**Assignment**

**Module 2 (Manual Testing)**

1. **What is software testing?**

* Testing is the process of evaluating a system or its component(s) with the intent to find that whether it satisfies the specified requirements or not.
* In simple words testing is executing a system in order to identify any gaps, errors or missing requirements in contrary to the actual desire or requirements.
* Software Testing is a process used to identify the correctness, completeness, and quality of developed computer software.

1. **What is Exploratory Testing?**

* Exploratory testing is a concurrent process where
* Test design, execution and logging happen simultaneously
* Testing is often not recorded
* 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

1. **What is traceability matrix?**

* To protect against changes you should be able to trace back from every system component to the original requirement that caused its presence.
* A software process should help you keeping the virtual table up-to-date.
* Simple technique may be quite valuable (naming convention)
* Traceability Matrix (also known as Requirement Traceability Matrix - RTM) is a table which is used to trace the requirements during the Software development life Cycle. It can be used for forward tracing (i.e. from Requirements to Design or Coding) or backward (i.e. from Coding to Requirements).
* 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.

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

* Boundary value analysis is a methodology for designing test cases that concentrates software testing effort on cases near the limits of valid ranges Boundary value analysis is a method which refines equivalence partitioning.
* Boundary value analysis generates test cases that highlight errors better than equivalence partitioning.
* The trick is to concentrate software testing efforts at the extreme ends of the equivalence classes.
* At those points when input values change from valid to invalid errors are most likely to occur.
* 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

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

* Aim is to treat groups of inputs as equivalent and to select one representative input to test them all
* EP can be used for all Levels of Testing
* Equivalence partitioning is the process of defining the optimum number of tests by:
* Reviewing documents such as the Functional Design Specification and Detailed Design Specification, and identifying each input condition within a function,
* Selecting input data that is representative of all other data that would likely invoke the same process for that particular condition.
* If we want to test the following IF statement: “If value is between 1 and 100 (inclusive) (e.g value >=1 and value <=100) Then...”

1. **What is Integration testing?**

* Integration 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.
* The purpose of this level of testing is to expose faults in the interaction between integrated units.
* Test drivers and test stubs are used to assist in Integration Testing.
* 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.
* There are 2 levels of Integration Testing
  + Component Integration Testing
  + System Integration Testing
* There is two types methods of Integration Testing:
* Bing Bang Integration Testing
* Incremental Integration Testing
* Top Down Approach
* Bottom Up Approach

1. **What determines the level 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 :
* 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

* Product Risk :
* 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

1. **What is Alpha testing?**

* It is always performed by the developers at the software development site.
* Sometimes it is also performed by Independent Testing Team.
* Alpha Testing is not open to the market and public
* It is conducted for the software application and project.
* It is always performed in Virtual Environment.
* 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 organization location with the involvement of developers.
* 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.
* 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

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.
* 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.

1. **What is component testing?**

* Component (Unit) – A minimal software item that can be tested in isolation. It means “A unit is the smallest testable part of software.”
* Component Testing – The testing of individual software components.
* Unit Testing is a level of the software testing process where individual units/components of a software/system are tested.
* The purpose is to validate that each unit of the software performs as designed.
* Unit testing is the first level of testing and is performed prior to Integration Testing.
* Sometimes known as Unit Testing, Module Testing or Program Testing.
* 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.
* 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.
* A unit 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.
* Unit tests find problems early in the development cycle.
* Unit testing is performed by using the White Box Testing method.

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

* Functional Testing based on an analysis of the specification of the functionality of a component or system.
* ‘Specification’ – E.g. Requirements specification, Use Cases, Functional specification or maybe undocumented.
* Functional testing verifies that each function of the software application operates in conformance with the requirement specification.
* 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
* Types of Functional testing are
  + Unit Testing
  + Smoke Testing
  + Sanity Testing
  + Integration Testing
  + White box testing
  + Black Box testing
  + User Acceptance testing
  + Regression Testing

1. **What is 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
* 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
* Types of Nonfunctional testing are
  + Performance Testing
  + Load Testing
  + Volume Testing
  + Stress Testing
  + Security Testing
  + Installation Testing
  + Compatibility Testing
  + Migration Testing

1. **What is GUI Testing?**

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 the controls like menus, buttons, icons, and all types of bars – tool bar, menu bar, dialog boxes and windows etc.

* Check all the GUI elements for size, position, width, length and acceptance of characters or numbers.
* For instance, you must be able to provide inputs to the input fields.
* Check you can execute the intended functionality of the application using the GUI
* Check Error Messages are displayed correctly
* Check for Clear demarcation of different sections on screen
* Check Font used in application is readable
* Check the alignment of the text is proper
* Check the 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

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.
* Main aim of this testing is to find defects by random checking.
* Adhoc testing can be achieved with the testing technique called Error Guessing.
* 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.

