**Module 2: Manual Testing**

1. **What is Error, Defect, Bug and failure?**

**Ans. ERROR:** A mistake in coding is called “ERROR”.

**DEFECT:** Error found by tester is called “Defect”.

**BUG**: Defect accepted by development team is called “Bug”.

**FAILURE:** Build Does not meet the requirements then it is called “FAILURE”.

**2. Difference Between QA v/s QC v/s TESTING.**

**Ans.**

|  |  |  |  |
| --- | --- | --- | --- |
| **TOPIC** | **QA** | **QC** | **Testing** |
| **Full form** | QA Stands for Quality Assurance. | QC stands for Quality Control | Software Testing |
| **Focus:** | Preventing defects and ensuring high-quality software development processes. | Identifying defects and ensuring high-quality software products. | Verifying that software meets the required specifications and standards. |
| **Approach:** | Proactive, process-oriented, and involves the entire development team | Reactive, product-oriented, and involves a testing team | Varies depending on the testing methodology (e.g., Agile, Six Sigma) |
| **Activities:** | Process improvement, test planning, reviews, and defining standards and methodologies | Test execution, defect reporting, and inspection | Manual or automated testing, depending on the project requirements |
| **Goal:** | High-quality software development process | High-quality software product | Measuring the quality of software and identifying defects |
| **Stage in SDLC** | Throughout the lifecycle | Later stages | Throughout the lifecycle, but typically in the later stages |

**3. What is seven key principles? Explain in detail?**

**Ans.** Software testing is a complex activity, and a tester should understand these testing principles to have a broader understanding of the testing process.

**7 key principles of SDLC**

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 depending

7. Absence of error fallacy

1. **Testing Shows Presence of defects**

* Testing can show that defects are present but cannot prove that there are no defects.
* However, even if testing doesn’t find any defects, it doesn’t guarantee that the software is completely defect-free.
* We test to find faults.

**2. Exhaustive Testing is impossible!**

* Exhaustive testing, which is also known as complete testing, occurs when all the testers in your team are exhausted and when all the planned tests have been executed.
* It is a quality assurance testing technique in which all scenarios or data is tested for testing.

**3. Early Testing**

* In software development, early testing helps to identify and eliminate defects in the early stages of the Software Development Life Cycle.
* Early testing ensures that the software product is of high quality and reliable, reducing the risk of errors and defects.

**4. Defect clustering**

* Defect clustering refers to a situation where a small number of areas or modules in a software application have a large number of bugs or errors.
* This means that a few specific parts of the application are more prone to defects, while other parts are relatively bug-free.
* For example, think of it like a game where a few players on a team keep making mistakes, causing the team to lose, while other players are doing well.
* In software testing, defect clustering is a common incident where a small number of defects or errors cause a large percentage of system failures.

**5. 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 regular reviewed and revised
* Testing identifies bugs, and programmers respond to fix them.
* As bugs are eliminated by the programmers, the software improves.
* As software improves the effectiveness of previous tests erodes.

**6. Testing is context depending**

* Different kinds of sites are tested differently.
* For example , Safety – critical software is tested differently from an E-commerce site.
  + 3 to 10 Failures per thousand lines of code(KLOC) typical for commercial s/w.
  + 1 to 3 failures per kilos typical for industrial s/w
  + 0.01 failures per kloc for NASA shuttle code.
* Also different industries impose different testing standards.

**7. Absence of error fallacy**

* let us understand this with the help of an example.   
  Imagine, our company developed a school management software. [Complete testing](https://www.oodlestechnologies.com/blogs/Software-Testing-Tips) of the software is done and about 99% of the defects which were identified have been fixed and also retesting on them is done. This report has been submitted to management and this has given them pretty much confidence about the product quality w.r.t. defects.  
  Now when the demonstration of software is given to the client, he gives his feedback saying “Though it is called defect free, but still this is not what I required.AS I wanted a simple UI which can handle the user load.”
* Now before receiving this feedback, everyone in the Testing team was very much confident and sure about the 'quality' of product (**i.e. absence of errors**) but in the end, it proved to be false (**i.e. fallacy**). This straightaway means that the system is not usable by the client as it does not fulfill his expectations.
* Therefore, it is very much important to note that the Client's requirements and expectations are as important as the quality of product.

