***Module 2 (Manual Testing)Assignment***

1. **What is Exploratory Testing ?**

Exploratory testing is a software testing approach where testers simultaneously design, execute, and learn about the software under test, without relying on pre-defined test cases.

IT is dynamic & flexible method it is useful when specification are lacking when time is constrained.

exploratory testing is a dynamic and adaptable approach that allows testers to leverage their skills and creativity to uncover defects, assess usability, and ensure the quality of the software.

1. **What is traceability matrix ?**

A traceability matrix in testing is a document, typically a table, that maps the relationships between requirements and test cases, ensuring that all requirements are covered by tests.

 It's a crucial tool for quality assurance, helping to verify that every requirement is addressed and that the software is thoroughly tested.

 It's also known as a Requirements Traceability Matrix (RTM) or Cross Reference Matrix (CRM).

Its purpose is to establish a clear link between requirements and test cases, ensuring that all requirements are tested and nothing is missed.

There are types of it

**Forward traceability:** Maps requirements to test cases, ensuring that each requirement has at least one test case.

**Backward traceability**: Maps test cases back to their corresponding requirements, helping to identify the impact of changes.

**Bidirectional traceability:** Combines both forward and backward traceability, providing a comprehensive view of the relationship between requirements and tests.

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

BVA 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.

BVA 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

Always results in two test cases per boundary for valid inputs and three tests cases per boundary for all inputs.

inputs should be in the smallest significant values for the boundary (e.g. Boundary of ‘a > 10.0’ should result in test values of 10.0, 10.1 & 10.2) only applicable for numeric (and date) fields. Ex :0 to 100

2 4

1 100

0 101

Valid invalid

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

Aim is to treat group of inputs as equivalent & to select one representative input to test them all.

Time would be wasted by specifying test cases that covered a range of values within each of the three partitions, unless the code was designed in an unusual way.

There are more effective techniques that can be used to find bugs in such circumstances (such as code inspection)

EP can help reduce the number of tests from a list of all possible inputs to a minimum set that would still test each partition

If the tester chooses the right partitions, the testing will be accurate and efficient

EP is used to achieve good input and output coverage, knowing exhaustive testing is often impossible

It can be applied to human input, input via interfaces to a system, or interface parameters in integration testing

Equivalence partitions where applicable typically in range test.

Ex: >=1 & value <=100 then

1. **What is integration testing ?**

This Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems.

Integration Testing is a level of the software testing process where individual units are combined and tested as a group.

This testing is to expose faults interaction between integrated units.

Test drivers and test stubs are used to assist in Integration Testing.

Integration testing is done by a specific integration tester or test team.

Integration testing tests integration or interfaces between components, interactions to

different parts of the system such as an operating system, file system and hardware or

interfaces between systems.

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?**

The level of risk in software testing is primarily determined by the likelihood of a defect occurring and the impact that defect would have if it did occur.

Essentially, you need to consider both how probable it is that a problem will arise and how serious the consequences would be if it did.

**1. Likelihood:**

* This refers to the probability of a specific risk event happening during testing or in the live environment.
* It can be assessed qualitatively (e.g., high, medium, low) or quantitatively (e.g., a percentage chance of occurrence).
* Factors that influence likelihood include:
  + The complexity of the code or module.
  + The frequency of use of a particular feature or functionality.
  + The stability and maturity of the code.
  + Past defect history of similar code or modules.
  + The quality of the development process.

**2. Impact:**

* This refers to the severity of the consequences if the risk event occurs.
* It can be assessed based on various factors, including:
  + **Business impact:** Financial losses, damage to reputation, legal issues, or loss of customer trust.
  + **Technical impact:** System crashes, data loss, performance degradation, or security breaches.
  + **User impact:** Negative user experience, difficulty using the software, or inability to perform necessary tasks.
* Impact can also be assessed qualitatively (e.g., minor, major, critical) or quantitatively (e.g., estimated cost of recovery or lost revenue).

