# Module – 2

# (Manual Testing)

# 

1. **What is Exploratory Testing?**

* Exploratory testing is an approach to software testing that is often described as simultaneous learning, test design, and execution.
* It focuses on discovery and relies on the guidance of the individual tester to uncover defects that are not easily covered in the scope of other tests.
* In exploratory testing tester focuses more on how the software actually works, testers do minimum planning and maximum execution of the software by which they get in depth idea about the software functionality, once the tester starts getting insight into the software, he can make decisions to what to test next.
* Exploratory testing is mostly used if the requirements are incomplete and time to release the software is less.

1. **What is traceability matrix?**

* Traceability Matrix (also known as Requirement Traceability Matrix - RTM) is a table which is used to trace the requirements during the Software development life Cycle.
* It can be used for forward tracing or backward.
* A traceability matrix is a document that details the technical requirements for a given test scenario and its current state.
* It helps the testing team understand the level of testing that is done for a given product.
* The traceability process itself is used to review the test cases that were defined for any requirement.
* The traceability matrix document is prepared to show clients that the coverage is complete.
* It usually includes the following columns: requirement, baseline document reference number, test case/condition and defect/bug ID.
* Using this document the person can track the Requirement based on the Defect id.
* **Types of Traceability Matrices:**

1. **Forward Traceability:** Mapping of requirements to test cases.
2. **Backward Traceability:** Mapping of test cases to requirements.
3. **Bi-Directional Traceability:** A good example of a bi-directional traceability matrix used in software testing is the references from test cases to basis documentation and vice versa.
4. **What is Boundary value testing? **

* Software testing technique in which tests are designed to include representatives of boundary values. It is performed by the QA testing teams.
* Boundary Value Analysis (BVA) is a Black-Box testing technique used to check the errors at the boundaries of an input domain.
* The name comes from the Boundary, which means the limits of an area.
* So, BVA mainly focuses on testing both valid and invalid input parameters for a given range of a software component.

1. **What is Equivalence partitioning testing?**

* Equivalence partitioning testing is software testing technique that divides the input data of a software unit into partitions of data from which test cases can be derived. it is usually performed by the QA teams.
* Equivalence Partitioning Method is also known as Equivalence class partitioning (ECP).
* It is a software testing technique or black-box testing that divide input domain into classes of data, and with the help of these classes of data, test cases can be derived.
* An ideal test case identifies class of error that might require many arbitrary test cases to be executed before general error is observed.
* Values from each partition must be tested at least once. Partitions with valid values are used for Positive Testing.
* While, partitions with invalid values are used for negative testing.

1. **What is Integration testing?**

* Integration testing is associated with the architectural design phase. Integration tests are performed to test the coexistence and communication of the internal modules within the system.
* Integration testing is a type of software testing in which the different units, modules or components of a software application are tested as a combined entity.
* Integration Testing is a level of the software testing process where individual units are combined and tested as a group.
* Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems.
* Integration testing is done by a specific integration tester or test team.
* Components may be code modules, operating systems, hardware and even complete systems.
* **There are 2 levels of Integration Testing**

1. **Component Integration Testing**
2. **System Integration Testing**
3. **What determines the level of risk?**

* Risk is determined by a combination of Probability and Severity; the main area of the Matrix reveals the Risk Levels.
* The levels are Low, Medium, High. To have a low level of risk, we must have a somewhat limited probability and level of severity.
* Risk should be evaluated at the Business level, Technological level, Project level and Testing level.

1. **What is Alpha testing?**

* Alpha Testing is performed by the Testers within the organization.
* Alpha Testing is performed in VIRTUAL ENVIRONMENT.
* Alpha Testing is performed by Independent Testing Team.
* Alpha Testing is conducted for the software application and project.
* Alpha Testing is performed at Developer’s site. (Within the organization)
* Alpha Testing is the form of Acceptance Testing.
* Reliability and Security testing are not performed in Alpha Testing.
* Alpha Testing involves both White box and Black box testing.
* Alpha Testing identifies possible errors.
* Alpha Testing requires testing environment.
* Alpha Testing requires long execution cycle.
* Critical issues and bugs are addressed and fixed immediately in Alpha Testing.
* Alpha Testing can be easily implemented as it is done before the near end of development.
* Alpha Testing is always performed at the time of Acceptance Testing when developers test the product and project to check whether it meets the user requirements or not.
* Alpha Testing is considered as the User Acceptance Testing (UAT) which is done at developer’s area.
* Alpha testing is to ensure the quality of the product before moving to Beta testing.
* Unit testing, integration testing and system testing when combined are known as Alpha Testing.

