1. **What is software testing?**

* Testing is the process of evaluating system or components of system to find whether it satisfies the specified requirements or not.
* Testing means executing a system in order to identify any gaps, errors or missing requirements in contrary to the actual requirements.
* Hence, we can say that Software Testing is a process used to identify the correctness, completeness and quality of developed software.

1. **What is Exploratory Testing?**

* Exploratory Testing is a concurrent process where
* Test design, execution and logging happen simultaneously
* Testing is often not recorded
* Testing on a test charter that may include
* Scope of the testing (in and out)
* The focus of exploratory testing is more on testing as a “thinking” activity
* A brief description of how tests will be performed.
* Expected problems.
* It is carried out in time box intervals.
* It is more structured than error guessing.

1. **What is traceability matrix?**

* To protect against changes you should be able to trace back from every system component to the original requirement that caused its presence.
* 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 (i.e. from Requirements to Design or Coding) or backward (i.e. from Coding to Requirements). There are many user defined templates for RTM.
* Each requirement in the RTM document is linked with its associated test case, so that testing can be done as per the mentioned requirements.
* Bug ID is also include and linked with its associated requirements and test case.
* The main goals for this matrix are:
* Make sure Software is developed as per the mentioned requirements.
* Helps in finding the root cause of any bug.
* It also helps in tracing the developed documents during different phases of SDLC.

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

* Software testing technique in which tests are designed to include representatives of boundary values.
* It operates on the basis that experience shows us that errors are most likely to exist at the boundaries between partitions and in doing so incorporates a degree of negative testing into the test design
* Test cases are designed to exercise the software on and at either side of boundary values
* Find the boundary and then test one value above and below it. It always results in two test cases per boundary for valid inputs and three tests cases per boundary for all inputs.

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

* The aim is to treat groups of inputs as equivalent and to select one representative input to test them all.
* It can be used for all Levels of Testing
* Equivalence partitioning is the process of defining the optimum number of tests by:
* Reviewing documents such as the Functional Design Specification and Detailed Design Specification, and identifying each input condition within a function,
* Selecting input data that is representative of all other data that would likely invoke the same process for that particular condition.
* If the tester chooses the right partitions, the testing will be accurate and efficient

1. **What is Integration testing?**

* This testing performed to expose defects in the interfaces and in the interactions between integrated components or systems
* It is a level of the software testing process where individual units are combined and tested as a group.
* The purpose of this testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.
* It 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?
4. **What is Alpha testing?**

* It is always performed by the developers at the software development site.
* Sometimes it is also performed by Independent Testing Team.
* Alpha Testing is not open to the market and public, it is conducted for the software application and project.
* It is always performed in Virtual Environment within the organization.
* Alpha Testing is definitely performed and carried out at the developing organizations location with the involvement of developers.
* It comes under the category of both White Box Testing and Black Box Testing.
* During this phase, the following will be tested in the application:
* Spelling Mistakes
* Broken Links
* Cloudy Directions
* 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.
* Unit testing, integration testing and system testing when combined are known as alpha testing.

1. **What is beta testing?**

* It is always performed by the customers at their own site.
* It is not performed by Independent Testing Team.
* Beta Testing is always open to the market and public.
* It is usually conducted for software product and is performed in Real Time Environment.
* It is always performed outside the organization.
* Beta Testing (field testing) is performed and carried out by users or you can say people at their own locations and site using customer data.
* It is only a kind of Black Box Testing.
* Beta Testing is always performed at the time when software product and project are marketed.
* It is always performed at the user’s premises in the absence of the development team.
* It is also considered as the User Acceptance Testing (UAT) which is done at users area.
* Beta testing can be considered “pre-release” testing.

1. **What is component testing?**

* Component Testing is also called Unit Testing.
* Component (Unit) – A minimal software item that can be tested in isolation. It means “A unit is the smallest testable part of software.”
* Component Testing – The testing of individual software components.
* Unit Testing is a level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed.
* Unit 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 testing frameworks, drivers, stubs and mock or fake objects are used to assist in
* unit 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.
* It usually has one or a few inputs and usually a single output. In procedural programming a unit may be an individual program, function, procedure, etc.
* In object-oriented programming, the smallest unit is a method, which may belong to a base/super class, abstract class or derived/child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.)
* Unit testing is a method by which individual units of source code are tested to determine if
* they are fit for use.
* Unit tests find problems early in the development cycle.
* Unit testing is performed by using the White Box Testing method.