**4. Difference between verification and Validation**

**Ans.**

|  |  |  |
| --- | --- | --- |
| Topic | Verification | Validation |
| Definition: | Checking if the software is developed according to specifications, designs, and requirements. | Ensuring the software meets the customer’s needs and expectations. |
| Focus: | Ensuring the software is built correctly, i.e., “Was it built right?” | Verifying the software is usable, functional, and meets the customer’s requirements, i.e., “Is the right thing built?” |
| Activities | Reviewing documents, designs, and code; checking for conformance to specifications; static testing (e.g., inspections, code reviews). | Testing the actual software for functionality, usability, and performance; dynamic testing (e.g., black box testing, white box testing). |
| Goal: | To ensure the software meets its intended design and requirements. | Toensure the software delivers the desired value to the customer. |
| Example: | Imagine a team developing a mobile banking app. During verification, they review the requirements and design documents to ensure all specified features (e.g., fund transfer, account balance check) are included and correctly detailed. | Continuing the mobile banking app example, during validation, the team tests the app to ensure it allows customers to easily transfer funds, check their account balances, and access their transaction history. The team also evaluates the app’s usability and performance to ensure it meets customer expectations. |

**5. Explain the difference between Functional testing and Non-Functional testing**

**Ans.**

|  |  |
| --- | --- |
| **Functional Testing** | **Non-Functional testing** |
| It verifies the operations and actions of an application. | It verifies the behaviour of an application. |
| It is based on requirements of customer. | It is based on expectations of customer. |
| It helps to enhance the behaviour of the application. | It helps to improve the performance of the application |
| Functional testing is easy to execute manually. | It is hard to execute non-functional testing manually. |
| It tests what the product does. | It describes how the product does. |
| examples: Unit testing, smoke testing, integration testing, regression testing. | examples : performance testing, load testing, stress testing, scalability testing. |

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

**Ans.** Boundary value testing is a software testing technique used to identify errors at the boundaries of input values rather than within the range of values.

The core idea is that errors are more likely to occur at the edges of input ranges rather than in the middle.

**Here's how boundary value testing generally works:**

**Identify Boundaries**: Determine the boundaries for the input data. For example, if an input field accepts values from 1 to 100, the boundaries are 1 and 100.

**Test at Boundaries**: Test the system with values exactly at the boundaries (e.g., 1 and 100).

**Test Just Outside Boundaries**: Test with values just outside the boundaries (e.g., 0 and 101).

**Test Just Inside Boundaries**: Test with values just inside the boundaries (e.g., 2 and 99).

**Test Multiple Boundaries**: For ranges that have multiple boundaries or constraints, test combinations of boundary values to ensure that the system handles all scenarios properly.

**Why Boundary Value Testing is Important**

**Error Detection:** Many software bugs occur at the boundaries of input ranges. By focusing on these areas, boundary value testing increases the likelihood of detecting these types of errors.

**Efficient Testing:** Testing boundaries often requires fewer test cases than exhaustive testing of all possible inputs while still providing significant coverage.

**Validation of Conditions:** Ensures that the system correctly handles edge cases and enforces constraints properly.

**Example:**

If a system accepts ages from 18 to 65:

Boundary values: 18 (lower boundary) and 65 (upper boundary).

Just outside boundaries: 17 (just below lower boundary) and 66 (just above upper boundary).

Just inside boundaries: 19 (just above lower boundary) and 64 (just below upper boundary).

By testing these values, you can ensure the system behaves as expected across the full range of valid and invalid inputs.

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

**Ans.** Big bang integration testingis a testing approach where all components or modules are integrated and tested as a single unit.

* This is done after all modules have been completed and before any system-level testing is performed.
* This is in contrast to incremental integration testing, in which components are tested one at a time or in small groups.
* This approach is typically used when there is a tight deadline for delivering the software product, and all development teams are working in parallel on their respective components.
* **For example**,consider a simple system with three modules A, B, and C. Module A has been tested and found to be working correctly. The same is true for modules B and C. To test the system as a whole, all three modules are integrated and tested together.
* In some cases, big bang integration testing may be the only option available. For example, if the system to be tested is too complex or too large to test piecemeal, then big bang integration testing is the only option.

**8. 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 an early possible stage.

- Ad hoc testing is done randomly and it is usually an unplanned activity which does not follow any documentation and test design techniques to create test cases.