3. Putting it together:

* By analysing both the likelihood and the impact of potential risks, you can categorize them into different risk levels, such as high, medium, and low.
* This allows you to prioritize your testing efforts, focusing on the areas with the highest risk to ensure that critical issues are identified and addressed early in the development lifecycle.
* A risk matrix can be a helpful tool for visualizing the relationship between likelihood and impact, allowing for a more structured approach to risk assessment.

1. **What is Alpha testing?**

Alpha testing is a type of software testing performed internally by a development team, often including QA specialists and other internal stakeholders, to identify bugs and issues before releasing the product to external users or the public.

It's a form of user acceptance testing (UAT)

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.

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 as the 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.

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.

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.

1. **What is component testing?**

Component testing is a software testing technique that verifies individual software components or modules in isolation to ensure they function correctly according to their specifications.

It focuses on the functionality, usability, and behaviour of each component, independently of other parts of the system.

A minimum software item that can be tested in isolation .

A unit is the smallest testable part of software.

Unit testing is level of software testing process where individual unit/components of software/system are tested.

* **Isolation:**

Component testing involves testing each component separately from the rest of the application. This can be achieved by using stubs and drivers to simulate the behaviour of other components that the tested component relies on.

* **Functionality and Usability:**

It verifies that each component performs its intended function correctly and meets the defined usability requirements.

* **Early Defect Detection:**

By testing components individually, it helps catch defects early in the development process, before they propagate to other parts of the system.

* **Reduced Integration Issues:**

Thorough component testing ensures that each component is working as expected, which reduces the likelihood of integration problems later on.

* **Cost and Time Savings:**

Identifying and fixing defects during component testing is generally more cost-effective and faster than fixing them after the components have been integrated into the larger system.

* **Building Block Approach:**

Component testing forms the foundation for subsequent integration and system testing, ensuring that each building block of the software is reliable.

Unit testing is also known as Module Testing or Progamme Testing.

Unit testing is commonly automated, but may still be performed manually.

1. **What is Functional System Testing ?**

A requirement that specifies a function that a system or system component must perform.

Requirement may exist as a text document and/or a model.

There are 2 types of it

1. **Requirement Based Functional Testing**
2. **Process Based Testing**

**Accuracy :** Provision of right or agreed results or effects

**Interoperability:** Ability to interact with specified systems

**Compliance :** Adhere to applicable standards, conventions, regulations or laws

**Auditability :** Ability to provide adequate and accurate audit data

**Suitability :** Presence and appropriateness of functions for specified tasks

1. **Requirement Based 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

Starts by using the most appropriate black-box testing techniques

May support this with white-box techniques (e.g. menu structures, web page navigation)

Risk based approach

1. **Business Process Based Testing**

Test procedures and cases derived from:

Expected user profiles

Business scenarios

Use cases

Testing should reflect the business environment and processes in which the system will operate.

Therefore, test cases should be based on real business processes.

Business Process Testing (BPT) is a software testing methodology that focuses on validating end-to-end business processes within an organization’s software applications.

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

Testing the attributes of a component or system that do not relate to functionality,

Ex: . reliability, efficiency, usability, interoperability, maintainability and portability

It is testing of “how” the testing works.

May be performed at all Test levels (not just Non Functional Systems Testing)

Measuring the characteristics of the system/software that can be quantified on a varying scale- e.g. performance test scaling

Non-functional testing includes, but is not limited to, performance testing, load testing, stress testing, usability testing, maintainability testing, reliability testing and portability testing.

To address this issue, performance testing is carried out to check & fine tune system response times.

The goal of performance testing is to reduce response time to an acceptable level.

Hence load testing is carried out to check systems performance at different loads i.e. number of users accessing the system.

(i)Web based testing

(ii)Desktop based testing

(iii)Game based testing

1. **What is GUI Testing?**

GUI testing, or Graphical User Interface testing, is a type of software testing that focuses on verifying the visual elements and user interactions of an application.