1. **What is functional system testing?**

* A requirement that specifies a function that a system or system component must perform is called functional requirement.
* A Requirement may exist as a text document and/or a model
* There is two types of techniques to test Functional requirements:

1. **Requirement Based Functional Testing:**

* Testing against requirements and specifications
* Test procedures and cases derived from:
* detailed user requirements
* system requirements functional specification
* User documentation/instructions
* high level System design

1. **Process Based Testing**

* Test procedures and cases derived from:
* Expected user profiles
* Business scenarios
* Use cases
* Functional System Testing Functionality As below:

|  |  |
| --- | --- |
| **Accuracy** | Provision of right or agreed results or effects |
| **Interoperability** | Ability to interact with specified systems |
| **Compliance** | Adhere to applicable standards, conventions, regulations and laws |
| **Auditability** | Ability to provide adequate and accurate audit data |
| **Suitability** | Presence and appropriateness of functions for specified tasks |

1. **What is Non-Functional Testing?**

* Testing of those requirements that do not relate to functionality
* Emphasis on non-functional requirements:
* Performance
* Load
* Data volumes
* Storage
* Recovery
* Usability
* Stress
* Security
* The non-functional aspects of a system are all the attributes other than business functionality, and are as important as the functional aspects. These include:
* the look and feel and ease of use of the system
* how quickly the system performs
* how much the system can do for the user
* It is also about:
* how easy and quick the system is to install
* how robust it is
* how quickly the system can recover from a crash
* Types of System Testing
* Usability Testing
* Load Testing
* Regression Testing
* Recovery Testing
* Migration Testing
* Testing Budget
* Functional Testing
* Hardware/Software Testing

1. **What is GUI Testing?**

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

1. **What is Adhoc testing?**

* It is also known as Error guessing technique.
* Adhoc testing is an informal testing type intended to break the system.
* It does not follow any specific test design techniques.
* It does not create any test cases.
* It can be performed only if the tester has very high knowledge of the system under test.
* Main aim is to find defects by random checking.
* In this technique experienced and good testers are encouraged to think of situations in which the software may not be able to cope.

1. What is white box testing and list the types of white box testing?
2. What is black box testing? What are the different black box testing techniques?
3. Mention what are the categories of defects?
4. **Mention what bigbang testing is?**

* In Big Bang integration testing all components or modules is integrated simultaneously, after which everything is tested as a whole.
* The advantage is that everything is finished before integration testing starts.
* The major disadvantage is that it is a time consuming process and difficult to trace the cause of failure because of this late integration.
* Here all component are integrated together at once, and then tested.

1. **What is the purpose of exit criteria?**

* Exit criteria is used to determine when testing at any stage is complete
* The set of generic and specific conditions, agreed upon with the stakeholders, for permitting a process to be officially completed
* Purpose of exit criteria is to define when we STOP testing either at the:
* End of all testing – i.e. product Go Live
* End of phase of testing (e.g. hand over from System Test to UAT)

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

* Regression testing should be carried out:
* when the system is stable and the system or the environment changes
* when testing bug-fix releases as part of the maintenance phase

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
8. **Testing shows presence of Defects**

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

1. **Exhaustive Testing in Impossible**

* Testing everything including all combinations of inputs and preconditions is not possible.
* So, instead of doing the exhaustive testing we can use risks and priorities to focus testing effort, that is, we must Prioritise our testing effort using a Risk Based Approach.

1. **Early Testing**

* Testing activities should start as early as possible in the software or system development lifecycle, and should be focused on defined objectives.

1. **Defect Clustering**

* Defects are not evenly spread in a system, but they are ‘clustered’ usually confined to a small number of modules
* Similarly, most operational failures of a system are usually confined to a small number of modules

1. **Pesticide Paradox**

* If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new defects.
* To overcome this “pesticide paradox”, the test cases need to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects.
* Testing identifies bugs, and programmers respond to fix them which improves the software.
* As bugs are eliminated by the programmers, the software improves which reduces the effectiveness of previous tests.
* Therefore we must learn to create and use new tests based on new techniques to catch new bugs
* N.B It's called the "pesticide paradox" after the agricultural phenomenon, where bugs such as the boll weevil build up tolerance to pesticides, leaving you with the choice of ever-more powerful pesticides followed by ever-more powerful bugs or an altogether different approach.’ – Beizer

1. **Testing is Context Dependent**

* Testing is basically context dependent
* Different kinds of sites are tested differently.
* For example

Safety – critical software is tested differently from an e-commerce site.