1. **What is beta testing?**

* Beta Testing is performed by the end users.
* Beta Testing is performed in REAL TIME ENVIRONMENT.
* Beta Testing is not performed by Independent Testing Team.
* Beta Testing is usually conducted for software product.
* Beta Testing is performed at Client’s location. (Outside the organization)
* Beta Testing is also the form of Acceptance Testing.
* Reliability, Security and Robustness are checked during Beta Testing.
* Beta Testing mainly involves Black box testing.
* Beta Testing checks the quality of the product.
* Beta Testing doesn’t require any lab environment or testing environment.
* Beta Testing requires only few weeks of execution.
* Issues and bugs are collected from the end users and further implemented in Beta Testing.
* Beta Testing will be implemented in the future version of the product.
* Beta Testing is always performed at the time when software product and project are marketed.
* Beta Testing is also considered as the User Acceptance Testing (UAT) which is done at customers or users’ area
* Beta testing also concentrates on the quality of the product, but gathers users input on the product and ensures that the product is ready for real time users.
* Pilot Testing is testing to product on real world as well as collect data on the use of product in the classroom.

1. **What is component testing?**

* Component testing is also known as Unit Testing.
* Component Testing is the testing of individual software components.
* Component is A minimal software item that can be tested in isolation. It means “A unit is the smallest testable part of software.”
* Component testing, also known as program or module testing, is done after unit testing. In this type of testing those test objects can be tested independently as a component without integrating with other components e.g., modules, classes, objects, and programs. This testing is done by the development team.
* Component testing is like unit testing with the difference that the developer uses real data instead of dummy data for testing of the written code.
* Unit testing is the first level of testing and is performed prior to Integration Testing.
* Sometimes known as Unit Testing, Module Testing or Program Testing.
* Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended with debugging tool.
* Unit testing is performed by using the White Box Testing method.
* Entry criteria for component testing:
* The minimum number of components that are included in the UT must be developed and tested through unit testing.
* **Exit criteria for component testing:**

1. the functionality of all the components must be working as expected;
2. there should be no critical, high, medium severity and priority defect/bug left in the software.
3. **What is functional system testing?**

* Functional Testing is a type of Software Testing in which the system is tested against the functional requirements and specifications.
* Functional testing ensures that the requirements or specifications are properly satisfied by the application.
* There is two types of techniques
* Requirement Based Functional Testing
* Process Based Testing
* **Types of Functional System Testing :**

1. **Unit Testing**
2. **Smoke Testing**
3. **Sanity Testing**
4. **Integration Testing**
5. **White box testing**
6. **Black Box testing**
7. **User Acceptance testing**
8. **Regression Testing**
9. **What is Non-Functional Testing?**

* Non-functional testing assesses application properties that aren't critical to functionality but contribute to the end-user experience.
* Non Functional testing has a goal to validate the performance of the software.
* Non-Functional testing checks the Performance, reliability, scalability and other non functional aspects of the software system.
* **Types of Nonfunctional testing are:**

1. **Performance Testing**
2. **Load Testing**
3. **Volume Testing**
4. **Stress Testing**
5. **Security Testing**
6. **Installation Testing**
7. **Penetration Testing**
8. **Compatibility Testing**
9. **Migration Testing**
10. **What is GUI Testing?**

* GUI testing is a testing technique in which the application's user interface is tested whether the application performs as expected with respect to user interface behaviour.
* GUI Testing includes the application behaviour towards keyboard and mouse movements and how different GUI objects such as tool bars, buttons, menu bars, dialog boxes, edit fields, lists, behavior to the user input.
* **GUI Testing Guidelines:**

1. Check Screen Validations
2. Verify All Navigations
3. Check usability Conditions
4. Verify Data Integrity
5. Verify the object states
6. Verify the date Field and Numeric Field Formats

* **Approach of GUI Testing:**

1. MANUAL BASED TESTING
2. RECORD AND REPLAY
3. MODEL BASED TESTING
4. **What is Adhoc testing?**

* Adhoc Testing is also known as Error Guessing.
* Adhoc testing is an informal testing type with an aim to break the system.
* When a software testing performed without proper planning and documentation, it is said to be Adhoc Testing.
* It does not follow any test design techniques to create test cases.
* Adhoc Testing does not follow any structured way of testing and it is randomly done on any part of application.
* Adhoc testing can be achieved with the testing technique called Error Guessing.
* Error guessing can be done by the people having enough experience on the system to
* “guess” the most likely source of errors.
* 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.
* There are different types of Adhoc testing and they are listed as below:

1. **Buddy Testing**

**2. Pair testing**

**3. Monkey Testing**

1. **What is load testing?**

* Load testing helps to determine the behavior of an application when large numbers of users access the application simultaneously.
* It is a type of performance testing which identifies the maximum operating capacity of an application.
* This helps to know whether the existing infrastructure is capable to run the application or not.
* Load testing is conducted to determine the number of concurrent users that an application can support without deterioration in performance.
* This testing is usually performed for client/server and web-based application.
* 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.
* Load testing will determine whether system needs to be fine-tuned or modification of hardware and software is required to improve performance
* Load Testing helps identify the bottlenecks in the system under heavy user stress scenarios before they happen in a production environment.