It ensures that the application's user interface functions correctly, looks as expected, and is easy to use.

This includes ensuring the GUI behaves in accordance with established requirements and operates as expected across a wide variety of supported devices and platforms.

Some popular, modern graphical user interface examples include Microsoft Windows, macOS, Ubuntu Unity, and GNOME Shell for desktop environments, and Android, Apple's iOS, BlackBerry OS, Windows 10 Mobile, Palm OS-WebOS, and Firefox OS for smartphones.

1. **What is Adhoc testing?**

Adhoc testing is an informal testing type with an aim to break the system.

It does not follow any test design techniques to create test cases.

In fact is does not create test cases altogether!

This testing is primarily performed if the knowledge of testers in the system under test is very high.

Testers randomly test the application without any test cases or any business requirement document.

Adhoc Testing does not follow any structured way of testing and it is randomly done on any part of application.

Main aim of this testing is to find defects by random checking.

Adhoc testing can be achieved with the testing technique called Error Guessing.

Error guessing can be done by the people having enough experience on the system to “guess” the most likely source of errors.

The Error guessing is a technique where the experienced and good testers are encouraged to think of situations in which the software may not be able to cope.

Some people seem to be naturally good at testing and others are good testers because they have a lot of experience either as a tester or working with a particular system and so are able to find out its weaknesses.

This is why an error guessing approach, used after more formal techniques have been applied to some extent, can be very effective.

It also saves a lot of time because of the assumptions and guessing made by the experienced testers to find out the defects which otherwise won’t be able to find.

Using experience to postulate errors.

Use Error Guessing to Complement Test Design Techniques.

1. **What is load testing?**

Load testing is a type of performance testing that evaluates how a system behaves under expected or peak user loads.

It simulates multiple users accessing a system simultaneously to identify performance bottlenecks, response delays, or stability issues.

The goal is to ensure the system can handle expected traffic and deliver a smooth user experience during peak usage.

Load testing simulates real-world scenarios by subjecting a system to a predetermined number of virtual users performing actions like browsing, adding items to a cart, or completing transactions.

 This helps identify potential issues before they affect actual users.

1. **What is stress testing?**

Stress testing software simulates extreme conditions & heavy loads on a system to evaluate its stability ,Reliability,& performances under pressure.

Tools like Apache J meter, load runner, Gating &web load.

It is type of performances testing that pushes a system beyond its normal operating capacity

to access its robustness & behaviour under extreme conditions.

It’s a crucial for identify potential failure points & ensuring a system can handle unexpected surges in traffic or usage.

1. **What is white box testing and list the types of white box testing?**

White box testing is a software testing method where the internal structure, design, and code of the software are known to the tester.

It focuses on verifying the internal workings of a system, ensuring that all code paths are tested and that the code functions as intended.

It's also known as clear box, glass box, or open box testing.

* 1. **Statement Coverage**
  2. **Branch Coverage**
  3. **Path Coverage**
  4. **Condition Coverage**
  5. **Data Flow Coverage**
  6. **Loop Testing**
  7. **Unit Testing**
  8. **Integration Testing**
  9. **Mutation Testing**
  10. **Static Analysis**
  11. **Penetration Testing**

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

Black box testing is a software testing method where the internal workings of the system are not known to the tester.

The focus is on the functionality of the software, testing it based on inputs and expected outputs, without knowledge of the code or internal structure

Different Black Box Testing Techniques:

1. **1. Equivalence Partitioning:**

Divides input data into partitions where all members are expected to be processed the same way. For example, if a field accepts ages between 1 and 100, you'd create partitions for valid values (1-100), and invalid values (less than 1, greater than 100).

1. **2. Boundary Value Analysis:**

Tests values at the boundaries of equivalence partitions, including the minimum and maximum values. For example, testing the input values of 0, 1, 99, and 100 when the valid range is 1 to 100.

1. **3. Decision Table Testing:**

Uses a table to map different input conditions to corresponding actions or outputs. It helps in testing all combinations of inputs and their expected results.

