malfunction in a computer system. Today, "bug" is a colloquial term used to refer to defects or flaws in software. In practice, there is no significant difference between a bug and a defect—they both represent flaws in the software's behaviour.

4. Failure:

* A failure occurs **when the software does not behave as expected or does not meet the specified requirements or user expectations**. It is the manifestation of a defect during the execution of the software. **Failures can result from one or multiple defects or a combination of system-related factors.** Failures can include crashes, incorrect outputs, system hangs, performance issues, or any other deviation from the expected behaviour. Failures are typically observed by end-users or testers during the testing or operational phases of the software.
* **Difference between Priority and Severity**

|  |  |  |
| --- | --- | --- |
| **SN.** | **Priority** | **Severity** |
| **1** | 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. |
| **2** | Priority means how **fast defect has to be fixed.** | Severity means how **severe defect is affecting the functionality.** |
| **3** | Priority is related to **scheduling to resolve the problem.** | Severity is related to **the quality standard**. |
| **3** | **Product manager** **decides** the priorities of defects. | **Testing engineer decides** the severity level of the defect. |
| **5** | Its value is **subjective.** | Its value is **objective.** |
| **6** | Priority is of **3** types:  Low  Medium  High | Severity is of **5** types:  Critical  Major  Moderate  Minor  Cosmetic |

* **What is Bug Life Cycle?**
* A *defect* is an *error* or a *bug*, in the application which is created.
* A programmer while designing and building the software can make *mistakes* or *errors*. These mistakes or errors mean that there are *flaws* in the software. These are called *defects*.
* The duration or time span between the first time defects is found and the time that it is closed successfully, rejected, postponed or deferred is called as ‘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.

* **Explain the difference between Functional testing and Non-Functional testing:**

|  |  |  |
| --- | --- | --- |
| **SN.** | **Functional Testing** | **Non-Functional Testing** |
| **1** | It verifies the operations and actions of an application. | It verifies the behavior of an application. |
| **2** | It is based on requirements of customer. | It is based on expectations of customer. |
| **3** | It helps to enhance the behavior of the application. | It helps to improve the performance of the application. |
| **4** | Functional testing is easy to execute manually. | It is hard to execute non-functional testing manually. |
| **5** | Examples:  1. Unit Testing  2. Smoke Testing  3. Integration Testing  4. Regression Testing | Examples:  1. Performance Testing  2. Load Testing  3. Stress Testing  4. Scalability Testing |

* **To create HLR & Test Case of**

1. **Instagram only first page :**

* [**https://github.com/Foram-77/Foram\_Testing\_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Instagram%20First%20Page%20HLR%20And%20Test%20Case.xlsx**](https://github.com/Foram-77/Foram_Testing_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Instagram%20First%20Page%20HLR%20And%20Test%20Case.xlsx)

1. **Facebook First page :**

* [**https://github.com/Foram-77/Foram\_Testing\_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Facebook%20First%20Page%20HLR%20And%20Test%20Case.xlsx**](https://github.com/Foram-77/Foram_Testing_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Facebook%20First%20Page%20HLR%20And%20Test%20Case.xlsx)
* **What is the difference between the STLC (Software Testing Life Cycle) and SDLC (Software Development Life Cycle)?**

|  |  |  |
| --- | --- | --- |
| **SN.** | **STLC**  **(Software Testing Life Cycle)** | **SDLC**  **(Software Development Life Cycle)** |
| **1** | STLC is mainly related to software testing. | SDLC is mainly related to software development. |
| **2** | In STLC, testing team(Test Lead or Test Architect) makes the plans and designs. | In SDLC, development team makes the plans and designs based on the requirements. |
| **3** | It helps in making the software defects free. | It helps in developing good quality software. |
| **4** | STLC phases are performed after SDLC phases. | SDLC phases are completed before the STLC phases. |
| **5** | STLC Phases :   1. Requirement Analysis 2. Test Planning 3. Test case development 4. Test Environment setup 5. Test Execution 6. Test Cycle closure | SDLC Phases :   1. Requirements Collection/Gathering 2. Analysis 3. Design 4. Implementation 5. Testing 6. Maintenance |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **SN.** | **Test scenarios** | **Test cases** | **Test script** |
| **1** | A test scenario is a high-level document that defines the functionality to be tested from beginning to end. | A test case is a high-level document with instructions on the specific functionality of the software product to be tested. | Test Script is a step-by-step instruction to test each software product’s functionality (test case). |
| **2** | It emphasizes "what to test" rather than "how to test." | A test case is the software development life cycle’s ‘What to test’ component. | Test script is the software development life cycle’s ‘How to test’ component. |
| **3** | BRS, SRS, and other test artifacts are used to create test scenarios. | Test cases are written in simple English. | Test scripts are written in programming languages like VB Script, Python, Java, etc. |
| **4** | It aids in the rapid testing of end-to-end functionality. | Test scenarios serve as an outline for writing test cases. | Test Case serves as an outline for writing test scripts. |

* **Explain what Test Plan is? What is the information that should be covered.**
* A document describing the scope, approach, resources and schedule of intended test activities Determining the scope and risks, and identifying the objectives of testing. Defining the overall approach of testing (the test strategy), including the definition of the test levels and entry and exit criteria.
* Test Planning activities:

**Approach**: Defining the overall approach of testing (the test strategy), including the definition of the test levels and entry and exit criteria.

