**Module–2 (Manual Testing)**

**What is software testing?**

* Software Testing is a process used to identify the correctness, completeness, and quality of developed computer software.
* Testing is a process rather than a activity, that’s take place throughout the SDLC
* Testing Activities like planning, preparation and evaluation of software product.
* Mainly two types of testing

Static Testing and Dynamic Testing

**What is Exploratory Testing?**

* **Exploratory Testing** is a one type of approach of software testing, where Test cases are not created in advance but testers check system on the fly.
* They may note down ideas about what to test before test execution. The focus of exploratory testing is more on testing as a “thinking” activity.
* Exploratory testing is a new way of thinking.
* it is highly teachable and manageable
* What actions you perform next is governed by what you are doing currently

**What is traceability matrix?**

* Traceability Matrix is a table which is used to trace the requirements during the Software development life Cycle.
* It is also known as Requirement Traceability Matrix - RTM
* It can be used for forward tracing and backward tracing
* Forward tracing is a tracing from requirement to design or coding
* Backward tracing is a tracing from coding to requirement.
* Further, Bi-Directional Traceability is a Good Traceability matrix is the References from test cases to basis documentation and vice versa.

**What is Boundary value testing?**

* Software testing technique in which tests are designed to include representatives of boundary values. It is performed by the QA testing teams.

OR

* It is one type of black box testing techniques, which is used for detecting any errors or threats that happened at boundary values of valid or invalid partitions rather than focusing on center of the input data.

**What is Equivalence partitioning testing?**

* Equivalence partitioning is a technique of software testing in which input data is divided into partitions of valid and invalid values, and it is mandatory that all partitions must exhibit the same behavior.
* If a condition of one partition is true, then the condition of another equal partition must also be true, and if a condition of one partition is false, then the condition of another equal partition must also be false.
* The principle of equivalence partitioning is, test cases should be designed to cover each partition at least once. Each value of every equal partition must exhibit the same behavior as other.

**What is Integration testing?**

* The phase in software testing in which individual software modules are combined and tested as a group. It is usually conducted by testing teams.
* Integration Testing *is* 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.

**What determines the level of risk?**

* **The likelihood of an adverse event and the impact of the event**.

**What is Alpha testing?**

* 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.
* Unit testing, integration testing and system testing when combined are known as alpha testing.
* Alpha Testing is not open to the market and public.

**What is beta 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 always performed at the user’s premises in the absence of the development team.
* Beta testing can be considered “**pre-release**” testing.
* Beta Testing is always performed at the time when software product and project are marketed.
* Beta Testing is always open to the market and public.

**What is component testing?**

* 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.

**What is functional system testing?**

* Functional System Testing : A requirement that specifies a function that a system or system component must perform
* A Requirement may exist as a text document and/or a model
* Functional System Testing Functionality As below:
* Accuracy, Interoperability, Compliance, Auditability, Suitability
* There is two types of techniques

Requirement Based Functional Testing and Process Based functional Testing

**What is 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.

**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.

**What is Adhoc testing?**

* 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 is an informal testing type with an aim to break the system.
* It does not follow any test design techniques to create test cases.

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

* Testing technique based on knowledge of the internal logic of an application’s code and includes tests like coverage of code statements, branches, paths, conditions. It is performed by software developers.
* Testing of software with complete knowledge of its internal code and logic.
* **Testing based on an analysis of the internal structure of the component or system.**
* 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.
* **7 Different types of white-box testing**

1. Unit Testing
2. Static Analysis
3. Dynamic Analysis
4. Statement Coverage
5. Branch testing Coverage
6. Security Testing
7. Mutation Testing

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

* The technique of testing without having **any knowledge of the interior workings of the application** is Black Box testing.
* **Testing, either functional or non-functional, without reference to the internal structure of the component or system.**
* The testers **have no knowledge of how the system or component is structured inside the box**. In black-box testing the tester is concentrating on what the software does, not how it does it.
* **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

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

* **Data Quality/Database Defects:** Deals with improper handling of data in the database.
* Examples:

Values not deleted/inserted into the database properly

Improper/wrong/null values inserted in place of the actual values

* **Critical Functionality Defects:** The occurrence of these bugs hampers the crucial functionality of the application.
* Examples: - Exceptions
* **Functionality Defects:** These defects affect the functionality of the application.
* Examples:

All JavaScript errors

Buttons like Save, Delete, Cancel not performing their intended functions

A missing functionality (or) a feature not functioning the way it is intended to

Continuous execution of loops

* **Security Defects:** Application security defects generally involve improper handling of data sent from the user to the application. These defects are the most severe and given highest priority for a fix.
* Examples:

Authentication: Accepting an invalid username/password

Authorization: Accessibility to pages though permission not given

* **User Interface Defects:** As the name suggests, the bugs deal with problems related to UI are usually considered less severe.
* Examples:

Improper error/warning/UI messages

Spelling mistakes

Alignment problems

**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.

**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)

* Exit Criteria ensure that the testing of the application is completed and ready for release.
* Here are the exit criteria:-

All the planned requirements must be met

All the high Priority bugs should be closed

All the test cases should be executed

If the scheduled time out is arrived

Test manager must sign off the release

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

* Regression testing means testing your software application when it undergoes a code change to ensure that the new code has not affected other parts of the software.
* Change in requirements and code is modified according to the requirement New feature is added to the software
* Defect fixing
* Performance issue fix

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

* Testing shows presence of Defects
* Exhaustive Testing is Impossible!
* Early Testing
* Defect Clustering
* The Pesticide Paradox
* Testing is Context Dependent
* Absence of Errors Fallacy
* **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.
* **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 testingefforts.
  + 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**.
  + This is very unlikely that the project timescales would allow for this number of tests.
  + So, accessing and managing risk is one of the most important activities and reason for testing in any project.
  + We have learned that we cannot test everything (i.e. all combinations of inputs and preconditions).
  + That is we must **Prioritise** our testing effort using a **Risk Based Approach**.
* **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!
* **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
* **Pesticide Paradox**
  + If the same tests are repeated over and over again, eventually the same set of **test cases will no longer find any new defects**.
  + To overcome this “pesticide paradox”, the test cases need to be **regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects.**
* **Testing 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.**
* **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’

**Difference between QA v/s QC v/s Testing**

|  |  |  |  |
| --- | --- | --- | --- |
| **SN** | **Quality Assurance** | **Quality Control** | **Testing** |
| 1 | Activities which ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements. | Activities which ensure the verification of developed software with respect to documented (or not in some cases) requirements. | Activities which ensure the identification of bugs/error/defects in the Software. |
| 2 | Focuses on processes and procedures rather than conducting actual testing on the system. | Focuses on actual testing by executing Software with intend to identify bug/defect through implementation of procedures and process. | Focuses on actual testing. |
| 3 | Process oriented activities. | Product oriented activities. | Product oriented activities. |
| 4 | Preventive activities. | It is a corrective process. | It is a preventive process |
| 5 | It is a subset of Software Test Life Cycle (STLC). | QC can be considered as the subset of Quality Assurance. | Testing is the subset of Quality Control. |

**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 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 |
| 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 |

**Difference between verification and Validation**

|  |  |
| --- | --- |
| **Verification Phase** | **Validation Phase** |
| The process of evaluating work-products (not the actual final product) of a development phase to determine whether they meet the specified requirements for that phase. | The process of evaluating software during or at the end of the development process to determine whether it satisfies specified business requirements. |
| To ensure that the product is being built according to the requirements and design specifications. In other words, to ensure that work products meet their specified requirements. | To ensure that the product actually meets the user’s needs, and that the specifications were correct in the first place. In other words, to demonstrate that the product fulfills its intended use when placed in its intended environment. |
| Are we building the product right? | Are we building the right product? |
| Plans, Requirement Specs, Design Specs, Code, Test Cases | The actual product/software. |
| Reviews Walkthroughs Inspections | Testing |

**Explain types 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.
* 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. **Load testing**
2. **Stress testing**
3. **Endurance testing**
4. **Spike testing**
5. **Volume testing**
6. **Scalability testi**

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

* Error is the - Difference between Expected Output and Actual Output. In other word Functionality is working but not correctly. Ex. Taxi Charge
* Defect - Actual output is different from customers expectations Ex. Cycle Wheel square instead of round.
* Failure - If software does not work as per its capacity is called failure. Ex. AC fan on but cooling not done
* Bug - Bug is a result of Programming error because of which a functionality do not give expected output.

