# Module 2 (Manual Testing)

Answer the following Questions

**Q. 1 What is software testing?**

**Ans. Software testing is a process used to identify the correctness, completeness, and quality of developed computer software. It is also the process of executing a program or application with the intent of finding software bugs.**

**Q. 2 What is Exploratory Testing?**

**Ans. Exploratory testing is a type of software testing where test cases are not created in advance but testers check the system on the fly. Before doing a test, they can list down ideas for what to test. Testing as a "thinking" activity is more the focus of exploratory testing. Exploratory testing is useful in Agile models because it focuses on discovery, exploration, and learning. It highlights the individual tester's own independence and responsibility.**

**Q.3 What is a traceability matrix?**

**Ans. 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 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. Furthermore, Bug ID is also included and linked with its associated requirements and test cases. The main goals for this matrix are: Make sure software is developed as per the mentioned requirements. It helps in finding the root cause of any bug.**

**A requirements traceability matrix is a document that traces and maps user requirements [requirement ids from a requirement specification document] with the test case ids. The purpose is to make sure that all the requirements are covered in test cases so that while testing, no functionality can be missed.**

Q. 4 What is boundary value testing?

Ans. The boundary value analysis' testing technique is used to identify errors at boundaries rather than finding those that exist in the centre of the input domain. The next step in the process of developing test cases is boundary value analysis, in which test cases are chosen from the boundaries of the equivalence classes.

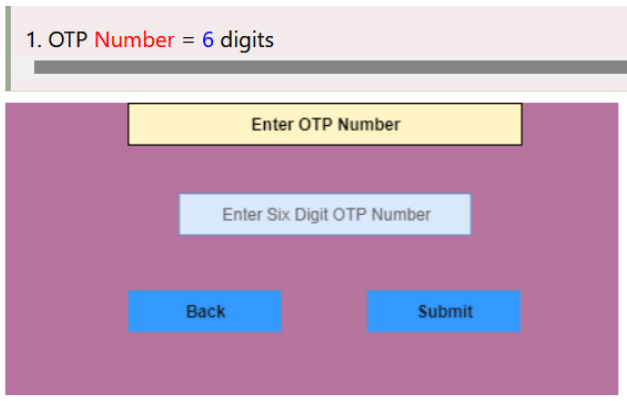
Q.5 What is equivalence partitioning testing?

Ans. In the software testing technique known as "equivalence partitioning," input data is split into parts of valid and invalid values, and it is required that every partition behave similarly. A condition of one partition must be true in order for the condition of another equal partition to also be true, and vice versa if the condition of one partition is false.

According to the equivalent partitioning concept, test cases should be created to at least for a short time touch on each partition. Every equal partition's values must behave similarly to one another. The specifications and needs of the software are used to create the equivalence partitions. Due to a decreased number of test cases–from infinite to finite–this strategy has the advantage of reducing testing duration.
At every stage of the testing procedure, it is relevant.

Examples of the Equivalence Partitioning technique

Assume that there is a function in a software application that accepts a particular number of digits, not greater or less than that particular number. For example, an OTP number that contains only six digits will not be accepted, and the application will redirect the user to the error page.




| INVALID | INVALID | VALID | VALID |
|---|---|---|---|
| 1 Test case | 2 Test case | 3 Test case | |
| DIGITS >=7 | DIGITS<=5 | DIGITS = 6 | DIGITS = 6 |
| 93847262 | 9845 | 456234 | 451483 |

Q.6 What is integration testing?

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

Unit testing uses modules for testing purposes, and these modules are combined and tested in integration testing. The Software is developed with a number of software modules that are coded by different coders or programmers. The goal of integration testing is to check the correctness of communication among all the modules. Once all the components or modules are working independently, we need to check the data flow between the dependent modules. This is known as integration testing.

Q.7 What determines the level of risk?