1. **4. State Transition Testing:**

Tests how a system behaves when transitioning between different states. This is useful for testing systems with different modes or states, such as a light switch or a stateful application.

1. **5. Error Guessing:**

Testers use their experience and knowledge to guess potential errors or vulnerabilities in the system. For example, testing for null values, incorrect data types, or security vulnerabilities.

1. **6. Use Case Testing:**

Tests the system based on user interactions or use cases. It focuses on how users will interact with the system and the expected results of those interactions.

1. **7. All-Pairs Testing:**

A combinatorial testing technique that tests all possible combinations of inputs. It's useful for finding interactions between different input parameters.

1. **8. Cause-Effect Graphing:**

Creates a graph to represent the relationships between inputs and outputs, then derives test cases from the graph.

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

Software defects can be categorized in several ways, but generally fall under functional, performance, usability, security, and compatibility defects.

These categories help testers prioritize and address issues during the software development lifecycle.

**1. Functional Defects**: These defects occur when the software fails to perform its intended functions as per the requirements.

* **Examples:** Login functionality not working, incorrect calculations, data not being saved properly.

**2. Performance Defects**: These relate to the speed, stability, and scalability of the software.

* **Examples:** Slow response times, high resource consumption, inability to handle large datasets.

**3. Usability Defects:** These defects make the software difficult or confusing to use.

* **Examples:** Poor navigation, unclear instructions, confusing interface.

**4. Security Defects:** These are vulnerabilities that could be exploited to compromise the system or data.

* **Examples:** Weak password encryption, SQL injection vulnerabilities, Cross-Site Scripting (XSS).

**5. Compatibility Defects**: These occur when the software doesn't work as expected across different environments (browsers, operating systems, devices).

* **Examples:** Feature not working on a specific browser, application not displaying correctly on a certain screen size.

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

In Big Bang integration testing all components or modules is 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 component are integrated together at **once**, and then tested.

. **Advantages:**

Convenient for small systems.

.**Disadvantages:**

Fault Localization is difficult.

Given the sheer number of interfaces that need to be tested in this approach, some

interfaces links to be tested could be missed easily.

Since the integration testing can commence only after “all” the modules are

designed, testing team will have less time for execution in the testing phase.

Since all modules are tested at once, high risk critical modules are not isolated &

tested on priority. Peripheral modules which deal with user interfaces are also not

isolated and tested on priority.

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

Exit criteria in a test plan define the conditions that must be met before a testing phase can be considered complete and the software is ready for the next stage. They ensure that sufficient testing has been conducted requirements are met, and the software meets quality standards. Essentially, they serve as a "green light" for moving forward in the software development lifecycle.

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

Regression testing should be performed after any code change, during major releases, after integrating components, in Continuous Integration environments, and periodically for legacy systems.

It ensures that new changes haven't introduced new bugs or broken existing functionality.

1. **7 keys principal of testing.**
2. **Testing shows presence of defects**
3. **Exhaustive testing is possible**
4. **Early testing**
5. **Defect clustering**
6. **Pesticide paradox**
7. **Testing is context dependent**
8. **Absence of error fallacy**
9. **Testing shows presence of defects**

Testing can show that defects are present, but cannot prove that there are no defects in software

1. **Exhaustive testing is possible**

We cant test every test cases ,if you try to test all the test cases again & again then you will waste your time & money.

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 life cycle, and should be focused on defined objectives.

Testing activities should start as **early** as possible in the development life cycle.

These activities should be focused on defined objectives – outlined in the Test Strategy.

1. **Defect clustering**

A small number of modules contain most of the defects discovered during pre-release testing, or are responsible for the most operational failures.

Defects are not evenly spread in a system They are clustered In other words, most defects found during testing are 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.

1. **Testing is context dependent**

Different software needs different type of testing .

Like e commerce web site is different from android application.

So it depend on context of software.

1. **Absence of error 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.