* Whilst, Testing can be 50% of development costs, in NASA's Apollo program it was 80% testing

1. **Absence of Errors Fallacy**

* If the system built is unusable and does not fulfil the user’s needs and expectations then finding and fixing defects does not help.
* It doesn’t make it a good system even after defects have been resolved, as it may still be unusable and/or does not fulfil the user needs.

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **QA**  **(Quality Assurance)** | **QC**  **(Quality Control)** | **Tester** |
| **1** | Activities which ensure the implementation of processes, procedures and standards in context to verification of developed software with 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 processes. | 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 QA | Testing is a subset of QC |

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

* Typical Software projects contain thousands of source code files, and creating a working program is complicated and time consuming.
* So we need to use “build” software to create an executable program and the process is called “Software Build”.

1. **Smoke Testing:**

* It is performed after the software build to ascertain that critical functionalities of the program is working fine.
* It is performed before any functional or regression tests are executed.
* The purpose is to reject badly broken application.
* The test cases chosen cover the most important functionality or component of the system.

1. **Sanity Testing:**

* It is performed after receiving the software build with minor changes in code or functionality, to ascertain that the bugs have been fixed and no further issues are introduced due to these changes.
* If it fails, then build is rejected to save time and costs involved in a more rigorous testing.
* The objective is “not” to thoroughly verify the functionality, but to determine that the developer has applied some rationality (Sanity) while producing the software.

1. **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 product is being built according to the requirements and design specifications | To ensure that the product actually meets the user’s needs and that the specifications were correct in the first place. |
| **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 or software |
| **Activities** | * Reviews * Walkthroughs * Inspection | * Testing |

1. Explain types of Performance testing.
2. **What is Error, Defect, Bug and failure?**

* “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 discrepancy between a computed, observed, measured value or condition and the true, specified, theoretically correct value or condition. This can be a misunderstanding of the internal state of the software, an oversight in terms of memory management, confusion about the proper way to calculate a value, etc. **It is the Deviation from actual and the expected value.**
* **Failure:** The inability of a system or component to perform its required functions within specified performance requirements.
* **Bug:** A fault in a program which causes the program to perform in an unintended or unanticipated manner. It is found in the development environment before the product is shipped to the respective customer.
* **Defect:** Commonly refers to several troubles with the software products, with its external behaviour or with its internal features. It is found in the product itself after it is shipped to the respective customer

1. **Difference between Priority and Severity**

* **Defect 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.
* **For example:** 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.
* **Defect Severity:**
* Severity is absolute and Customer-Focused.
* It is the extent to which the defect can affect the software.
* In other words 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.

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

* “A computer bug is an error, flaw, mistake, failure, or fault in a computer program that prevents it from working correctly or produces an incorrect result. Bugs arise from mistakes and errors, made by people, in either a program’s source code or its design.”
* **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’**.
* When a bug is discovered, it goes through several states and eventually reaches one of the terminal states, where it becomes inactive and closed.
* The process by which the defect moves through the life cycle is depicted in below diagram.

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

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Functional Testing** | **Non-Functional Testing** |
| 1 | It is performed using the functional specification provided by the client and verifies the system against the functional requirements | It checks the performance, reliability, scalability and other non-functional aspects of the software system. |
| 2 | It is executed first | It should be performed after functional testing |
| 3 | Manual testing or Automation tools can be used | Using tools will be effective for this type of testing |
| 4 | Business requirements are the inputs | Performance parameters like speed, scalability are the inputs |
| 5 | It describes what the product dies. | It describes how good the product works. |
| 6 | Easy to do Manual testing | Tough to do Manual testing |
| 7 | Types of Functional testing are   * Unit Testing * Smoke Testing * Sanity Testing * Integration Testing * White box testing * Black Box testing * User Acceptance testing * Regression Testing | Types of Non-functional testing are   * Performance Testing * Load Testing * Volume Testing * Stress Testing * Security Testing * Installation Testing * Penetration Testing * Compatibility Testing * Migration Testing |