1. **What is stress Testing?**

* Stress Testing is a software testing technique that determines the robustness of software by testing beyond the limits of normal operation.
* Stress testing is particularly important for critical software but is used for all types of software.
* Stress testing emphasizes robustness, availability, and error handling under a heavy load rather than what is correct behavior under normal situations.
* Stress testing is defined as a type of software testing that verifies the stability and reliability of the system.
* This test particularly determines the system on its robustness and error handling under extremely heavy load conditions.
* It even tests beyond the normal operating point and analyses how the system works under extreme conditions.
* Stress testing is performed to ensure that the system would not crash under crunch situations.
* Stress testing is also known as Endurance Testing or Torture Testing.
* It also checks whether system demonstrates effective error management under extreme conditions.
* Example:
  + The application under testing will be stressed when 5GB data is copied from the website and pasted in notepad.
  + Notepad is under stress and gives ‘Not Responded’ error message.
  + During festival time, an online shopping site may witness a spike in traffic, or when it announces a sale.
* Types of Stress Testing

1. **Application Stress Testing**
2. **Transactional Stress Testing**
3. **Systemic Stress Testing**
4. **Exploratory Stress Testing**
5. **What is white box testing and list the types of white box testing?**

* White box testing is a testing technique, that examines the program structure and derives test data from the program logic/code.
* It is also called as glass box testing, clear box testing, open box testing, logic driven testing or path driven testing or structural testing.
* **Advantages of White Box Testing:**
* Code optimization by finding hidden errors.
  + - White box tests cases can be easily automated.
    - Testing is more thorough as all code paths are usually covered.
    - Testing can start early in SDLC even if GUI is not available.
* **Disadvantages of White Box Testing:**
* White box testing can be quite complex and expensive.
  + - Developers who usually execute white box test cases detest it. The white box testing by developers is not detailed can lead to production errors.
    - White box testing requires professional resources, with a detailed understanding of programming and implementation.
    - White-box testing is time-consuming, bigger programming applications take the time to test fully.
* **Types of code coverage:**
  + - Statement Coverage
    - Decision Coverage
    - Condition Coverage
* **Types of other White Box testing:**
  + - Branch Condition Testing
    - Branch Condition Combination Testing
    - Modified Condition Decision Testing
    - Data Flow Testing
    - Linear Code Specification And Jump Testing

1. **What is black box testing? What are the different black box testing techniques?**

* Black box Testing is either functional or non-functional, without reference to the internal structure of the component or system.
* Black Box Testing is a software testing method in which the functionalitys of software applications are tested without having knowledge of internal code structure, implementation details and internal paths.
* Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.
* In black-box testing, the tester is concentrating on what the software does, not how it does it.

Input

Output

* **Advantages :**
  + -  Well suited and efficient for large code segments.
    - Code Access not required.
    - Clearly separates user's perspective from the developer's perspective through visibly defined roles.
    - Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language or operating systems.
* **Disadvantages :**
  + - Limited Coverage since only a selected number of test scenarios are actually performed.
    - Inefficient testing, due to the fact that the tester only has limited knowledge about an application.
    - Blind Coverage, since the tester cannot target specific code segments or error prone areas.
    - The test cases are difficult to design.
* **There are four specification-based or black-box technique:**

**1.Equivalence partitioning**

**2. Boundary value analysis**

**3. Decision tables**

**4. State transition testing**

**5. Use-case Testing**

**6. Other Black Box Testing**

* **Syntax or Pattern Testing**

1. **Mention what are the categories of defects?**

* **Data Quality/Database Defects:** Deals with improper handling of data in the database.
* Examples:
* Values not deleted/inserted into the database properly
* Improper/wrong/null values inserted in place of the actual values
* **Critical Functionality Defects:** The occurrence of these bugs hampers the crucial functionality of the application.
* Examples: - Exceptions
* **Functionality Defects:** These defects affect the functionality of the application.
* Examples:
* All JavaScript errors
* Buttons like Save, Delete, Cancel not performing their intended functions
* A missing functionality (or) a feature not functioning the way it is intended to
* Continuous execution of loops
* **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.
* Examples:
* Authentication: Accepting an invalid username/password
* Authorization: Accessibility to pages though permission not given
* **User Interface Defects:** As the name suggests, the bugs deal with problems related to UI are usually considered less severe.
* Examples:
* Improper error/warning/UI messages
* Spelling mistakes
* Alignment problems

1. **Mention what big bang testing is?**

* Big Bang Integration Testing is an integration testing strategy wherein all units are linked at once, resulting in a complete system.
* When this type of testing strategy is adopted, it is difficult to isolate any errors found, because attention is not paid to verifying the interfaces across individual units.
* Big Bang integration testing all components or modules are integrated simultaneously, after which everything is tested as a whole.
* Big Bang integration testing all the modules are integrated without performing any integration testing and then it’s executed to know whether all the integrated modules are working fine or not.
* **Advantage of Big Bang Integration Testing:**

1. Big Bang testing has the advantage that everything is finished before integration testing starts. Disadvantages of Big Bang Integration Testing:

* **The major disadvantage is that in general it is very time consuming.**

1. It is very difficult to trace the cause of failures because of this late integration.
2. The chances of having critical failures are more because of integrating all the components together at same time.
3. If any bug is found then it is very difficult to detach all the modules in order to find out the root cause of it.
4. There is high probability of occurrence of the critical bugs in the production environment.
5. **What is the purpose of exit criteria?**

* Exit Criteria ensure that the testing of the application is completed and ready for release.
* Purpose of exit criteria is to define when we STOP testing either at the:
  + All the planned requirements must be met
  + All the high Priority bugs should be closed
  + All the test cases should be executed
  + End of all testing
  + End of phase of testing

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

* Regression testing should be performed whenever your code base has been modified or altered in any way as well as to verify any previously discovered issues marked as fixed.
* Regression testing will help your developers fix the reported defects on time, and your project to avoid any long-term pitfalls and technical debt caused by poor code quality.
* However, even though an occasional project might have the resources to perform the tests after the slightest changes have been introduced to the code base, for most projects designing and maintaining such a multiplicity of regression tests may simply be infeasible. Therefore, it is important to understand when you need to start regression testing.
* The most common reason to run regression tests is the introduction of new functionality.
* It is hard for developers to follow every thread in the code when modifying it, and there’s always a risk of compatibility issues with the existing code.
* Regression testing can save developers a lot of time with timely detection of bugs that would otherwise cause the project a lot of pain in the long run.
* Sometimes, however, sudden shifts in business strategy and requirements can lead to complete revision of the existing functionality, which requires developers to adjust, reshape, or even discard some of the features.
* Heavy interference with the source code can potentially cause a lot of damage to the remaining functionality, which makes regression testing an absolute must in such cases.
* Attempts at fixing one bug can at times turn into even more bugs appearing in code base areas you expect them the least.
* Debugging suggests making a lot of big and small changes to the source code, checking bug statuses, and going through the same process over again.
* This, in turn, points out the importance of following up the debugging stage with regression testing to guarantee it didn’t cause more issues than it fixed; to make sure everything works as intended after debugging.
* Last but not least, regression testing has proven to be an effective means of ensuring seamless and bug-free integration with external systems.

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

* **7 key Principles of Testing:**

**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 talks about the presence of defects and doesn’t talk about the absence of 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.
* In software testing, we look for bugs to be fixed before we deploy systems to live environments – this gives us confidence that our systems will work correctly when goes live to users.
* The testing process does not guarantee that software is 100% error-free.
* The probability of undiscovered defects remaining in a system.
* It is true that testing greatly reduces the number of defects buried in software, however discovering and repairing these problems does not guarantee a bug-free product or system.

**2. Exhaustive Testing is Impossible!**

* Testing everything including all combinations of inputs and preconditions is not possible.
* Exhaustive testing usually tests and verifies all functionality of a software application while using both valid and invalid inputs and pre-conditions.
* No matter how hard you try, testing EVERYTHING is pretty much impossible.
* The inputs and outputs alone have an infinite number of combinations, so it is 100% not possible to test an application from every angle.
* So, instead of doing the exhaustive testing we can use risks and priorities to focus testing efforts.
* For example: In an application in one screen there are 15 input fields, each having 5 possible values, then to test all the valid combinations you would need 30 517 578 125 (515) tests.
* This is very unlikely that the project timescales would allow for this number of tests.
* So, accessing and managing risk is one of the most important activities and reason for testing in any project.

**3.Early Testing**

* Early testing means incorporating testing as early as possible in the development process.
* Testing the requirements before coding begins.
* Early Testing should start as early as possible in the Software Development Life Cycle.
* So that any defects in the requirements or design phase are captured in early stages.
* It is much cheaper to fix a Defect in the early stages of testing.

**4.Defect Clustering**

* Defect Clustering which states that a small number of modules contain most of the defects detected during pre-release testing or are responsible.
* This is because defects are not evenly distributed within a system but are clustered.
* It could be due to multiple factors, such as the modules might be complicated or the coding related to such modules might be complex.
* The defect clustering method relies on the team’s knowledge and experience to identify which modules to test.
* You can identify such risky modules from your experience. Therefore, the team only has to focus on those “sensitive” areas, saving both time and effort.

**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.
* The Pesticide Paradox generally refers to the practice of repeating the exact same test cases over and over again.
* As time passes, these test cases will cease to find new bugs.
* Developers will create tests which are passing so they can forget about negative or edge cases.
* This is based on the theory that when you repeatedly spray the same pesticide on crops in order to eradicate insects, the insects eventually develop an immunity, making the pesticide ineffective.
* Therefore, in order to overcome the Pesticide Paradox, it is imperative to regularly review and update the test cases so that more defects can be found.
* However, if this process is not followed, and the same tests are repeated over and over again, then eventually there will be no new bugs found, but it doesn’t mean the system is 100 % bug free.
* To make testing more effective, testers must constantly look for ways to improve the existing test methods.
* To test new features of the software or system, new tests must be developed.