**Integrating and coordinating the testing activities into the software life cycle activities:** acquisition, supply, development, operation and maintenance. Making decisions about:

**What to test**

**Who does test?** i.e. what roles will perform the test activities

**When and how** the test activities should be done and when they should be stopped (exit criteria – see next slides)

**How** **the test** results will be evaluated Assigning resources for the different tasks defined.

**Test ware:** Defining the amount, level of detail, structure and templates for the test documentation. Selecting metrics for monitoring and controlling test preparation and execution, defect resolution and risk issues.

**Process:** Setting the level of detail for test procedures in order to provide enough information to support reproducible test preparation and execution.

* **What is priority?**
* Priority **is Relative and Business-Focused**. Priority defines the order in which we should resolve a defect. Should we fix it now, or can it wait? This priority status is set by the tester to the developer mentioning the time frame to fix the defect. If high priority is mentioned then the developer has to fix it at the earliest. The priority status is set based on the customer requirements.
* E.g: If the company name is misspelled in the home page of the website, then the priority is high and severity is low to fix it.
* Priority can be of following types:

1. **Low:** The defect is an irritant which should be repaired, but repair can be deferred until after more serious defect has been fixed.
2. **Medium:** The defect should be resolved in the normal course of development activities. It can wait until a new build or version is created.
3. **High:** The defect must be resolved as soon as possible because the defect is affecting the application or the product severely. The system cannot be used until the repair has been done.
4. **Critical:** Extremely urgent, resolve immediately

* **What is severity?**
* Severity refers to the degree of impact or **seriousness of a defect or issue on the functionality, performance**, or usability of a software application or system. It is a measure of how severe or critical the issue is and helps in prioritizing and allocating resources for its resolution.
* When a defect is identified during testing or reported by users, it is assigned a severity level to indicate the impact it has on the software. The severity level is
* typically assigned based on the following scale or categories:

1. **Critical Severity:** Defects that cause complete failure or shutdown of the system, preventing it from performing its intended function. These issues result in the system being unusable or inoperable and require immediate attention.
2. **High Severity:** Defects that have a significant impact on the functionality or performance of the system, resulting in major errors or limitations. These issues affect critical features or workflows and may severely impact the user experience or prevent the software from meeting its intended purpose.
3. **Medium Severity:** Defects that have a moderate impact on the system's functionality or performance. These issues may result in errors or limitations in specific scenarios or features but do not cause complete failure or significantly hinder the system's core functionality.
4. **Low Severity:** Defects that have a minor or negligible impact on the system's functionality or performance. These issues may include cosmetic defects, minor inconveniences, or non-critical features that are not essential for the basic functioning of the software.

* **Advantage of Bugzila :**
* Bugzilla is an **open-source issue/bug tracking system** that allows developers effectively to keep track of outstanding problems with their product. It is written in Perl and uses MYSQL database.
* Bugzilla is a defect tracking tool, however it can be used as a test management tool as such it can be easily linked with other test case management tools like Quality Centre, Test link etc.
* **Advantages :**

1. It improves the quality of the product.
2. It enhances the communication between the developing team and the testing team.
3. It has the ability to adapt to multiple situations.
4. Automatic duplicate bug detection.
5. Modify/file Bugs by e-mail
6. Time tracking
7. Ideal for small projects.

* **What are the different Methodologies in Agile Development Model?**
* The Agile methodology is a way **to manage a project by breaking it up into several phases.** It involves constant collaboration with stakeholders and continuous improvement at every stage. Once the work begins, teams’ cycle through a process of planning, executing, and evaluating.
* Agile is **based on the iterative-incremental model.** In an incremental model, we create the system in increments, where each increment is developed and tested individually.
* **Agile four manifesto:**

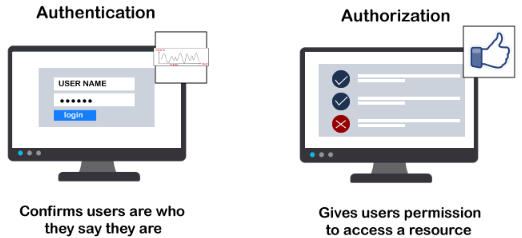
1. **Individuals and interactions**, Over processes and tools
2. **Working software**, over comprehensive documentation
3. **Customer collaboration**, Over contract negotiation
4. **Responding to change**, over following a plan