1. **Difference between QA V/S QC V/S Tester .**

Quality Assurance (QA) and Quality control (QC) are both important methods in software engineering to get high-quality software.

Quality Assurance (QA) prevents software defects or minimizes the number of defects in software before delivery by making sure that proper methods and processes are followed during the software development process.

Whereas Quality Control (QC) identifies and fixes the defects or errors that exist after development.

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

**Smoke testing** and **Sanity testing** are two important types of testing in software development.

Both the tests are used to validate the functionality of a software product and identify any critical issues, however they are quite different in their scope.

The basic difference between the two is that smoke testing is one that ensures that the features of an application are working fine or not, whereas sanity

|  |  |  |
| --- | --- | --- |
| **No.** | **Smoke Testing** | **Sanity Testing** |
| 1. | It is done to ensure that the functionalities of the program are working fine. | It is done to check if the bugs have been fixed after the build or not. |
| 2. | It is considered a subset of acceptance testing. | It is considered as a subset of regression testing. |
| 3. | It is documented. | It isn't documented. |
| 4. | It can be done by developers or testers. | It is generally performed by testers. |
| 5. | It may or may not be stable. | It is stable. |
| 6. | It is scripted. | It is not scripted. |
| 7. | It is done to understand the stability of the system or product. | It is done to understand the measure of rationality of the product. |
| 8. | It is done to test the functionality of the product or system. | It is used only in case of modified or defective functions of product. |
| 9. | It can be done manually or using automation. | It is generally done manually, not using automation. |
| 10. | It is done when a new product is built. | It is done after completing regression testing. |

testing is one that is performed to test whether all the errors have been fixed after the build of the application.

1. **Difference between verification and validation.**

Verification and validation are both crucial aspects of software testing, but they focus on different goals.

Verification is the process of ensuring that the software is built according to the specified requirements and design, essentially checking "are we building it right?".

Validation, on the other hand, focuses on confirming that the software meets the actual user needs and expectations, asking "are we building the right thing?".

**Verification:**

* **Focus:**

Building the product correctly, according to design specifications and requirements.

* **Methods:**

Includes reviews, inspections, walkthroughs, and static analysis of code and documentation.

* **Goal:**

To identify and correct errors early in the development lifecycle before they become more costly to fix.

* **Example:**

Checking if the user registration form has the correct fields (username, email, etc.) as per the requirements.

**Validation:**

* **Focus:**

Ensuring the software meets user needs and expectations, confirming it is fit for its intended purpose.

* **Methods:**

Involves dynamic testing, such as black box and white box testing, where the software is executed and its behavior is observed.

* **Goal:**

To identify if the software addresses the user's needs and works as expected in a real-world scenario.

* **Example:**

Testing the user registration form to ensure it correctly creates a user account and sends the confirmation email.

Key Differences Summarized:

|  |  |  |
| --- | --- | --- |
| **Feature** | **Verification** | **Validation** |
| Focus | Building the product right | Building the right product |
| Stage | Earlier in development | Later in development |
| Method | Static (no code execution) | Dynamic (code execution) |
| Example | Code review | User acceptance testing |

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

Performance testing in software testing encompasses various types, each designed to evaluate different aspects of a system's behaviour under specific conditions.

These types include load testing, stress testing, spike testing, endurance testing, scalability testing, and volume testing.

Each focuses on different performance characteristics like speed, stability, scalability, and reliability.

Types of Performance Testing:

* **Load Testing:**

Evaluates the system's behaviour under expected user loads, ensuring it performs well under normal and peak conditions.

* **Stress Testing:**

Determines the breaking point of the system by pushing it beyond its normal capacity, identifying its stability and recovery capabilities under extreme conditions.

* **Spike Testing:**

Assesses the system's response to sudden and significant increases in traffic, like a surge of users, ensuring it can handle the spike and recover smoothly.

* **Endurance Testing (Soak Testing):**

Verifies the system's long-term stability by running it under a sustained load for an extended period, identifying potential issues like memory leaks or performance degradation over time.