- 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 the people having enough experience on the system to “guess” the most likely source of errors.

- Adhoc testing has – No Documentation, No Test cases, No Test Design.

Types of Adhoc Testing:

**1. Buddy Testing** – Buddy testing is a type of Adhoc testing where two bodies will be involved one is from the Developer team and one from the tester team. So that after completing one module and after completing Unit testing the tester can test by giving random inputs and the developer can fix the issues too early based on the currently designed test cases.

**2. Pair Testing** – Pair testing is a type of Adhoc testing where two bodies from the testing team can be involved to test the same module. When one tester can perform the random test, another tester can maintain the record of findings. So, when two testers get paired, they exchange their ideas, opinions, and knowledge so good testing is performed on the module.

**3. Monkey Testing / Gorilla Testing** – Monkey testing is a type of Adhoc testing in which the system is tested based on random inputs without any test cases the behaviour of the system is tracked and all the functionalities of the system are working or not is monitored. As the randomness approach is followed there is no constraint on inputs so it is called Monkey testing.

**9. What is white box testing and list the techniques (coverage) of white box testing?**

**Ans.**

* **White box Testing :** Testing based on an analysis of the internal structure of the component or system.
* White box testing, also known **as clear box testing, glass box testing, or structural testing**, is a software testing method where the tester has full visibility into the internal workings of the application being tested.
* Unlike black box testing, where the **tester only focuses on the inputs and outputs of the application without knowing its internal structure**, white box testing involves examining and testing the internal logic, structure, and code of the software.

**TYPES OF COVERAGE (Whitebox testing)**

1. Statement coverage
2. Condition coverage
3. Decision coverage

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

**Ans.**

* **Black box testing** isa software testing methodology in which the tester evaluates the functionality of an application without any knowledge of its internal code or structure.
* Testers provide inputs to the system and check if the outputs are as expected, without concern for the internal workings of the application.

**TECHNIQUES OF BLACK BOX TESTING**

 **Equivalence Partitioning**:

 **Boundary Value Analysis**:

 **Decision Table Testing**:

 **State Transition Testing**:

 **Use Case Testing**:

 **Exploratory Testing**:

**11. What is Integration testing?**

**Ans.**

* **Integration Testing** is a level of the software testing process where individual units are combined and tested as a group.
* The **purpose** of this level of testing is **to expose faults in the interaction between integrated units**.
* Integration testing is a critical phase in the software development lifecycle, focusing on verifying that different modules or systems interact correctly and work together seamlessly.
* It aims to uncover issues related to interfaces and data flow between integrated components, ensuring that the software functions correctly as a whole.

**There are two levels of Integration testing :**

* Component integration testing
* System Integration Testing

**12. What is component testing?**

**Ans.**

* A Unit/Component testing is the smallest testable part of an application like functions / procedures , classes , interfaces.
* The goal of unit testing is to isolate each part of the program and show that the individual parts are correct.
* A unit test provides a strict, written contract that the piece of code must satisfy. As a result, it affords several benefits.
* Unit tests find problems early in the development cycle.
* **Component / Unit testing is performed by using the White Box Testing Method.**

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

**Ans.**

* The purpose of the exit criteria is to define when we stop testing.
* Successful testing of integration application.
* Executed test cases are documented.
* All high prioritized bugs fixed and closed.
* End of all testing (For example: Product go for live).
* End of phases of testing (For example: Hand over from system test to UAT).

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

**Ans.**

* Functional system testing is a type of testing that evaluates a system's ability to perform its intended functions and meet its requirements.
* It focuses on verifying that the system behaves as expected, without considering the internal workings or implementation details.

During functional system testing, testers:

1. Examine the system's functionality against its specifications and requirements.

2. Validate that the system performs its intended functions correctly.

3. Test the system's user interface, APIs, and other external interactions.

4. Verify that the system produces the expected outputs for given inputs.

5. Check for errors, exceptions, and boundary cases.

The goals of functional system testing include:

1. Ensuring the system meets its functional requirements.

2. Identifying defects or bugs in the system's functionality.

3. Verifying that the system works as expected in real-world scenarios.

4. Providing confidence that the system is ready for deployment or release.

Functional system testing is typically performed by testers or quality assurance teams, and it's an essential part of the software development life cycle.