**6.Testing is Context Dependent**

* Testing is context dependent which basically means that the way you test an e-commerce site will be different from the way you test a commercial off the shelf application.
* All the developed software’s are not identical.
* You might use a different approach, methodologies, techniques, and types of testing depending upon the application type.
* According to this principle, testing depends on the context of the software developed, and this is entirely true.
* The reality is that every application has its own unique set of requirements, so we can’t put testing in a box.
* For example, health industry applications require more testing than gaming applications, safety-critical systems require more testing than company presentation websites, and online banking applications will require different testing approaches than e-commerce sites or advertising sites.

**7.Absence of Errors Fallacy**

* The software which we built not only must be 99% bug-free software but also it must fulfil the business, as well as user requirements otherwise it will become unusable software.
* Even bug-free software may still be unusable if incorrect requirements are incorporated into the software, or if the software fails to meet the business needs.
* Software testing is not finding defects, but also to check that software addresses the business needs.
* The absence of Error is a Fallacy i.e., finding and fixing defects does not help if the system build is unusable and does not fulfil the user’s needs & requirements.

1. **Difference between QA v/s QC v/s Tester**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Quality Assurance** | **Quality Control** | **Tester** |
| **1** | Quality assurance is process oriented. | Quality control, alternatively, is product oriented. | Testing is a subset of Quality Control. |
| **2** | Quality assurance is all about preventing defects by ensuring the processes used to manage and create deliverable works. | Quality control is the function of software quality that determines the ending result is what was expected. | Testing is the process of executing a system in order to detect bugs in the product so that they get fixed. |
| **3** | Quality Assurance is Preventive activities. | Quality Control is a corrective process. | Testing is a preventive process. |
| **4** | QA is about engineering processes that assure quality is achieved in an effective and efficient way. | QC detects bugs by inspecting and testing the product. | Testing is an integral part of QC as it helps demonstrate that the product runs the way it is expected and designed for. |
| **5** | 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 requirements. | Activities which ensure the identification of bugs/error/defects in the Software. |
| **6** | 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. |
| **7** | 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. |

1. **Difference between Smoke and Sanity?**

|  |  |  |
| --- | --- | --- |
| **No.** | **Smoke Testing** | **Sanity Testing** |
| **1** | Smoke Testing has a goal to verify “stability” . | Sanity Testing has a goal to verify “rationality”. |
| **2** | Smoke Testing is done by both developers or testers. | Sanity Testing is done by testers. |
| **3** | Smoke Testing verifies the critical functionality of the system. | Sanity Testing verifies the new functionality like bug fixes. |
| **4** | Smoke testing is a subset of acceptance testing. | Sanity testing is a subset of Regression Testing. |
| **5** | Smoke testing verifies the entire system from end to end. | Sanity Testing verifies only a particular component. |
| **6** | Smoke testing is usually documented and scripted. | Sanity testing is not documented and is unscripted. |
| **7** | It is a well elaborate and planned testing. | This is not a planned test and is done only when there is a shortage of time. |
| **8** | This is a wide and deep testing. | This is a wide and shallow testing. |

1. **Difference between verification and Validation**

|  |  |  |
| --- | --- | --- |
| **No.** | **Verification** | **Validation** |
| **1** | Verification is static testing. | Validation is dynamic testing. |
| **2** | Verification means Are we building the product right? | Validation means Are we building the right product? |
| **3** | It includes checking documents, design, codes and programs. | It includes testing and validating the actual product. |
| **4** | Verification does not involve code execution. | Validation involves code execution. |
| **5** | Methods used in verification are reviews, walkthroughs, inspections and desk-checking. | Methods used in validation are Black Box Testing, White Box Testing and non-functional testing. |
| **6** | Verification checks whether the software confirms a specification. | Validation checks whether the software meets the requirements and expectations. |
| **7** | Verification finds the bugs early in the development cycle. | Validation finds the bugs that verification cannot catch. |
| **8** | Verification process targets on software architecture, design, database, etc. | Validation process targets the actual software product. |
| **9** | Verification is done by the QA team. | Validation is done by the involvement of testing team with QA team. |
| **10** | Verification process comes before validation. | Validation process comes after verification. |

1. **Explain types of Performance testing.**

* Performance testing is a non-functional testing technique performed to determine the system parameters in terms of responsiveness and stability under various workload.
* Performance testing measures the quality attributes of the system, such as scalability, reliability and resource usage.
* **Types of Performance Testing:**

1. **Load testing**
2. **Stress testing**
3. **Endurance testing**
4. **Spike testing**
5. **Volume testing**
6. **Scalability testing**
7. **What is Error, Defect, Bug and failure?**

* Testing is the process of identifying defects, where a defect is any variance between actual and expected results.
* “A mistake in coding is called Error, error found by tester is called Defect, defect accepted by development team then it is called Bug, build does not meet the requirements then it Is Failure.”
* **Error:** A bug is the result of a coding error. An Error found in the development environment before the product is shipped to the customer. A programming error that causes a program to work poorly, produce incorrect results or crash. An error in software or hardware that causes a program to malfunction. A bug is the terminology of Tester.
* **Defect:** It can be simply defined as a variance between expected and actual. The defect is an error found AFTER the application goes into production. It commonly refers to several troubles with the software products, with their external behaviour or with its internal features. In other words, a Defect is a difference between expected and actual results in the context of testing. It is the deviation of the customer requirement.
* **Bug:** A bug is the result of a coding error. An Error found in the development environment before the product is shipped to the customer. A programming error that causes a program to work poorly, produce incorrect results or crash. An error in software or hardware that causes a program to malfunction. A bug is the terminology of Tester.
* **Failure:** A failure is the inability of a software system or component to perform its required functions within specified performance requirements. When a defect reaches the end customer it is called a Failure. During development, Failures are usually observed by testers.