Ans. A factor that could result in future negative consequences is usually expressed as impact and likelihood. When testing does find defects, the quality of the software system increases when those defects are fixed. The quality of systems can be improved through lessons learned from previous projects. Analysis of root causes of defects found in other projects can lead to Process Improvement Process Improvement can prevent those defects reoccurring Which in turn, can improve the Quality of future systems

Q.8 What is alpha testing?

Ans. Alpha testing is the initial phase of validating whether a new product will perform as expected. Internal staff carry out alpha tests early in the development process and are followed up with beta tests, in which a sampling of the intended audience actually tries the product out.

It is always performed by the developers at the software development site. Sometimes it is also performed by Independent Testing Team. Alpha Testing is not open to the market and public It is conducted for the software application and project. It is always performed in Virtual Environment. It is always performed within the organization. It is the form of Acceptance Testing. Alpha Testing is definitely performed and carried out at the developing 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. It is always performed at the developer's premises in the absence of the users. It is considered User Acceptance Testing (UAT) which is done at developer's area. Unit testing, integration testing, and system testing when combined are known as alpha testing.

Q.9 What is beta testing?

Ans. Users always perform it on their own property. Independent The Independent Testing Team does not take it out. Beta testing is always accessible to the general public and the market. It is usually carried out for software products. It is carried out in a real-time setting. It is always done outside of the company. It also takes the form of an acceptance test. Users, or you could say, people at the site, perform and conduct beta testing (field testing). using client information, their own locations and websites. It only represents a subset of black box testing. Beta testing is always performed while a software project and product are in the market. In the absence of the development team, it is always conducted on the user's property.

Beta Testing is performed by real users of the software application in a real environment. Beta testing is one of the types of user acceptance testing. A beta version of the software, whose feedback is needed, is released to a limited number of end-users of the product to obtain feedback on the product's quality. Beta testing helps in the minimization of product failure risks, and it provides increased quality of the product through customer validation. It is the last test before shipping a product to the customer. One of the major advantages of beta testing is direct feedback from customers.

Q. 10 What is component testing?

Ans. Component testing is a sub-category of software testing in which each individual component of the software is separately tested without integrating with other components. It is also known as module testing, when viewed from an architecture perspective. A software, in general, is made up of several components

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 Component can be tested in isolation – stubs/drivers may be employed Unit testing frameworks, drivers, stubs and mock or fake objects are used to assist in unit testing. Test cases derived from component specification (module/program spec) Functional and Non-Functional 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.

Q. 11 What is functional system testing?

Ans. FUNCTIONAL TESTING is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the functional requirements.

Functional testing mainly involves black box testing and it is not concerned about the source code of the application. This testing checks the user interface, APIs, Database, Security, Client/Server and other functionality of the application under test. The testing can be done either manually or using automation.

Functional System Testing : A requirement that specifies a function that a system or system component must perform. A requirement may exist as a text document and/or a model There is two types of techniques. Requirement Based Functional Testing Process Based Testing

Q.11 What is Non-Functional Testing?

Ans. NON-FUNCTIONAL TESTING is defined as a type of software testing to check non-functional aspects (performance, usability, reliability, etc.) of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.

An excellent example of a non-functional test would be to check how many people can simultaneously log into a software. Non-functional testing is equally important as functional testing and affects client satisfaction. 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 they 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

Q.12 What is GUI Testing?

Ans. Graphical User Interface (GUI) testing is the process of testing the system's GUI of the system under test. GUI testing involves checking the screens with controls like menus, buttons, icons, and all types of bars – tool bars, menu bars, dialog boxes, and windows, etc.

Wherein we can check the below listed points.

- 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 the screen
- Check Font used in application is readable
- Check the alignment of the text is proper
- Check the color of the font and warning messages is aesthetically pleasing
- Check that the images have good clarity
- Check that the images are properly aligned
- Check the positioning of GUI elements for different screen resolution.

Q.13 What is ad hoc testing?