* **Scalability Testing:**

Examines the system's ability to handle increasing loads and user demands, ensuring it can scale up or down efficiently.

* **Volume Testing:**

Focuses on the system's performance when handling large volumes of data, often involving database testing with significant data amounts.

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

Error : A mistake in coding its called error. Developer uses this word most.

Defect: error found by tester is called defect.

Failure : Defect accepted by development team then its called bug, builds does not meet the requirements then it is called failure.

1. **Difference between Priority and Severity.**

Severity and priority, while related, represent different aspects of a defect.

Severity describes the technical impact of a defect on the functionality of a software product, while priority describes the urgency with which the defect should be fixed.

Essentially, severity is about the extent of the damage a bug causes, and priority is about how quickly that damage needs to be addressed.

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

The Bug Life Cycle is the standardized process a bug follows from identification to resolution, ensuring effective management and early detection to address issues promptly, preventing them from becoming deeply embedded in the code.

The bug life cycle in testing refers to a cycle of defects in which it goes through different states throughout its life.  The life cycle begins with a new defect discovered by a tester while testing the application.

It continues until the tester discovers a specific solution and closes the bug, so it does not reoccur.

The overall bug tracking life cycle involves multiple bug stages that enable the testers to track, debug, and improve the quality of the software.

This cycle ensures that all defects in the software are systematically addressed and resolved, leading to a more reliable and stable product.

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

|  |  |
| --- | --- |
| **Functional Testing** | **Non-Functional Testing** |
| Functional testing is performed using the functional specification provided by the client and verifies the system against the functional requirements. | Non-Functional testing checks the Performance, reliability, scalability and other non-functional aspects of the software system. |
| Functional testing is executed first | Non functional testing should be performed after functional testing |
| Manual testing or automation tools can be used for functional testing | Using tools will be effective for this testing |
| Business requirements are the inputs to functional testing | Performance parameters like speed , scalability are inputs to non-functional testing. |
| Functional testing describes what the product does | Nonfunctional testing describes how good the product works |
| Easy to do manual testing | Tough to do manual testing |
| 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 Nonfunctional testing are  • Performance Testing  • Load Testing  • Volume Testing  • Stress Testing  • Security Testing  • Installation Testing  • Penetration Testing  • Compatibility Testing  • Migration Testing |

**Functional Testing**

• This testing mainly involves black box testing and it is not concerned about the source code of the application.

• Each & every functionality of the system is tested by providing appropriate input, verifying the output and comparing the actual results with the expected results.

• This testing involves checking of User Interface, APIs, Database, security, client/ server applications and functionality of the Application under Test. The testing can be done either manually or using automation

**Non-Functional Testing**

• Non-Functional Testing: Testing the attributes of a component or system that do not relate to functionality, e.g. reliability, efficiency, usability, interoperability, maintainability and portability

• May be performed at all Test levels (not just Non Functional Systems Testing)

• Measuring the characteristics of the system/software that can be quantified on a varying scale- e.g. performance test scaling

• Non-functional testing includes, but is not limited to, performance testing, load testing, stress testing, usability testing, maintainability testing, reliability testing and portability testing.

• It is the testing of “how” the system works. Non-functional testing may be performed at all test levels.

• The term non-functional testing describes the tests required to measure characteristics of systems and software that can be quantified on a varying scale, such as response times for performance testing.

• To address this issue, performance testing is carried out to check & fine tune system response times. The goal of performance testing is to reduce response time to an acceptable level.

• Hence load testing is carried out to check systems performance at different loads i.e. number of users accessing the system.

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

The Software Development Life Cycle (SDLC) encompasses the entire process of creating software, from initial planning to deployment and maintenance.

The Software Testing Life Cycle (STLC), on the other hand, focuses specifically on the testing phase within the SDLC, ensuring the software is functional, reliable, and meets quality standards. Essentially, STLC is a subset of SDLC, dedicated to the testing aspects of software development.