1. **Difference between Priority and Severity**

|  |  |  |
| --- | --- | --- |
| **No.** | **Priority** | **Severity** |
| **1** | Priority is a term that defines how fast we need to fix a defect. | Severity is a term that denotes how severely a defect can affect the functionality of the software. |
| **2** | Priority is basically a parameter that decides the order in which we should fix the defects. | Severity is basically a parameter that denotes the total impact of a given defect on any software. |
| **3** | Priority relates to the scheduling of defects to resolve them in software. | Severity relates to the standards of quality. |
| **4** | The product manager basically decides a defect’s priority level. | The testing engineer basically decides a defect’s severity level. |
| **5** | Priority is the order in which developer has to fix the bug. | Severity is how seriously the bug is affecting the application. |
| **6** | If high priority is mentioned then the developer has to fix it at the earliest.priority status is set based on the customer requirements. | Severity type is defined by the tester based on the written test cases and functionality. |
| **7** | Its value is subjective. | Its value is objective. |
| **8** | Its value changes from time to time. | Its value doesn’t change from time to time. |
| **9** | 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. |
| **10** | Priority means how fast defect has to be fixed. | Severity means how severe defect is affecting the functionality. |
| **11** | Priority is related to scheduling to resolve the problem. | Severity is related to the quality standard. |
| **12** | Product manager decides the priorities of defects. | Testing engineer decides the severity level of the defect. |
| **13** | Priority is of 3 types: Low, Medium, and High. | Severity is of 5 types: Critical, Major, Moderate, Minor, and Cosmetic. |

1. **What is Bug Life Cycle?**

* When the expected and actual behavior is not matching, an incident needs to be raised. An incident may be a Bug. It is a programmer's fault where a programmer intended to implement a certain behavior, but the code fails to correctly conform to this behavior because of incorrect implementation in coding. It is also known as Defect.
* **The different phases of Bug life cycle are :**
* New or Opened
* Assigned
* Fixed
* Tested
* Closed

1. **Explain the difference between Functional testing and Non Functional testing**

|  |  |  |
| --- | --- | --- |
| **No.** | **Functional Testing** | **Non-Functional Testing** |
| **1** | It verifies the operations and actions of an application. | It verifies the behaviour of an application. |
| **2** | It is performed before non-functional testing. | It is performed after the functional testing. |
| **3** | It is based on customer’s requirements. | It focuses on customer’s expectation. |
| **4** | It is easy to define functional requirements. | It is difficult to define the requirements for non-functional testing. |
| **5** | Helps to validate the behaviour of the application. | Helps to validate the performance of the application. |
| **6** | Carried out to validate software actions. | It is done to validate the performance of the software. |
| **7** | Functional testing is carried out using the functional specification. | This kind of testing is carried out by performance specifications |
| **8** | Functional testing is easy to execute by manual testing. | It’s very hard to perform non-functional testing manually. |
| **9** | It tests what the product does. | It describes how the product does. |
| **10** | Functional testing is based on the business requirement. | Non-functional testing is based on the performance requirement. |
| **11** | Examples of Functional Testing:  1. Unit Testing  2. Smoke Testing  3. Integration Testing  4. Regression Testing | Examples of Non-Functional Testing:  1. Performance Testing  2. Load Testing  3. Stress Testing  4. Scalability Testing |

1. **What is the difference between the STLC (Software Testing Life Cycle) and SDLC (Software Development Life Cycle)?**