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

* White Box Testing: Testing based on an analysis of the internal structure of the component or system.
* Structure-based testing technique is also known as ‘white-box’ or ‘glass-box’ testing• technique because here the testers require knowledge of how the software is implemented, how it works
* White box testing is the detailed investigation of internal logic and structure of the code. White box testing is also called glass testing or open box testing.
* In order to perform white box testing on an application, the tester needs to possess knowledge of the internal working of the code.
* The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.
* Techniques of White box Testing :
  + Statement Coverage
  + Branch Coverage
  + Decision Coverage

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

* Black-box testing: Testing, either functional or non-functional, without reference to the internal structure of the component or system.
* Specification-based testing technique is also known as ‘black-box’ or input/output driven testing techniques because they view the software as a black-box with inputs and outputs.
* Specification-based techniques are appropriate at all levels of testing (component testing through to acceptance testing) where a specification exists.
* The technique of testing without having any knowledge of the interior workings of the application is Black Box testing.
* Typically, when performing a black box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.
* Techniques of Black Box Testing:
  + Equivalence partitioning
  + Boundary value analysis
  + Decision tables
  + State transition testing
  + Use-case Testing
  + Other Black Box Testing
    - Syntax or Pattern Testing

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

* Unreachable code
* Undeclared variables
* Parameter type mismatches
* Uncalled functions and procedures
* Possible array bound violations
* Security Violations
* Inconsistent interface between modules and components
* Incorrect variable usage
* Syntax checking
* Violations of code standards
* Use of variables without first defining them
* variables that are declared but never used
* Use of variables after they have been “killed”

1. **Mention what bigbang testing is?** 
   * + In Big Bang integration testing all components or modules is integrated simultaneously, after which everything is tested as a whole.
     + 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.

* + - **Advantage:**
* 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.

1. **What is the purpose of exit criteria?** 
   * + 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)
2. **When should "Regression Testing" be performed?**

* Regression Testing: Testing of a previously tested program following modification to ensure that defects have not been introduced or uncovered in unchanged areas of the software, as a result of the changes made. It is performed when the software or its environment is changed.
* 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

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

1. **Testing shows presence of Defects:**

* Testing can show that defects are present, but cannot prove that there are no defects.
* Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness.
* We test to find Faults As we find more defects, the probability of undiscovered defects remaining in a system reduces.

1. **Exhaustive Testing is Impossible! :**

* Testing everything including all combinations of inputs and preconditions is not possible.
* So, instead of doing the exhaustive testing we can use risks and priorities to focus testing efforts.
* For example: In an application in one screen there are 15 input fields, each having 5 possible values, then to test all the valid combinations you would need 30 517 578 125 (515) tests.

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
* Remember from our Definition of Testing, that Testing doesn’t start once the code has been written!

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. **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.

1. **Testing is Context Dependent :**

* Testing is basically 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!

1. **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’ Requirements.

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

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

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

|  |  |
| --- | --- |
| **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 |
| The objective of this testing is to verify the "stability" of the system | The objective of the testing is to verify the "rationality" of the system |
| This testing is performed by the developers or testers | 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 health check up |

1. **Difference between verification and Validation**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Verification** | **Validation** |
| Definition | The process of evaluating work-products (not the actual final product) of a development phase to determine whether they meet the specified requirements for that phase. | The process of evaluating software during or at the end of the development process to determine whether it satisfies specified business requirements. |
| Objective | to ensure that work products meet their specified requirements. | s, to demonstrate that the product fulfills its intended use when placed in its intended environment. |
| Evaluation Items | Plans, Requirement Specs, Design Specs, Code, Test Cases | The actual product/software. |
| Activities | Reviews  Walkthroughs  Inspections | Testing |

1. **Explain of Performance testing.**

* Software performance testing is a means of quality assurance (QA). It involves testing software applications to ensure they will perform well under their expected workload.
* Features and Functionality supported by a software system is not the only concern. A software application’s performance like its response time, do matter.
* The goal of performance testing is not to find bugs but to eliminate performance bottlenecks
* The focus of Performance testing is checking a software programs
* Speed – Determines whether the application responds quickly
* Scalability – Determines maximum user load the software application can handle.
* Stability – Determines if the application is stable under varying loads

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

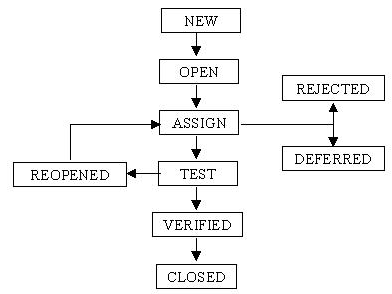
* “A mistake in coding is called error, error found by tester is called defect, defect accepted by development team then it is called bug, build does not meet the requirements then it is failure”