Ans. Ad hoc testing is an informal or unstructured software testing type that aims to break the testing process in order to find possible defects or errors at the earliest possible stage. Ad hoc testing is done randomly, and it is usually an unplanned activity that does not follow any documentation or test design techniques to create test cases.

Ad hoc testing does not follow any structured method of testing and it is randomly done on any part of application. The main aim of this testing is to find defects by random checking. Adhoc testing can be achieved with the software testing technique called Error Guessing. Error guessing can be done by people having enough experience with the system to "guess" the most likely source of errors.

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

Q.14 What is white box testing, and what are the types of white box testing?

Ans. White Box Testing is a software testing technique in which the internal structure, design, and coding of software are tested to verify the flow of input-output and to improve design, usability, and security. In white box testing, code is visible to testers, so it is also called clear box testing, open box testing, transparent box testing, Code-based testing, and glass box testing. It is one of two parts of the Box Testing approach to software testing. Its counterpart, blackbox testing, involves testing from an external or end-user perspective. On the other hand, white box testing in software engineering is based on the inner workings of an application and revolves around internal testing.

Types of White Box Testing

White box testing encompasses several testing types used to evaluate the usability of an application, block of code, or specific software package. They are listed below.

- Unit Testing: It is often the first type of testing done on an application. Unit testing is performed on each unit or block of code as it is developed. Unit testing is essentially done by the programmer. As a software developer, you develop a few lines of code, a single function or an object, and test it to make sure it works before continuing. Unit testing helps identify a majority of bugs, early in the software development lifecycle. Bugs identified in this stage are cheaper and easier to fix.
- Testing for Memory Leaks: Memory leaks are a leading cause of slower running applications. A QA specialist who is experienced at detecting memory leaks is essential in cases where you have a slow running software application.

Apart from the above, a few testing types are part of both black box and white box testing. They are listed as below.

- White Box Penetration Testing: In this testing, the tester/developer has full access to the application's source code, detailed network information, IP addresses involved, and all server information the application runs on. The aim is to attack the code from several angles to expose security threats
- White Box Mutation Testing: Mutation testing is often used to discover the best coding techniques to use for expanding a software solution.

Q.15 What is black box testing? What are the different black box testing techniques?

Ans. Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details, and internal paths. Black Box Testing mainly focuses on the input and output of software applications, and it is entirely based on software requirements and specifications. It is also known as behavioral testing.

The above Black-Box can be any software system you want to test. For example, an operating system like Windows, a website like Google, a database like Oracle, or even your own custom application. Under Black Box Testing, you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation. Consider the following video tutorial-

Types of Black Box Testing

There are many types of black box testing, but the following are the most prominent ones. –

- Functional testing – This black box testing type is related to the functional requirements of a system; it is done by software testers.
- Non-functional testing – This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, and usability.
- Regression testing – Regression testing is done after code fixes, upgrades, or any other system maintenance to check that the new code has not affected the existing code.

Q.16 Mention what are the categories of defects?

Ans. A software bug arises when the expected result don't match with the actual results. It can also be an error, flaw, failure, or fault in a computer program. Most bugs arise from mistakes and errors made by developers, architects.

Common Types of Defects

The following are the common types of defects that occur during development:

- Arithmetic Defects
- Logical Defects
- Syntax Defects
- Multithreading Defects
- Interface Defects
- Performance Defects

Q.16 Mention what bigbang testing is?

Ans. 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.

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

Q.17 What is the purpose of exit criteria?

Ans. 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)

Exit criteria typically measure:

- Measures, such as coverage of requirements or of code or risk coverage,
- estimates of defect density or reliability measures. (e.g., how many defects are open by category)
- Cost
- Residual risks, such as defects not fixed or lack of test coverage in certain areas.
- Schedules - such as those based on time to market.