|  |  |  |
| --- | --- | --- |
| **No.** | **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 | It focuses only on testing the software. | Besides development other phases like testing is also included. |
| 3 | STLC involves only five phases or steps.   1. Test Planning and Controlling 2. Test Analysis and Design 3. Test Implementation and Execution 4. Evaluating Exit Criteria and Reporting 5. Test Closure Activities | SDLC involves total six phases or steps.   1. Requirements Gathering 2. Analysis 3. Design 4. Implementation 5. Testing 6. Maintenance |
| 4 | In STLC, less number of members (testers) are needed. | In SDLC, more number of members (developers) are required for the whole process. |
| 5 | 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. |
| 6 | Goal of STLC is to complete successful testing of software. | Goal of SDLC is to complete successful development of software. |
| 7 | It helps in making the software defects free. | It helps in developing good quality software. |
| 8 | STLC phases are performed after SDLC phases. | SDLC phases are completed before the STLC phases. |
| 9 | Regression tests are run by QA team to check deployed maintenance code and maintains test cases and automated scripts. | Post deployment support , enhancement , and update are to be included if necessary. |
| 10 | A tested software system is the end result of STLC. | Creation of reusable software systems is the end result of SDLC. |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Test Scenarios** | **Test Cases** | **Test Script** |
| **1** | Test Scenario is any functionality that can be tested. | Test cases is set of actions executed to verify particular features or functionality. | Test script is a set of instruction to test an app automatically. |
| **2** | Test Scenario is derived from test artifacts like Business Requirement Specification(BRS) and Software Requirement Specification. | Test Cases is mostly derived from test scenarios. | Test Script is mostly derived from test cases. |
| **3** | Test Scenario helps test the end to end functionality in an agile way. | Test Cases helps in exhaustive testing of an application. | Test Script helps to test specific things repeatedly. |
| **4** | Test Scenario is more focused on what to test. | Test Cases is focused on what to test and how to test. | Test Script is focused on the expected result. |
| **5** | Test Scenario takes less time and fewer resources to create. | Test Cases requires more resources and time. | Test Script requires less time for testing but more resources for scripts creating and updating. |
| **6** | Test Scenario includes an end to end functionality to be tested. | Test Cases includes test steps,data,expected results for testing. | Test Script includes different commands to develop a script. |
| **7** | Test Scenario allows quickly accessing the testing scope. | Test Cases allows detecting errors and defects. | Test Script allows carrying out an automatic execution of test cases. |
| **8** | The main task of Test Scenario is to check the full functionality of a software application. | The main task of Test Cases is to verify compliance with the applicable standards, guidelines and customer requirements. | The main task of Test Script is to verify that nothing is skipped and the results are true as the desired testing plan. |
|  |  |  |  |

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

* A Test Plan is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverable, and resources required to perform testing for a software product.
* Test Plan helps us determine the effort needed to validate the quality of the application under test.
* The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager.
* The Test Plan document is usually prepared by the Test Lead or Test Manager and the focus of the document is to describe what to test, how to test, when to test and who will do what test.
* **Types of Test Plan :**

**-->There are three types of the test plan**

* + - Master Test Plan
    - Phase Test Plan
    - Testing Type Specific Test Plans
* **Master Test Plan**

Master Test Plan is a type of test plan that has multiple levels of testing. It includes a complete test strategy.

* **Phase Test Plan**

A phase test plan is a type of test plan that addresses any one phase of the testing strategy. For example, a list of tools, a list of test cases, etc.

* **Specific Test Plans**

Specific test plan designed for major types of testing like security testing, load testing, performance testing, etc. In other words, a specific test plan designed for non- functional testing.

* **The information that should be covered in Test Plan**
  + - Introduction to the Test Plan document
    - Assumptions when testing the application
    - List of test cases included in Testing the application
    - List of features to be tested
    - What sort of Approach to use when testing the software
    - List of Deliverable that need to be tested
    - The resources allocated for testing the application
    - Any Risks involved during the testing process
    - A Schedule of tasks and milestones as testing is started

1. **What is priority?**

* Priority is defined as the order in which the defects should be resolved.
* The priority status is usually set by the testing team while raising the defect against the dev team mentioning the 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 end customer requirement.
* For example: If the company logo is incorrectly placed in the company's web page then the priority is high but it is of low severity.
* **Priority can be marked as either of the following states:**

1. **Low**
2. **Medium**
3. **High**
4. **Critical**
5. **Low** - This defect can be fixed after the critical ones are fixed.
6. **Medium** - The defect should be resolved in the subsequent builds.
7. **High** - The defect must be resolved immediately because the defect is affecting the application to a considerable extent and the relevant modules cannot be used until it's fixed.
8. **Critical** - The defect must be resolved immediately because the defect is affecting the application or the product severely and the product cannot be used until it has been fixed.
9. **What is severity?**

* Severity is absolute and Customer-Focused.
* Severity is the extent to which the defect can affect the software.
* It defines the impact that a given defect has on the system.
* For example: If an application or web page crashes when a remote link is clicked, in this case clicking the remote link by an user is rare but the impact of application crashing is severe. So the severity is high but priority is low.
* **Severity can be of following types:**

1. **Critical**
2. **Major**
3. **Moderate**
4. **Minor**
5. **Cosmetic**
6. **Critical:** The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable and there is no acceptable alternative method to achieve the required results then the severity will be stated as critical.
7. **Major:** The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable but there exists an acceptable alternative method to achieve the required results then the severity will be stated as major.
8. **Moderate:** The defect that does not result in the termination, but causes the system to produce incorrect, incomplete or inconsistent results then the severity will be stated as moderate.
9. **Minor:** The defect that does not result in the termination and does not damage the usability of the system and the desired results can be easily obtained by working around the defects then the severity is stated as minor.
10. **Cosmetic:** The defect that is related to the enhancement of the system where the changes are related to the look and field of the application then the severity is stated as cosmetic.
11. **Bug categories are…**