1. **Difference between Priority and Severity**

|  |  |
| --- | --- |
| **Servity** | **Priority** |
| Defined by the impact on the application’s functionality. | Defined by the impact on business. |
| Category decided by testers. | Category decided by developers or product owners. |
| Deals with the technical aspects of the application. | Deals with the timeframe or order to fix the defects. |
| The value does not change with time, it’s fixed. | Value of priority is subjective and may change after comparison with other defects. |

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

* “A computer bug is an error, flaw, mistake, failure, or fault in a computer program that prevents it from working correctly or produces an incorrect result. Bugs arise from mistakes and errors, made by people, in either a program’s source code or its design.”
* The duration or time span between the first time defects is found and the time that it is closed successfully, rejected, postponed or deferred is called as ‘Defect Life Cycle’.



* As you can see from above diagram, a defect‘s state can be divided into Open or Closed. When a bug reaches one of the Closed or Terminal states, its lifecycle ends. Each state has one or more valid states to move to.
* This is to ensure that all necessary steps are taken to resolve or investigate that defect.

**Defect Stages :**

* **New** : The Bug is newly found out and entered in the Bug tracking or Bug Reporting tool.
* **Open**: The Development or Test Lead reviews the defect. If it is determined to be a true defect, he or she adjusts the severity and priority and changes the status to open. A status of Open indicates that the defect is a true defect but that it has not yet been assigned to a developer for correction.
* **Assigned** : The bug is assigned to the Developer.
* **Tested** : The bug is tested by the Software tester.
* **Verified** : The bug is verified by the QA Lead.
* **Closed**: The tester verifies that the defect has been resolved and changes the status to Closed. A status of Closed indicates that the defect has been fixed and re-tested to the satisfaction of the person who first logged the defect
* **Rejected**: If the Test Lead finds that the system is working according to the specifications or the defect is invalid as per the explanation from the development team, he/she rejects the defect and marks its status as ‘Rejected’. A status of Rejected indicates that the defect is invalid, and therefore closed. No further work will be done on it.
* **Deferred**: In some cases the client may determine that a particular defect stands less important and can be deferred to a later stage. In that case it may be marked with ‘Deferred’, with a comment indicating when it should be reviewed again. A status of Deferred indicates that no further work will be done on the defect until that later date.
* **Reopened**: If after retesting the defect, the problem is not solved, the tester reopens changes the status to ‘Reopened’. A status of re-opened indicates that the developer failed to satisfactorily fix the defect and that it needs to be re-assigned to a developer for fixing.

**29) Explain the difference between Functional testing and NonFunctional testing**

|  |  |
| --- | --- |
| **Fuctional 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 |

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

|  |  |
| --- | --- |
| **SDLC** | **STLC** |
| SDLC is mainly related to softare development. | STLC is mainly related to software testing. |
| Beside development other phases like testing is also include. | It focues on only testing the software. |
| Goal of the SDLC is to complete successful development of software. | Goal of STLC is to complete successful testinf of software |
| It helps in developing good quality software. | It helps in making the software defects free. |
| SDLC phases are completed before STLC phases | STLC phases are performed after SDLC phases |

**31) What is the difference between test scenarios, test cases, and test script?**

**Test Scenarios:**

* A Test Scenario is any functionality that can be tested. It is also called Test Condition or Test Possibility.
* Test Scenario is ‘What to be tested’
* Test scenario is nothing but test procedure.
* The scenarios are derived from use cases.
* Test Scenario represents a series of actions that are associated together.
* Scenario is thread of operations

**Test Cases:**

* It is a document that contains the steps that has to be executed, it has been planned earlier
* Test cases involve the set of steps, conditions and inputs which can be used while performing the testing tasks.
* Test Case is ‘How to be tested’
* Test case consist of set of input values, execution precondition, expected Results and executed post-condition developed to cover certain test Condition.
* Test cases are derived (or written) from test scenario.

**Test Script:**

It is written in a programming language and it's a short program used to test part of functionality of the software system. In other words a written set of steps that should be performed manually.

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

* It is a high level document in which how to perform testing is described.
* The Test Plan document is usually prepared by the Test Lead or Test Manager and the focus of the document is to describe what to test, how to test, when to test and who will do what test.
* Master test plan: A test plan that typically addresses multiple test levels.
* Phase test plan: A test plan that typically addresses one test phase.
* A test plan will include the following.
* Introduction to the Test Plan document
* Assumptions when testing the application
* List of test cases included in Testing the application
* List of features to be tested
* What sort of Approach to use when testing the software
* List of Deliverables that need to be tested
* The resources allocated for testing the application
* Any Risks involved during the testing process
* A Schedule of tasks and milestones as testing is started