Q. 18 When should "regression testing" be performed?

Ans. Ideally, regression testing should be performed whenever your codebase has been modified or altered in any way, as well as to verify any previously discovered issues marked as fixed. The more often, the better: frequent partial regression testing will help your developers fix the reported defects on time, and your project 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 codebase, 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 for running 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.

Q.19 What are the 7 key principles? Explain in detail?

Ans. General Testing Principles

- Testing shows the presence of defects.
- Exhaustive Testing is Impossible!
- Early Testing
- Defect Clustering
- The Pesticide Paradox
- Testing is context dependent.
- Absence of Errors Fallacy

1. Testing shows the presence of defects: Testing can show the existence of defects but cannot remove them. Testing reduces the chance that undetected defects will remain in the software, but even if none exist, it does not prove that it is correct. We test to detect issues. The probability of undiscovered problems still being present in a system increases as we detect more flaws.

2. Exhaustive Testing is Impossible! : It is impossible to test everything, including all potential input and precondition combinations. Therefore, we can use risks and priorities to focus testing efforts rather than perform exhaustive testing. For instance, if an application had 15 input fields on a single screen, each with five possible values, it would take 30 517 578 125 (515) tests to cover all possible possibilities.

It is quite unlikely that this many tests could be performed within the project's time constraints. Therefore, one of the most crucial tasks and the main justification for testing in any project is accessing and managing risk. We now understand that we cannot test every scenario, including all possible input and precondition combinations. We must prioritise our testing efforts using a risk-based approach, in other words.

3. Early Testing : The life cycle of developing software or a system should begin with testing operations as early as feasible, with a clear focus on the goals. as early as feasible within the development life cycle. These efforts must be focused on clearly stated goals, as described in the Test Strategy. The testing process doesn't begin after the code has been developed, as we stated in our definition of testing.

4. Defect Clustering : The majority of defects found during pre-release testing are found in a limited number of modules, and they are also the main cause of operational issues. Defects are "clustered," which refers to the fact that they are typically confined to a small number of modules and are not distributed uniformly throughout the system. Similar to this, the majority of system operational faults are typically limited to a few modules.

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

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

6. Testing is Context Dependent : Testing is done differently in different contexts Different kinds of sites are tested differently. For example , Safety – critical software is tested differently from an e-commerce site. Whilst, Testing can be 50% of development costs, in NASA’s Apollo program it was 80% testing 3 to 10 failures per thousand lines of code (KLOC) typical for commercial software 1 to 3 failures per KLOC typical for industrial software 0.01 failures per KLOC for NASA Shuttle code! Also different industries impose different testing standards

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

Q. 20 Difference between QA v/s QC v/s Tester

Ans.

| Sr.No. | Quality Assurance | Quality Control | Testing |
|---|---|---|---|
| 1 | Activities which ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements. | Activities which ensure the verification of developed software with respect to documented (or not in some cases) requirements. | Activities which ensure the identification of bugs/error/defects in the Software. |
| 2 | Focuses on processes and procedures rather than conducting actual testing on the system. | Focuses on actual testing by executing Software with intend to identify bug/defect through implementation of procedures and process | Focuses on actual testing |

| | | | |
|---|---|---|---|
| 3 | Process oriented activities. | Product oriented activities | Product oriented activities. |
| 4 | Preventive activities | It is a corrective process. | It is a preventive process |
| 5 | It is a subset of Software Test Life Cycle (STLC). | QC can be considered as the subset of Quality Assurance. | Testing is the subset of Quality Control |

## Q.21 Difference between Smoke and Sanity.

## Ans.

| Sr. No. | Smoke Testing | Sanity Testing |
|---|---|---|
| 1 | Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine | Sanity Testing is done to check the new functionality / bugs have been fixed |
| 2 | The objective of this testing is to verify the "stability" of the system in order to proceed with more rigorous testing | The objective of the testing is to verify the "rationality" of the system in order to proceed with more rigorous testing |
| 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 |

## Q.22 Difference between verification and Validation

Ans.

| 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 work-products (not the actual final product) of a development phase to determine whether they meet the specified requirements for that phase. |
| **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.<br>In other words, to demonstrate that the product fulfills 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<br>Walkthroughs<br>Inspections | Testing |

## Q.23 Explain the types of performance testing.

Ans. There are several types of performance testing, as described below.