**Difference between Priority and Severity**

|  |  |
| --- | --- |
| **Severity** | **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. |
| Defect Priority has defined the order in which the developer should resolve a defect | Defect Severity is defined as the degree of impact that a defect has on the operation of the product |
| Priority is categorized into three types | Severity is categorized into five types |
| Low | Critical |
| Medium | Major |
| High | Moderate |
|  | Minor |
|  | Cosmetic |
| Priority is associated with scheduling | Severity is associated with functionality or standards |
| Priority indicates how soon the bug should be fixed | Severity indicates the seriousness of the defect on the product functionality |
| Priority of defects is decided in consultation with the manager/client | QA engineer determines the severity level of the defect |
| Priority is driven by business value | Severity is driven by functionality |
| Its value is subjective and can change over a period of time depending on the change in the project situation | Its value is objective and less likely to change |
| High priority and low severity status indicates, defect have to be fixed on immediate bases but does not affect the application | High severity and low priority status indicates defect have to be fixed but not on immediate bases |
| Priority status is based on customer requirements | Severity status is based on the technical aspect of the product |
| During UAT the development team fix defects based on priority | During SIT, the development team will fix defects based on the severity and then priority |

**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’
* The different phases of Bug life cycle are,
  + New or Opened
  + Assigned
  + Fixed
  + Tested
  + Closed

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

|  |  |
| --- | --- |
| **Functional testing** | **Non-functional testing** |
| It is performed before non-functional testing. | It is performed after the functional testing. |
| It is based on customer’s requirements. | It focusses on customer’s expectation. |
| It is easy to define functional requirements. | It is difficult to define the requirements for non-functional testing. |
| Helps to validate the behavior of the application. | Helps to validate the performance of the application. |
| Carried out to validate software actions. | It is done to validate the performance of the software. |
| Functional testing is carried out using the functional specification. | This kind of testing is carried out by performance specifications |
| Functional testing is easy to execute by manual testing. | It’s very hard to perform non-functional testing manually. |
| It describes what the product does. | It describes how the product works. |
| Check login functionality. | The dashboard should load in 2 seconds. |
| Examples of Functional Testing Types | Examples of Non-functional Testing Types |
|  |  |
| Unit testing | Performance Testing |
| Smoke testing | Volume Testing |
| User Acceptance | Scalability |
| Integration Testing | Usability Testing |
| Regression testing | Load Testing |
| Localization | Stress Testing |
| Globalization | Compliance Testing |
| Interoperability | Portability Testing |
|  | Disaster Recover Testing |

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

* Test Plan is A document describing the scope, approach, resources, and schedule of intended test activities.
* A test plan is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product.
* Test Plan helps us determine the effort needed to validate the quality of the application under test.
* The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager.

1. Analyze the product
2. Design the Test Strategy
3. Define the Test Objectives
4. Define Test Criteria
5. Resource Planning
6. Plan Test Environment
7. Schedule & Estimation
8. Determine Test Deliverables

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

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

**Explain the difference between Authorization and Authentication in Web testing.**

|  |  |
| --- | --- |
| **Authentication** | **Authorization** |
| Authentication is the process of identifying a user to provide access to a system. | Authorization is the process of giving permission to access the resources. |
| In this, the user or client and server are verified. | In this, it is verified that if the user is allowed through the defined policies and rules. |
| It is usually performed before the authorization. | It is usually done once the user is successfully authenticated. |
| It requires the login details of the user, such as user name & password, etc. | It requires the user's privilege or security level. |
| Data is provided through the Token Ids. | Data is provided through the access tokens. |
| Authentication credentials can be partially changed by the user as per the requirement. | Authorization permissions cannot be changed by the user. The permissions are given to a user by the owner/manager of the system, and he can only change it. |
| **Example:** Entering Login details is necessary for the employees to authenticate themselves to access the organizational emails or software. | **Example:** After employees successfully authenticate themselves, they can access and work on certain functions only as per their roles and profiles. |

**What are the common problems faced in Web testing?**

### Insufficient testing for browser compatibility

### Failing to conduct thorough functional testing across mobile

### Failing to conduct thorough functional testing across desktop

### Poor data security

### Failing to provide an intuitive experience

### Not performing testing frequently enough

### Leaving digital accessibility to the last minute

### Releasing new features breaks the existing live system

### Localisation and the global experience

### The most common bugs