**15. What is Non-functional testing?**

**Ans.**

* Non-functional testing helps ensure that the system meets the required standards, is user-friendly, and can handle real-world scenarios.
* It's typically performed in conjunction with functional testing to provide a comprehensive view of the system's quality.
* Some common non-functional testing techniques include:

1. Load testing

2. Stress testing

3. Penetration testing

4. Usability testing

5. Performance testing

6. Compatibility testing

7. Security testing

8. Benchmarking

* By conducting non-functional testing, teams can identify areas for improvement, optimize system performance, and ensure a better overall user experience.

**16. What is Exploratory Testing?**

**Ans.**

Exploratory testing is a concurrent process where

* Test design, execution and logging happen simultaneously
* Testing is often not recorded
* Makes use of experience, heuristics and test patterns
* Testing is based 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
* Is carried out in time boxed intervals
* More structured than Error guessing

Exploratory testing is useful for:

1. Identifying complex, subtle, or unexpected issues.

2. Testing applications with unclear or changing requirements.

3. Evaluating user experience, usability, and accessibility.

4. Supplementing scripted testing with additional coverage.

5. Reducing testing time and increasing test effectiveness.

Exploratory testing requires skilled testers who can think critically, analyze software behavior, and communicate effectively.

**17. What is traceability matrix?**

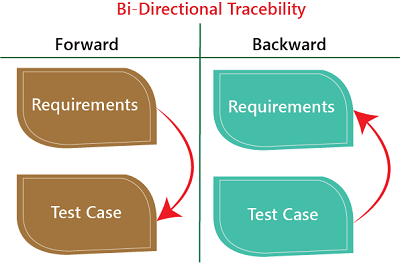
**Ans.** Traceability matrix is a table type document that is used in the development of software application to trace requirements. It can be used for both forward (from Requirements to Design or Coding) and backward (from Coding to Requirements) tracing. It is also known as **Requirement Traceability Matrix (RTM) or Cross Reference Matrix (CRM).**

**Types of Traceability matrix:**

**Forward Traceability** – Mapping of Requirements to Test cases

**Backward Traceability** – Mapping of Test Cases to Requirements

**Bi-Directional Traceability** - A Good Traceability matrix is the References from test cases to basis documentation and vice versa.

****

**18. What is GUI Testing? (UI)**

**Ans.**

* GUI Testing is a software testing type that checks the Graphical User Interface of the Software.
* The purpose of Graphical User Interface (GUI) Testing is to ensure the functionalities of software application work as per specifications by checking screens and controls like menus, buttons, icons, etc.
* The **Graphical user interface testing** process implemented either manually or automatically and repeatedly executed by the **third-parties organization** instead of the developers or the end-users.
* In other words, we can say that GUI testing is an approach in which the application's user interface is tested if the software or an application works as expected relating to the user interface performance.
* **GUI TESTING EXAMPLES**
* Web Based Testing & Desktop Based Testing
* Mobile Based Testing
* Game Based Testing

**19. What is load testing?**

**Ans.**

* Load testing determines the behaviour of the application when multiple users use it at the same time. It is the response of the system measured under varying load conditions.
  + The load testing is carried out for normal and extreme load conditions.
  + **Load testing is a type of performance testing** that simulates a real-world load on a system or application to see how it performs under stress.
  + The goal of load testing is to identify bottlenecks and determine the maximum number of users or transactions the system can handle.
  + It is an important aspect of software testing as it helps ensure that the system can handle the expected usage levels and identify any potential issues before the system is deployed to production.

**For Example ,**Popular toy store Toysrus.com, could not handle the increased traffic generated by their advertising campaign resulting in loss of both marketing dollars, and potential toy sales.

An Airline website was not able to handle 10000+ users during a festival offer.

Encyclopaedia Britannica declared free access to their online database as a promotional offer. They were not able to keep up with the onslaught of traffic for weeks.

**20. What is stress Testing?**

**Ans.**

* Stress testing - System is stressed beyond its specifications to check how and when it fails.
* Performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous input to system or database load.
* Stress testing is used to test the stability & reliability of the system. This test mainly determines the system on its robustness and error handling under extremely heavy load conditions.
* It even tests beyond the normal operating point and evaluates how the system works under those extreme conditions.
* Stress Testing is done to make sure that the system would not crash under crunch situations.
* Stress testing is also known as endurance testing.
* **Most prominent use of stress testing is to determine the limit, at which the system or software or hardware breaks.**
* It also checks whether system demonstrates effective error management under extreme conditions.
* 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.