- Load testing
- Stress testing
- Endurance testing
- Spike testing
- Volume testing
- Scalability testing

1. Load testing – checks the application's ability to perform under anticipated user loads. The objective is to identify performance bottlenecks before the software application goes live.

2. Stress testing – involves testing an application under extreme workloads to see how it handles high traffic or data processing. The objective is to identify the breaking point of an application.
3. Endurance testing – is done to make sure the software can handle the expected load over a long period of time.
4. Spike testing – tests the software's reaction to sudden large spikes in the load generated by users.
5. Volume testing – Under Volume Testing large no. of. Data is populated in a database and the overall software system's behavior is monitored. The objective is to check a software application's performance under varying database volumes.
6. Scalability testing – The objective of scalability testing is to determine the software application's effectiveness in "scaling up" to support an increase in user load. It helps plan capacity additions to your software system

Q.24 What is Error, Defect, Bug and failure?

Ans.

| Comparison basis | Bug | Defect | Error | Failure |
|---|---|---|---|---|
| Definition | It is an informal name specified to the defect. | The defect is the difference between the actual outcomes and expected outputs. | An is a mistake made in the code; that's why we cannot execute or compile the code. | If the software has lots of defects, it leads to failure or causes failure. |
| Raised by | The test engineers submitted the bug. | The test identifies the defect. And it was also solved by the developer in the development phase or stage. | The developers and automation test engineers raised the error | The failure was found by the manual test engineer through the development cycle. |
| Different types | Different types of bugs are as follows:<br>Logic bugs<br>Algorithmic bugs<br>Resource bugs | Different types of defects are as follows:<br>Based on priority:<br>High<br>medium<br>Low<br>And based on the severity:<br>Critical<br>Major | Different types of errors are as below:<br>Syntactic Error<br>User interface error<br>Flow control errors<br>Error handling error<br>Calculation error<br>Hardware error<br>Testing Error | ---------- |

| | | | | |
|---|---|---|---|---|
| | | Minor<br>Trivial<br>Different types of errors are as below:<br>Syntactic Error<br>User interface error<br>Flow control errors<br>Error handling error<br>Calculation error<br>Hardware error<br>Testing Error | | |
| Reasons behind | The following are reasons which may cause the bugs:<br>Missing coding<br>Wrong coding<br>Extra coding | The below reasons lead to the defects: giving incorrect and wrong inputs.<br>Dilemmas and errors in the outside behavior and inside structure and design.<br>An error in coding or logic affects the software and causes it to breakdown or fail. | The reasons for having an error are as follows:<br>Errors in the code.<br>The Mistake of some values.<br>If a developer is unable to compile or run a program successfully.<br>Confusions and issues in programming.<br>Invalid login, loop, and syntax.<br>There is inconsistency between actual and expected outcomes.<br>Blunders in design or requirement actions.<br>Misperception in understanding the requirements of the application | The following are some of the most important reasons behind the failure:<br>Environmental condition<br>System usage<br>Users<br>Human error |
| A way to prevent the reasons | The following are the ways to stop | With the help of the following, we | Below are ways to prevent the errors: | The way to prevent failure is as follows: |

| | | | | |
|---|---|---|---|---|
| | the bugs:<br>Test-driven development.<br>They offer programming language support.<br>Adjusting, advanced, and operative development procedures.<br>Evaluating the code systematically | can prevent the Defects:<br>Implementing several innovative programming methods.<br>use of primary and correct software development techniques.<br>Peer review<br>It is executing consistent code reviews to evaluate its quality and correctness. | Enhance the software quality with system review and programming.<br>Detect the issues and prepare a suitable mitigation plan.<br>Validate the fixes and verify their quality and precision. | Confirm re-testing.<br>Review the requirements and revisit the specifications.<br>Implement current protective techniques.<br>categorize and evaluate errors and issues. |