* The different Methodologies in Agile Development Model

1. Scrum
2. Kanban
3. DSDM (Dynamic Software Development Method)
4. FDD (Feature Driven Development)

* **Explain the difference between Authorization and Authentication in Web testing.**

**What are the common problems faced in Web testing?**

****

|  |  |  |
| --- | --- | --- |
| **SN.** | **Authorization** | **Authentication** |
| **1** | While in authorization process, a the person’s or user’s authorities are checked for accessing the resources | In the authentication process, the identity of users are checked for providing the access to the system. |
| **2** | While in this process, users or persons are validated. | In the authentication process, users or persons are verified. |
| **3** | While this process is done after the authentication process. | It is done before the authorization process |
| **4** | While it needs the user’s privilege or security levels. | It needs usually the user’s login details. |
| **5** | Generally, transmit information through an Access Token. | Generally, transmit information through an ID Token |
| **6** | The user authorization is not visible at the user end. | The user authentication is visible at user end. |
| **7** | The user authorization is carried out through the access rights to resources by using roles that have been pre-defined. | The user authentication is identified with username, password, face recognition, retina scan, fingerprints, etc. |

**common problems faced in Web testing :**

**1. Broken Links:** Links within a web application can become broken or dead over time, leading to a poor user experience. Web testing involves checking for broken links and ensuring that all links navigate correctly.

**2. Cross-Browser Compatibility:** Web applications need to be tested across different browsers and their versions to ensure consistent functionality and appearance. Incompatibilities may arise due to variations in rendering engines or browser-specific features.

**3. Performance Issues:** Web applications need to handle a large number of concurrent users efficiently. Testing for performance issues such as slow response times, bottlenecks, and scalability problems is crucial to ensure optimal user experience.

**4. Security Vulnerabilities:** Web applications are often targeted by malicious actors. Web testing involves identifying and addressing common security vulnerabilities, such as cross-site scripting (XSS), cross-site request forgery (CSRF), SQL injection, and insecure session management.

**5. Usability Testing:** Usability testing focuses on evaluating the user-friendliness of a web application. It involves assessing the application's intuitiveness, ease of navigation, clarity of instructions, and overall user satisfaction.

**6. Data Validation:** Input validation is critical to prevent security vulnerabilities and data integrity issues. Web testing involves checking for proper validation of user input to prevent malicious input or unintended errors.

**7. Compatibility with Different Devices:** With the wide variety of devices used to access the web, including smart phones, tablets, and desktops, web testing should include testing on various screen sizes, resolutions, and operating systems to ensure compatibility and responsiveness.

* **To create HLR & TestCase of WebBased (WhatsApp web , Instagram)** 
  1. **WhatsApp Web :**
* **https://github.com/Foram-77/Foram\_Testing\_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/whatsapp%20web%20HLR%20And%20Test%20Case.xlsx**
  1. **Instagram web :**
* [**https://github.com/Foram-77/Foram\_Testing\_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Instagram%20HLR%20And%20Tese%20case.xlsx**](https://github.com/Foram-77/Foram_Testing_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Instagram%20HLR%20And%20Tese%20case.xlsx)
* **To create HLR and TestCase on artoftesting :**
* **https://github.com/Foram-77/Foram\_Testing\_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Artoftesting%20HLR%20and%20Test%20Case.xlsx**

* **Write a scenario of only Whatsapp chat messages :**
* [**https://github.com/Foram-77/Foram\_Testing\_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Scenario%20of%20Whatsapp%20Chat.xlsx**](https://github.com/Foram-77/Foram_Testing_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Scenario%20of%20Whatsapp%20Chat.xlsx)
* **Write a Scenario of Pen** **:**
* [**https://github.com/Foram-77/Foram\_Testing\_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Test%20Scenario%20of%20R%20F%20C%20P%20B%20D.xlsx**](https://github.com/Foram-77/Foram_Testing_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Test%20Scenario%20of%20R%20F%20C%20P%20B%20D.xlsx)
* **Write a Scenario of Pen Stand :**
* **https://github.com/Foram-77/Foram\_Testing\_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Scenario%20of%20Pen%20Stand.xlsx**
* **Write a Scenario of Door** :
* **https://github.com/Foram-77/Foram\_Testing\_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Test%20Scenario%20of%20R%20F%20C%20P%20B%20D.xlsx**
* **Write a Scenario of ATM** **:**
* [**https://github.com/Foram-77/Foram\_Testing\_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Scenario%20of%20ATM.xlsx**](https://github.com/Foram-77/Foram_Testing_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Scenario%20of%20ATM.xlsx)
* **When to used Usablity Testing?**
* Usability testing is **typically used during the design and development process of a product or service to evaluate its usability.** It **involves** **observing and gathering feedback from users as they interact with a prototype** or a finished product to identify usability issues and make informed design decisions.
* specific scenarios when usability testing is commonly used:

1. **Early design stages:**

* Usability testing can be conducted during the initial stages of the design process to gather user feedback on concept ideas, wireframes, or low-fidelity prototypes. This helps in validating design assumptions, identifying potential usability problems, and informing iterative design improvements.