**Need For Stress Testing**

* During festival time, an online shopping site may witness a spike in traffic, or when it announces a sale.
* When a blog is mentioned in a leading newspaper, it experiences a sudden surge in traffic.
* To check whether the system works under abnormal conditions. Displaying appropriate error message when the system is under stress.
* System failure under extreme conditions could result in enormous revenue loss It is better to be prepared for extreme conditions by executing Stress Testing

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

**Ans.**

* 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
* It should be applied at all Test Levels
* It should be considered complete when agreed completion criteria for regression testing have been met
* Regression test suites evolve over time and given that they are run frequently are ideal candidates for automation
* **Need of Regression Testing :**
* Change in requirements and code is modified according to the requirement
* New feature is added to the software
* Defect fixing
* Performance issue fix

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

**Ans.**

|  |  |
| --- | --- |
| **Smoke Testing** | **Sanity testing** |
| 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 |
| This testing is performed by the developers or tester | Sanity testing is usually performed by testers |
| Smoke testing is usually documented or scripted | Sanity testing is usually not documented and is unscripted |
| Smoke testing is a subset of Regression testing | Sanity testing is a subset of Acceptance testing |
| Smoke testing exercises the entire system from end to end | Sanity testing exercises only the particular  component of the entire system |
| Smoke testing is like general health check up | Sanity testing is like specialized heath check up |
| 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 |

**23. Explain types of Performance testing.**

**Ans.** Performance testing is a type of software testing that focuses on evaluating how a system performs under certain conditions.

Here are the main **Types of performance testing**:

* Load testing
* Stress testing
* Endurance testing
* Spike testing
* Volume testing
* Scalability testing

1. **Load Testing:**

* Purpose: To assess the system's performance under expected user loads.
* Focus: Determines how the application behaves when multiple users access it simultaneously.
* Outcome: Helps identify bottlenecks and determine the system's capacity.

**2.** **Stress Testing:**

* Purpose: To evaluate the system's behavior under extreme conditions, often beyond normal operational capacity.
* Focus: Tests the system's stability and robustness under high stress, such as heavy traffic or data processing loads.
* Outcome: Helps identify the breaking point of the system and how it fails.

**3. Endurance Testing (Soak Testing):**

* Purpose: To determine how the system performs over an extended period.
* Focus: Checks for issues like memory leaks or performance degradation that can occur when the system runs continuously for a long time.
* Outcome: Ensures the system can handle long-term usage without degradation.

**4. Spike Testing:**

* Purpose: To test the system's response to sudden, extreme increases in load.
* Focus: Observes how the system handles unexpected spikes in user activity or data processing.
* Outcome: Helps ensure that the system can recover quickly and maintain performance during and after a spike.

**5. Scalability Testing:**

* Purpose: To assess the system's ability to scale up or down in response to changing demands.
* Focus: Evaluates how the system handles increased load by adding resources, such as more servers or processing power.
* Outcome: Determines the system's scalability limits and how efficiently it can scale.

**6. Volume Testing:**

* Purpose: To test the system's ability to handle large volumes of data.
* Focus: Evaluates performance when processing large datasets or high volumes of transactions.
* Outcome: Identifies potential issues related to data handling, such as slow processing or data corruption.

**24. What is Alpha testing?**

**Ans.**

* 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.

**25. What is Beta Testing? (Field Testing)**

Ans.

* It is always performed by the customers at their own site.
* It is not performed by Independent Testing Team.
* Beta Testing is always open to the market and public.
* It is usually conducted for software product.
* It is performed in **Real Time Environment**.
* It is always performed outside the organization.
* It is also the form of Acceptance Testing.
* Beta Testing (field testing) is performed and carried out by users or you can say people at their own locations and site using customer data.
* It is only a kind of Black Box Testing.
* 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 customers or users area.
* Beta testing can be considered **“pre-release”** testing.
* **Pilot Testing** is testing to product on real world as well as collect data on the use of product in the classroom.