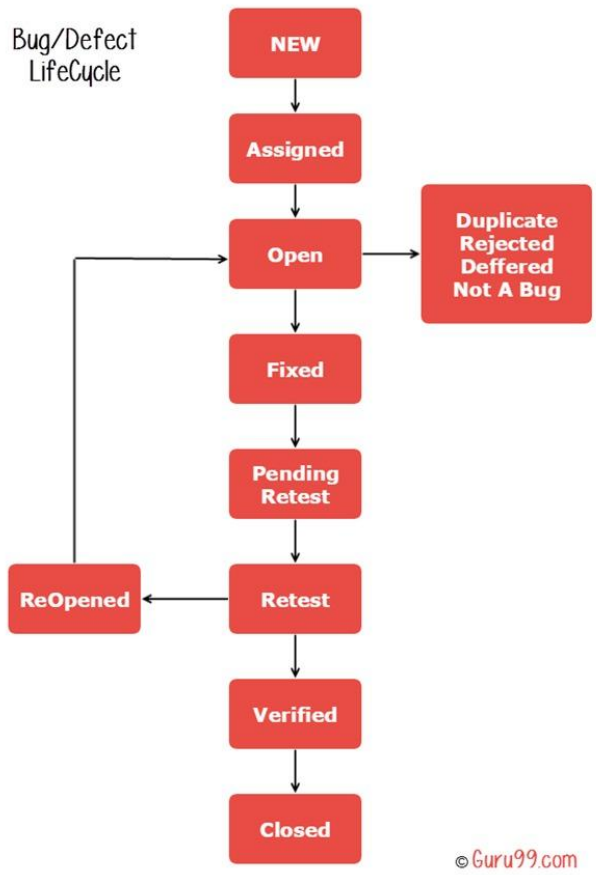
Q.25 Difference between Priority and Severity

Ans.

| Sr. No. | Severity | Priority |
|---|---|---|
| 1 | Severity is a parameter to denote the impact of a particular defect on the software. | Priority is a parameter to decide the order in which defects should be fixed. |
| 2 | Severity means how severe the defect is affecting the functionality. | Priority means how fast a defect has to be fixed. |
| 3 | Severity is related to the quality standard. | Priority is related to scheduling to resolve the problem. |
| 4 | A testing engineer decides the severity level of the defect. | The product manager decides the priorities of defects. |
| 5 | Its value is objective. | Its value is subjective. |
| 6 | Its value doesn't change from time to time. | Its value changes from time to time. |
| 7 | Severity is of 5 types: Critical, Major, Moderate, Minor, and Cosmetic. | Priority is of 3 types: Low, Medium, and High. |

Q. 26 What is the Bug Life Cycle?

Ans. The journey that a defect cycle–also referred to as a bug life cycle–goes through during its lifetime is known as the defect life cycle. Since it is managed by the software testing process and also dependent on the tools used, it varies from organization to organisation and also from project to project.




Defect States

**#1) New:** This is the first state of a defect in the Defect Life Cycle. When any new defect is found, it falls in a 'New' state, and validations & testing are performed on this defect in the later stages of the Defect Life Cycle.

**#2) Assigned:** In this stage, a newly created defect is assigned to the development team to work on the defect. This is assigned by the project lead or the manager of the testing team to a developer.

**#3) Open:** Here, the developer starts the process of analyzing the defect and works on fixing it, if required.

If the developer feels that the defect is not appropriate then it may get transferred to any of the below four states namely **Duplicate, Deferred, Rejected, or Not a Bug**-based upon a specific reason. We will discuss these four states in a while.