1. **Prototype evaluation:**

* Usability testing is useful when testing interactive prototypes that simulate the functionality of a product before its full development. It allows designers and developers to observe how users navigate through the prototype, interact with its features, and gather insights on user expectations and preferences.

1. **Iterative design:**

* Usability testing is often conducted throughout the iterative design process to validate design changes and improvements. By testing and collecting feedback on successive versions of a product, designers can refine and optimize the user experience based on user insights and preferences.

1. **Comparative evaluation:**

* Usability testing can be used to compare different design options or variants. By testing multiple designs with users, it helps in identifying the strengths and weaknesses of each design option, making informed decisions about which design direction to pursue.

**5. New feature assessment:**

* When introducing new features or significant updates to an existing product, usability testing can gauge how well users understand and utilize the new functionality. This ensures that the new features align with user needs and expectations and do not introduce any usability problems.

1. **Post-launch evaluation:**

* Usability testing can be conducted after the product is launched to gather user feedback, identify areas of improvement, and prioritize future updates or enhancements. This helps in continuously enhancing the user experience and maintaining a user-centric approach.
* **What is the procedure for GUI Testing?**

**Manual Testing:**

* This approach involves human tester, where each screen is manually checked to validate each functionality by creating and executing test cases. It is a useful approach when part of UI or a feature is ready, the probability of defects is more at the initial stage, and human intervention is required.

**Record and Replay Testing:**

* GUI record and replay tools are used to test applications for their user interface. Using such tools, testers run an application and record the user interaction with the app. A script runs to track and save the user actions, including cursor movements, which can be replayed several times to find the issues in the interface.

**Model-based testing:**

* In this type of GUI testing, a model is created to understand and evaluate the system’s behavior. This approach is useful in creating accurate test cases using system requirements.

**There are three essential aspects of model-based GUI testing:**

1. Automatically generated test cases from the model
2. Manually derived test cases from the model
3. Model and requirements coverage metrics

**Things to consider for model-based testing:**

1. Create the model
2. Determine the information as inputs in the system
3. Verifying the expected output
4. Execute tests
5. Checking and validating actual vs. expected
6. Take further action on the model

* **Write a scenario of Microwave Owen :**
* **https://github.com/Foram-77/Foram\_Testing\_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Scenario%20of%20Maicrowev%20Oven.xlsx**
* **Write a scenario of Coffee vending Machine :**
* **https://github.com/Foram-77/Foram\_Testing\_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Scenario%20of%20Coffee%20Vending%20Machine.xlsx**
* **Write a scenario of chair :**
* **https://github.com/Foram-77/Foram\_Testing\_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Test%20Scenario%20of%20R%20F%20C%20P%20B%20D.xlsx**
* **To Create Scenario (Positive & Negative)**

1. **Facebook Chat on Mobile :**

* **https://github.com/Foram-77/Foram\_Testing\_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Scenario%20of%20Facebook%20Chat%20on%20Mobile.xlsx**

1. **Gmail (Receiving Mail) :**

* **https://github.com/Foram-77/Foram\_Testing\_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Scenario%20of%20Gmail%20(Reciveing%20Mail).xlsx**

1. **Online shopping to buy product (flipkart) :**

* **https://github.com/Foram-77/Foram\_Testing\_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Scenario%20of%20Flipkart%20Product%20Purchase.xlsx**
* **Write a Scenario of Wrist Watch :**
* **https://github.com/Foram-77/Foram\_Testing\_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Scenario%20of%20Wrist%20Watch.xlsx**
* **Write a Scenario of Lift (Elevator) :**
* [**https://github.com/Foram-77/Foram\_Testing\_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Scenario%20of%20Elivetor.xlsx**](https://github.com/Foram-77/Foram_Testing_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Scenario%20of%20Elivetor.xlsx)
* **Write a Scenario of whatsapp Group (generate group) :**
* **https://github.com/Foram-77/Foram\_Testing\_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Scenario%20of%20Whtsapp%20Group.xlsx**
* **Write a Scenario of instagram ( video call with chat ) :**
* **https://github.com/Foram-77/Foram\_Testing\_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Scenario%20of%20Instagram%20(Call%20With%20Chat).xlsx**
* **Write a Scenario of Whatsapp payment :**
* **https://github.com/Foram-77/Foram\_Testing\_Tops/blob/main/Assignments/Module%202%20HLR%20and%20Test%20Case/Scenario%20of%20Whatsapp%20Payment.xlsx**