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

**Ans.** There are many software applications / websites, which miserably fail, once launched, due to following reasons –

Where do I click next?

Which page needs to be navigated?

Which Icon or Jargon represents what?

Error messages are not consistent or effectively displayed

Session time not sufficient.

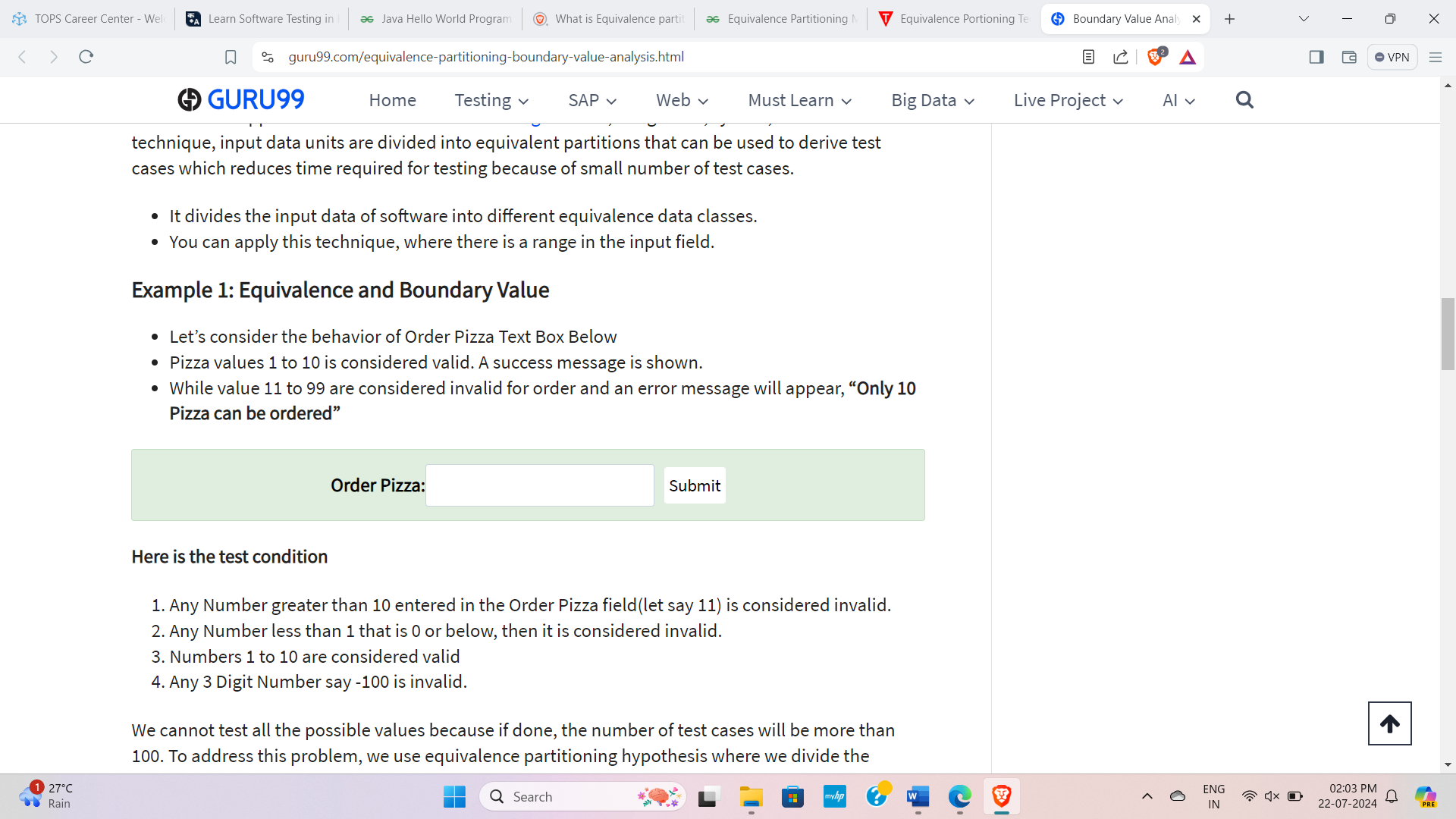
**27. What is Equivalence partitioning testing? (EP) (ECP)**

**Ans. Equivalence Partitioning or Equivalence Class Partitioning** is type of black box testing technique which can be applied to all levels of [software testing](https://www.guru99.com/software-testing.html) like unit, integration, system, etc.In this technique, input data units are divided into equivalent partitions that can be used to derive test cases which reduces time required for testing because of small number of test cases.