**#4) Fixed:** When the developer finishes the task of fixing a defect by making the required changes then he can mark the status of the defect as "Fixed".

**#5) Pending Retest:** After fixing the defect, the developer assigns the defect to the tester to retest the defect at their end, and until the tester works on retesting the defect, the state of the defect remains in "Pending Retest".

**#6) Retest:** At this point, the tester starts the task of retesting the defect to verify if the defect is fixed accurately by the developer as per the requirements or not.

**#7) Reopen:** If any issue persists in the defect, then it will be assigned to the developer again for testing and the status of the defect gets changed to 'Reopen'.

**#8) Verified:** If the tester does not find any issue in the defect after being assigned to the developer for retesting and he feels that if the defect has been fixed accurately then the status of the defect gets assigned to 'Verified'.

**#9) Closed:** When the defect does not exist any longer, then the tester changes the status of the defect to "Closed".

Q.27 Explain the difference between Functional testing and Non Functional testing

Ans.

| Functional Testing | Non-Functional Testing |
|---|---|
| testing carried out using a review of the functionality of a component or system's specification.<br>Functional testing is carried out first. | testing a system or component's properties related to functionality.<br>It should be carried out following functional testing. |
| Describes what the product does Easy to do manual testing | Describes how good the product works Tough to do manual testing |
| Types of Functional testing are<br>1. Unit Testing<br>2. Smoke Testing<br>3. Sanity Testing<br>4. Integration Testing<br>5. White box testing<br>6. Black Box testing<br>7. User Acceptance testing<br>8. Regression Testing | Types of Non-functional testing are<br>1. Performance Testing<br>2. Load Testing<br>3. Volume Testing<br>4. Stress Testing<br>5. Security Testing<br>6. Installation Testing<br>7. Compatibility Testing<br>8. Migration Testing |

Q.28 What is the difference between the STLC (Software Testing Life Cycle) and SDLC (Software Development Life Cycle)?

Ans.

| SDLC | STLC |
|---|---|
| builds a product as its main objective. | focuses on product testing |
| parent procedure. | A child of SDLC process. |
| building a product to meet consumer needs. | confirming that the product performs as expected. |
| Before testing, SDLC stages are finished. | After the completion of the SDLC phases, STLC phases begin. |
| The ultimate objective is to provide users with a high-quality product. | The final objective is to identify and correct any faults or problems. |

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

Ans.

| S.NO | Test Case | Test Scenarios |
|---|---|---|
| 1. | The test case is a detailed document that provides information about the testing strategy, testing process, preconditions, and expected output. | The test scenarios are those derived from the use case and give the one-line information about what to test. |
| 2. | It includes all the positive and negative inputs, navigation steps, expected results, pre and post conditions, etc. | Test scenarios are one-liner statements, but they are connected to several test cases. |
| 3. | These are low-level actions. | These are high-level actions. |
| 4. | The main objective of writing the test case is to verify the test scenario by implementing steps. | The test scenario's main objective is to cover end to end functionality of a software application. |

| | | |
|---|---|---|
| 5. | It takes more time as compared to test scenarios. | It takes less time as compared to testing cases. |
| 6. | The test cases are hard to maintain. | Test scenarios are easy to maintain due to their high-level design. |
| 7. | The test case will help with our in-depth testing of the application. | The test scenario will help us with an agile way of testing throughout the functionality. |
| 8. | The test case is based on the basics of ”how to be tested”. | The test scenarios are based on the basic to ”What to be tested”. |
| 9. | To write the tests, we need additional resources to create and perform test cases. | Fewer resources are sufficient to write test scenarios as compared to the test cases. |
| 10. | In a test case, we mainly focus on the document details. | In test scenarios, we will focus on thinking and discussing details. |
| 11. | It can be obtained from test scenarios. | It can be obtained from the use cases. |
| 12. | If the test engineer is working offsite, then the creation of test cases is significant. | If the tester is particularly working in Agile methodology, then creating test scenarios helps us in a time-sensitive situation. |
| 13. | Writing test cases is a one-time attempt that can be used in the future at the time of regression testing. | The test scenarios are a time saving process; that’s why it is preferred by the |

| | | |
|---|---|---|
| | | A new generation of software testing communit |

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

Ans. A Test Plan is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables, 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.