* Bug categories are :

1. Security
2. Database
3. Functionality (Critical/General)
4. UI
5. **Advantage of Bugzilla**

* **The Advantages of Bugzilla are:**
  + It is an open-source widely used bug tracker
  + It is easy in usage and its user interface is understandable for people without technical knowledge
  + It easily integrates with test management instruments
  + It integrates with an e-mailing system
  + It automates documentation.
  + Automatic Duplicate Bug Detection.
  + Search option with advanced features.
  + File/Modify Bugs By Email.
  + Move Bugs Between Installs.
  + Multiple Authentication Methods (LDAP, Apache server).
  + Time Tracking.
  + Automated bug reporting; has an API to interact with system.
  + Integrated email capabilities.
  + Detailed permissions system.
  + Optimized database structure to enhance performance.
  + Robust security.
  + Powerful query tool.
  + Ideal for small projects.

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

* **There are various methodologies present in agile testing and those are listed below:**

1. **Scrum**
2. **eXtreme Programming**
3. **eXtreme Programming**
4. **Dynamic System Development Method (DSDM)**
5. **Test Driven Development (TDD)**
6. **Feature Driven Development(FDD)**
7. **XBreed**
8. **Crystal**

* **Scrum :** SCRUM is an agile development method which concentrates particularly on how to manage tasks within a team based development environment. Scrum is derived from activity that occurs during rugby match. Scrum believes in empowering the development team and advocates working in small teams.
* **eXtreme Programming :** This is a light weight agile testing methodology in which development and testing happen in parallel. Business requirements are gathered in terms of stories.
* **Dynamic System Development Method (DSDM) :** This is an Iterative and incremental approach that emphasizes on the continuous user involvement.
* **Test Driven Development (TDD) :** This is a technique which has short iterations where new test cases covering the desired improvement or new functionality are written first.
* **Feature Driven Development(FDD) :**  This is an iterative and incremental software development process and this can aim depends on the features.
* **XBreed :** Agile enterprise previously known as Xbreed .It is agile way of managing, architecting and monitoring the enterprise.
* **Crystal :** Crystal is an adaptive technique mainly used for software development methodologies.

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

|  |  |  |
| --- | --- | --- |
| No. | **Authorization** | **Authentication** |
| 1 | Authorization determines what resources a user can access. | Authentication verifies who the user is. |
| 2 | Authorization works through settings that are implemented and maintained by the organization. | Authentication works through passwords, one-time pins, biometric information, and other information provided or entered by the user. |
| 3 | Authorization is the process of giving permission to access the resources. | Authentication is the process of identifying a user to provide access to a system. |
| 4 | In this, it is verified that if the user is allowed through the defined policies and rules. | In this, the user or client and server are verified. |
| 5 | It is usually done once the user is successfully authenticated. | It is usually performed before the authorization. |
| 6 | It requires the user's privilege or security level. | It requires the login details of the user, such as user name & password, etc. |
| 7 | Data is provided through the access tokens. | Data is provided through the Token Ids. |
| 8 | Authorization isn’t visible to or changeable by the user. | Authentication is visible to and partially changeable by the user. |
| 9 | Authorization permissions cannot be changed by the user. The permissions are given to a user by the owner/manager of the system, and he can only change it. | Authentication credentials can be partially changed by the user as per the requirement. |
| 10 | Example: After employees successfully authenticate themselves, they can access and work on certain functions only as per their roles and profiles. | Example: Entering Login details is necessary for the employees to authenticate themselves to access the organizational emails or software. |

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

* The common problems faced Web Application testing :
  + Cross browser compatibility issue
  + Application getting slow
  + Testing deadline
  + Take more time to create test case
  + Take more time to create HLR & bug report

1. **When to used Usability Testing?**

* Usability testing is an essential part of product development.
* You test before a redesign, you test during the redesign and then you test afterwards too.
* Usability testing can and should be conducted on the current iteration of a product before beginning any new design work, after you’ve begun the strategy work around a brand new site or app.
* This will quickly identify areas for opportunity, and reduce the amount of assumptions your design team will make with regard to what the user wants.
* After the usability tests analysis, the team should have the ability to pinpoint the steps needed to achieve the project goals with as little disruption as possible.

1. **What is the procedure for GUI Testing?**

* The first method of GUI testing is Manual based testing.
* The simplest way of executing the GUI testing is purely using the application manually. Usually, manual-based testing is implemented by enthusiastic parament test engineers.
* **Procedure for GUI :**

1. **MANUAL BASED TESTING**

 Under this approach, graphical screens are checked manually by testers in conformance with the requirements stated in business requirements document.

1. **RECORD AND REPLAY**

 GUI testing can be done using automation tools. This is done in 2 parts. During Record , test steps are captured into the automation tool. During playback, the recorded test steps are executed on the Application under Test. Example of such tools - QTP.

1. **MODEL BASED TESTING**

A model is a graphical description of system’s behavior. It helps us to understand and predict the system behavior. Models help in a generation of efficient test cases using the system requirements.