* It divides the input data of software into different equivalence data classes.
* You can apply this technique, where there is a range in the input field.

Example 1: Equivalence and Boundary Value

* Let’s consider the behaviour of Order Pizza Text Box Below
* Pizza values 1 to 10 is considered valid. A success message is shown.
* While value 11 to 99 are considered invalid for order and an error message will appear, “Only 10 Pizza can be ordered”

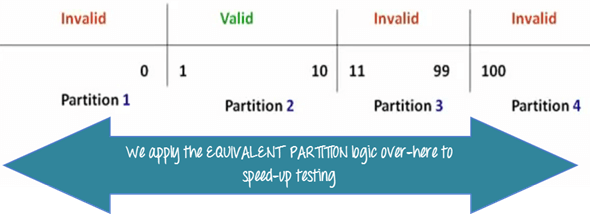


Order Pizza:

Here is the test condition

1. Any Number greater than 10 entered in the Order Pizza field (let say 11) is considered invalid.
2. Any Number less than 1 that is 0 or below, then it is considered invalid.
3. Numbers 1 to 10 are considered valid
4. Any 3 Digit Number say -100 is invalid.

We cannot test all the possible values because if done, the number of test cases will be more than 100. To address this problem, we use equivalence partitioning hypothesis where we divide the possible values of tickets into groups or sets as shown below where the system behaviour can be considered the same.

[](https://www.guru99.com/images/3-2016/032316_0620_Equivalence4.png)

**28. What determines the level of risk?**

**Ans.** A properly designed test that passes, reduces the overall level of Risk in a system

* + Risk – ‘A factor that could result in future negative consequences; usually expressed as impact and likelihood’
  + When testing does find defects, the Quality of the software system increases when those defects are fixed
  + 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
  + Testing should be integrated as one of the Quality assurance activities

**Types of Risk**

* + A Risk could be any future event with a negative consequence .
  + You need to identify the risks associated with your project
  + Risks are of two types
    - Project Risks
    - Product Risk

**Types of Risk Examples**

* Example of Project risk is Senior Team Member leaving the project abruptly.
* Every risk is assigned a likelihood i.e. chance of it occurring, typically on a scale of 1 to 10. Also the impact of that risk is identified on a scale of 1- 10 .
* But just identifying the risk is not enough. You need to identify mitigation. In this case mitigation could be Knowledge Transfer to other team members & having a buffer tester in place
* Example of product risks would be Flight Reservation system not installing in test environment.
* Mitigation in this case would be conducting a smoke or sanity testing. Accordingly you will make changes in your scope items to include sanity testing.

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

**Ans.** The following steps outline the procedure for GUI testing:

**1. Test Script Creation**

Define a test script template, including:

o Test Script ID, Title (part of functionality under test)

o Test Case ID (links to test cases), Test Setup (environment requirements)

o Test Data (values for usability and correctness checks), Procedure (step-by-step instructions)

**2. GUI Element Identification**

Identify all graphical user interface elements, including:

o Menus, Checkboxes, Buttons, Colors, Fonts, Sizes, Icons, Content, Images

**3. Test Case Development**

Create test cases for each GUI element, focusing on:

o Functionality, Usability, Correctness, Error handling

**4. Test Data Preparation**

Prepare test data values for each test case, including:

o Input values, Expected results, Edge cases

**5. Manual Testing**

Perform manual testing using the test scripts, focusing on:

o GUI element behavior, User interaction, Error handling, Usability

**6. Test Reporting**

Document test results, including:

o Pass/Fail status, Defects found, Test environment details

**7. Defect Fixing and Re-testing**

o Fix defects found during testing, Re-run test cases to ensure defects are resolved

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

**Ans.**

**1. Scrum**

Overview: SCRUM is an agile development method which concentrates particularly on how to manage tasks within a team based development environment.

Basically, Scrum is derived from activity that occurs during rugby match. Scrum believes in empowering the development team and advocates working in small teams (say- 7 to 9 members).