**SDLC (Software Development Life Cycle):**

* **Scope:** Covers the complete software development process.
* **Phases:** Includes requirements gathering, design, coding, testing, deployment, and maintenance.
* **Focus:** Developing the software product from start to finish.
* **Goal:** To build a functional software that meets user needs and business goals.

**STLC (Software Testing Life Cycle):**

* **Scope:** Focuses on the testing phase within the SDLC.
* **Phases:** Includes test planning, test case design and development, test environment setup, test execution, and test closure.
* **Focus:** Ensuring software quality through various testing activities.
* **Goal:** To identify defects, verify functionality, and ensure the software meets specified quality standards.

SDLC provides the overall framework for software creation, while STLC provides the detailed roadmap for testing the software within that framework.

The STLC is a crucial part of the SDLC, ensuring that the software is thoroughly tested and meets the required quality criteria before release.

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

Test scenarios, test cases, and test scripts represent different levels of detail in the testing process.

Test scenarios are high-level descriptions of what should be tested, while test cases are detailed instructions on how to test specific aspects.

Test scripts are automated instructions written in a programming language for executing tests, often based on test cases.

**Test Scenarios:**

* **Purpose:** Outline broad areas of functionality to be tested, focusing on what needs to be validated.
* **Level:** High-level, providing an overview of critical features and workflows.
* **Example:** "Test user login functionality," "Verify product search functionality".
* **Creation:** Derived from requirements documents or user stories.

**Test Cases:**

* **Purpose:** Detail specific steps, input data, and expected results to validate individual features or functions.
* **Level:** Low-level, providing detailed instructions on how to execute a test.
* **Example:**
  + **Scenario:** User login.
  + **Case:** Enter valid username, enter valid password, click login button, verify successful login message.
* **Creation:** Derived from test scenarios and requirements.

**Test Scripts:**

* **Purpose:** Automate the execution of test cases, typically using programming languages.
* **Level:** Code-level, providing instructions for a testing tool or framework.
* **Example:** (Using Python and Selenium)  driver.find\_element(By.ID, "username").send\_keys("user123");
* **Creation:** Derived from test cases, often by mapping test steps to automated actions.

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

A test plan is a comprehensive document that outlines the approach, scope, objectives, resources, schedule, and activities required for testing a software application or system.

It serves as a roadmap for the testing team, providing a detailed guide on how testing will be conducted throughout the development lifecycle.

* 1. Introduction
  2. Test Strategy
  3. Test Items and Features
  4. Test Environment
  5. Schedule and Resources
  6. Risk Management
  7. Test Deliverables
  8. Communication Plan
  9. Approval

1. **What is priority?**

Priority refers to something that is considered more important than other things and should be dealt with first.

t's the state or quality of being earlier in time, occurrence, or the right to precede others in order or rank. Essentially, it means giving special attention or precedence to one task, person, or thing over others.

1. **What is severity?**

Severity refers to the degree or intensity of something, often used to describe the seriousness of an issue, problem, or event.

It indicates how bad, harmful, or significant something is. In various contexts, severity can be used to assess the impact of a defect in software testing, the seriousness of an illness, or the intensity of a weather event.

1. **Bug categories are…**

Software bugs can be broadly categorized into functional, logical, usability, security, performance, syntax, and system-level integration bugs.

These categories help testers and developers identify and address issues in software applications.

Security bugs

Usability bugs

Compatibility bugs

Functional bugs

Performance bugs

Logical bugs

Unit level bugs

Syntax errors

System level integration bugs

Out of bound bugs

Calculation errors

Forced bugs

Functional errors

Workflow bugs

Runtime errors

Boundary value errors

Bug types description

Calculation bugs

Compiler errors

Documentation bugs

Hardware defects

1. **Advantage of Bugzila .**

There are many advantages to using Bugzilla. It is an open source tool, which means no licensing costs.