This section have to provide information about the document. It's mostly kept to give the reader an understanding of what the test plan will cover.

1. Test Planning : (A document describing the scope, approach, resources and schedule of intended test activities.)
2. Test Plan Strategy
3. Test Plan Factor
4. Test Plan Activities
5. Exit Criteria

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

Ans. The agile SDLC model contains incremental and iterative process models with a focus on process flexibility and customer satisfaction through the quick delivery of functional software products.The product is divided into small incremental builds via agile methods. These builds are provided in iterations Every iteration usually lasts between one and three weeks.

Cross-functional teams collaborate on numerous tasks, including planning, requirements analysis, design, coding, unit testing, and acceptability testing, simultaneously throughout each iteration. A functioning product is presented to the client and other key stakeholders at the conclusion of the iteration.

Agile definition.

Every project should be handled differently, according to the agile model, and existing methods should be modified as necessary to best meet the project's needs.

In agile, tasks are broken down into time boxes (short time frames) to provide certain features for a release. Working software builds are given at the end of each iteration, according to an iterative approach. The final build has all the features the user has requested, whereas each version adds new functionality incrementally.

Q.32 Explain the difference between Authorization and Authentication in Web testing

Ans.

| Authentication | Authorization |
|---|---|
| Users' identities are verified as part of the authentication process in order to give them access to the system. | While in authorization process, a the person's or user's authorities are checked for accessing the resources. |
| In the authentication process, users or persons are verified. | While in this process, users or persons are validated. |
| It is done before the authorization process. | While this process is done after the authentication process. |
| It needs usually the user's login details. | While it needs the user's levels of authority or security. |
| Authentication determines whether the person is user or not. | While it determines **What permission does the user have?** |
| Generally, transmit information through an ID Token. | Generally, transmit information through an Access Token. |
| The OpenID Connect (OIDC) protocol is an authentication protocol that is generally in charge of user authentication process. | The OAuth 2.0 protocol governs the overall system of user authorization process. |
| Popular Authentication Techniques-<br>● Password-Based Authentication<br>● Passwordless Authentication<br>● 2FA/MFA (Two-Factor Authentication / Multi-Factor Authentication)<br>● Single sign-on (SSO)<br>● Social authentication | Popular Authorization Techniques-<br>● Role-Based Access Controls (RBAC)<br>● SON web token (JWT) Authorization<br>● SAML Authorization<br>● OpenID Authorization<br>● OAuth 2.0 Authorization |
| The authentication credentials can be changed in part as and when required by the user. | The authorization permissions cannot be changed by user as these are granted by the owner of the system and only he/she has the access to change it. |
| The user authentication is visible at user end. | The user authorization is not visible at the user end. |
| The user authentication is identified with username, | The user authorization is carried out through the |

| | |
|---|---|
| password, face recognition, retina scan, fingerprints, etc. | access rights to resources by using roles that have been pre-defined. |
| **Example**: Employees in a company are required to authenticate through the network before accessing their company email. | **Example:** After an employee successfully authenticates, the system determines what information the employees are allowed to access. |

Q.33 What are the common problems faced in Web testing

Ans. During web testing, server issues are the most frequent problems raised. Either the server is down or maintenance is being done on it. Database issues, where the connection to the database might be lost, are the following issue that is frequently seen during online testing. Occasionally, while testing a web application, there may be hardware compatibility issues.

Q.34 To create HLR & TestCase of Facebook Login Page

Ans.