It consists of three roles and their responsibilities are explained as follows:

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

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

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

Key Artifacts: Product Backlog, Sprint Backlog, Increment

Key Events: Sprint Planning, Daily Scrum (Stand-up), Sprint Review, Sprint Retrospective

**2. Kanban**

Overview: Kanban is a visual management method that uses boards and cards to represent work items and their status. It focuses on continuous delivery and optimizing workflow.

Key Practices: Visualize work, limit work in progress (WIP), manage flow, make process policies explicit, improve collaboratively.

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

**Ans.**

* A document describing the scope, approach, resources and schedule of intended test activities
* Determining the scope and risks, and identifying the objectives of testing.
* Defining the overall approach of testing (the test strategy), including the definition of the test levels and entry and exit criteria.
* Integrating and coordinating the testing activities into the software life cycle activities:
  + acquisition, supply, development, operation and maintenance
* Making decisions about what to test, what roles will perform the test activities, how the test activities should be done, and how the test results will be evaluated?
* Scheduling test analysis and design activities.
* Scheduling test implementation, execution and evaluation.
* Assigning resources for the different activities defined
  + Defining the amount, level of detail, structure and templates for the test documentation.

Test Planning Activities

* **Approach**: Defining the overall approach of testing (the test strategy), including the definition of the test levels and entry and exit criteria.
* **Integrating and coordinating the testing activities into the software life cycle activities:** acquisition, supply, development, operation and maintenance.

Making decisions about:

* what to test
* who do testing? i.e. what roles will perform the test activities
* when and how the test activities should be done and when they should be stopped (exit criteria – see next slides)
* how the test results will be evaluated Assigning resources for the different tasks defined.

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

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

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

**Ans.**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **SDLC** | **STLC** |
| **Origin** | Software Development Life Cycle | Software Testing Life Cycle |
| **Objective** | The main object of SDLC life cycle is to complete successful development of the software including testing and other phases. | The only objective of the STLC phase is testing. |
| **Requirement Gathering** | In SDLC the business analyst gathers the requirements and create Development Plan | In STLC, the QA team analyze requirement documents like functional and non-functional documents and create System Test Plan |
| **High & Low-Level Design** | In SDLC, the development team creates the high and low-level design plans | In STLC, the test analyst creates the Integration Test Plan |
| **Coding** | The real code is developed, and actual work takes place as per the design documents. | The testing team prepares the test environment and executes them |
| **Maintenance** | SDLC phase also includes post-deployment supports and updates. | Testers, execute regression suits, usually automation scripts to check maintenance code deployed. |

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

**Ans.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr no.** | **Test scenario** | **Test Cases** | **Test Script** |
| **1** | The test scenario is just a document that is detailed and provides details about the assessment method, testing process, precondition, and anticipated output. | Test cases is a step by step procedure to test any functionality of the software application/product | Test script is set of instruction or a short program to test any functionality of software application/product. |
| **2** | The test scenarios are the ones based on the use situation and give one-line information one what to check. | Test cases is a manual approach of software testing. | Test script is an automatic approach of software testing |
| **3** | Test scenarios are one-liner statement, however, it is linked to a few test instances. | It is a set up that is used by the tester to test any specific function of the software product. | It is a program developed by the tester, intended to test any specific function of the software product. |
| **4** | These are high level actions. | Point by point test case configuration encourages tester to test viably. | Automatic testing approach is beneficial for constant execution. |
| **5** | Writing the test scenario’s primary objective is an address end to get rid of functionality of a software program. | Test cases can written manually. | Test scripting is done by scripting format. |
| **6** | It will take less time as compared to test cases | Test case is developed in form of templates. | Test script is developed in form of scripting. |
| **7** | The test scenario will help us in a way that is nimble of through the functionality | If the tester does not have a good understanding of how the program is used or about the recent risks to the program, then it will be difficult to use the test cases properly. | Active software projects frequently change.so testers have to make a continuous effort to update the scripts to match the changes of the new product. |
| **8** | Test scenario are really easy to maintain due to their high level design. | Test case is used in manual testing environment. | Test script is used in automatic testing environment. |
| **9** | The test scenarios tend to be work on the essential to “things to be tested”. | Test cases are classified as delegated, positive, reusable, negative and UI test cases | Test script are characterized as manual test script and automatic test scripts. |