Bugzilla offers advanced bug tracking features that are comparable to commercial solutions.

it is an open-source widely used bug tracker;

it is easy in usage and its user interface is understandable for people without technical knowledge;

it easily integrates withtest management instruments;

it integrates with an e-mailing system;

it automates documentation.

1. **Difference between priority and severity**

In software testing, severity refers to the degree to which a defect impacts the functionality of the software, while priority refers to how quickly that defect needs to be fixed.

Severity focuses on the technical impact, while priority focuses on the business impact and urgency.

**Severity:**

* **Definition:** Severity indicates the seriousness of the defect in terms of its impact on the software's functionality.
* **Focus:** Technical impact on the product.
* **Examples:**
  + **Critical:** The defect causes a system crash or major functionality failure, making the software unusable.
  + **High:** The defect significantly impairs functionality but doesn't cause a complete system failure.
  + **Medium:** The defect causes minor functionality issues or impacts a specific feature without major consequences.
  + **Low:** The defect has a minimal impact on functionality and can be easily ignored or worked around.
* **Assigned by:** Typically, the tester assigns the severity level.

**Priority:**

* **Definition:** Priority indicates the urgency with which a defect needs to be fixed.
* **Focus:** Business impact and urgency.
* **Examples:**
  + **High:** The defect must be fixed immediately as it blocks a critical feature or has a major impact on the business.
  + **Medium:** The defect should be fixed soon, but it may not be critical or have workarounds.
  + **Low:** The defect can be fixed in a later release or when time is available.

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

Agile development encompasses various methodologies, each with its unique approach to iterative development and collaboration.

Key agile methodologies include Scrum, Kanban, Extreme Programming (XP), Lean, and Crystal. These methodologies share core principles like frequent iterations, continuous learning, and delivering high-quality software, but they differ in their specific practices and focus.

* + 1. Scrum
    2. Kanban
    3. Extreme Programming (XP)
    4. Lean
    5. Crystal
    6. Feature-Driven Development (FDD)
    7. Dynamic Systems Development Method (DSDM)

1. **Explain the difference between Authorization and Authentication in Web testing. What are the common problems faced in Web testing?**

In web testing, authentication verifies a user's identity (e.g., logging in with a username and password).

while authorization determines what a user is allowed to access or do within the system (e.g., a user with admin rights can access the admin panel).

Common problems in web testing include cross-browser compatibility, responsiveness, security vulnerabilities, and performance issues.

Authentication is the process of verifying that a user is who they claim to be. It's like showing your ID to prove your identity.

 In web testing, this usually involves checking credentials like usernames and passwords or using other authentication methods like two-factor authentication.

Authorization, on the other hand, determines what resources a user is allowed to access or what actions they can perform within the system.

It's like having a boarding pass that allows you to access a specific gate and board a particular flight. In web testing, this involves verifying user roles and permissions to ensure they can only access what they are authorized to see and do.

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

User testing should be conducted throughout the entire product development lifecycle, from initial concept to post-launch optimization.

It's beneficial to test early, often, and continuously to gather user feedback and refine the product based on their needs.

The goal of usability testing is to understand how real users interact with your website and make changes based on the results

Usability testing is a strategy UX researchers use to spot errors and aid in the design process

UX researchers study the behaviour of consumers through research methods such as usability testing, data analysis, and findings reports.

Usability testing is a method used to learn how easy or difficult it is for people to use a service by observing them completing “real life” tasks within the service

The goal is to identify your users' pain points and create solutions that are user-focused to produce a service that is easy to use.

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

GUI (Graphical User Interface) testing ensures the application's interface functions correctly and provides a satisfactory user experience.

This involves verifying the visual elements like buttons, icons, menus, and overall layout, as well as the functionality of these elements.

The process can be manual, where testers interact with the interface, or automated using tools and scripts.

Here's a breakdown of the GUI testing procedure:

* + 1. Planning and Preparation
    2. Test Execution
       1. Manual Testing
       2. Automation testing
    3. Reporting and Defect Management
    4. Best Practices