**TESTING**

* Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not
* testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

Difference between Testing and Debugging:

**Testing:**

* It involves the identification of bug/error/defect in the software without correcting it.
* Done in testing phase

**Debugging:**

* It involves identifying, isolating and fixing the problems/bug.
* Debugging is the part of white box or unit testing
* Done in developing phase

Difference between Verification & Validation:

**Verification**:

* Are you building it right?
* Ensure that the software system meets all the functionality.
* Verification takes place first and includes the checking for documentation, code etc
* Done by developers
* Have static activities as it includes the reviews, walkthroughs, and inspections to verify that software is correct or not.
* It is an objective process and no subjective decision should be needed to verify the Software.

**Validation:**

* Are you building the right thing?
* Ensure that functionalities meet the intended behavior.
* Validation occurs after verification and mainly involves the checking of the overall product.
* Done by Testers.
* Have dynamic activities as it includes executing the software against the requirements.
* It is a subjective process and involves subjective decisions on how well the Software works.

Difference between QA & QE:

**QA-Quality Analyst:**

* One who ensures/maintains the quality of a product by executing on Code Science's quality procedures.
* Focus on testing for defects
* Goal-Defect identification

**QE-Quality engineer:**

* One who automates quality procedures to minimize manual testing efforts
* Focus on building in quality
* Goal-Defect prevention

Different between Bug and a defect:

**Bug:**

* Bug is a mistake in the program -> unit / dev-Integration phases

**Defect:**

* this is a deviation in the requirement. The program is not behaving as per the requirement.

**Defect Life cycle:**

* A defect lifecycle, or bug lifecycle, is a specific set of states that a software bug goes through from discovery to fixation.
* The lifecycle may vary from organization to organization depending on factors like company policy, software developmental model (e.g., Agile, Waterfall, etc.), and project timeline.

Testing Documentation:

**Test plan:**

* The test plan is a base of software testing.
* It is a detailed document, which includes several testing attributes such as test objectives, scope, test schedule, template, required resources (human resources, software, and hardware), test estimation and test deliverables, risk, mitigation plan, defect tracking, entry and exit criteria, test environment, etc.
* which defines software testing areas and activities.
* It is a formal document used to define the scope of testing and different testing activities.
* A test plan is developed by Test Lead or test/ Project manager.

**Test Strategy:**

* The test strategy is a high-level document used to validate the test levels to be executed for the product.
* Describes what kind of technique has to be used and which module will be tested.
* It contains various components like documentation formats, objectives, test processes, scope, customer communication strategy, etc.
* The Test Strategy's main purpose is to deliver a systematic approach to the software testing process to ensure reliability, quality, traceability, and better planning.
* It is a high-level document that involves planning for all the testing activities and delivering a quality product.
* Generally, the test strategy is developed by the Business Analyst and approved by the Project Manager.

**Test Scenario:**

* The test scenario is a detailed document of test cases that cover end to end functionality of a software application in liner statements.
* The liner statement is considered as a scenario.
* The test scenario is a high-level classification of testable requirements.
* These requirements are grouped on the basis of the functionality of a module and obtained from the use cases.
* there is a detailed testing process due to many associated test cases.
* Before performing the test scenario, the tester has to consider the test cases for each scenario.
* testers need to put themselves in the place of the user because they test the software application under the user's point of view.
* Preparation of scenarios is the most critical part, and it is necessary to seek advice or help from customers, stakeholders or developers to prepare the scenario.

**Test Case:**

* The test case is defined as a group of conditions under which a tester determines whether a software application is working as per the customer's requirements or not.
* Test case designing includes preconditions, case name, input conditions, and expected result.
* A test case is a first level action and derived from test scenarios.
* Test cases involve the set of steps, conditions and inputs which can be used while performing the testing tasks.
* The main intent of this activity is to ensure whether the Software Passes or Fails in terms of its functionality and other aspects.
* There are many types of test cases like: functional, negative, error, logical test cases, physical test cases, UI test cases etc.

**Test Script:**

• Same as a test case but created programmatically.

**User stories:**

* A user story is an informal, general explanation of a software feature written from the perspective of the end user .

It includes,

* FRD: Functional Requirement Document
* BRD: Business Requirement Document
* HLD: High level Design Document
* LLD: Low Level Design Document

**Requirement Traceability Matrix – RTM:**

* It 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).

• There are many user defined templates for RTM.

* Each requirement in the RTM document is linked with its associated test case, so that testing can be done as per the mentioned requirements.
* Furthermore, Bug ID is also include and linked with its associated requirements and test case.
* The main goals for this matrix are: Make sure Software is developed as per the mentioned requirements, Helps in finding the root cause of any bug, Helps in tracing the developed documents during different phases of SDLC.

**STLC-Software Testing Life Cycle:**

* The Software Testing Life Cycle (STLC) is a sequence of specific actions performed during the testing process to ensure that the software quality objectives are met.

• The STLC includes both verification and validation.

* It consists of a series of methodological activities to help certify your software product.
* It includes

Requirement is confirmed->Test planning->RTM->Test strategy->Test plan->Test scenario->Test case->Test execution->go/ no go(defect fixing by developers)

**Test Management:**

* Test Management is a process of managing the testing activities in order to ensure high quality and high-end testing of the software application.
* The method consists of organizing, controlling, ensuring traceability and visibility of the testing process in order to deliver the high quality software application.
* It ensures that the software testing process runs as expected.

**PROJECT DEVELOPMENT METHODOLOGIES:**

* **Water fall model:**
* It is a linear life cycle model, Water fall model followed in the sequential order and so the project development team only moves to next phase of development or testing if the previous step is completed successfully.
* Linear sequential life cycle model
* Methodology is structured
* Sequential design process
* Testing comes after the build phase
* Changing the requirements once the project development starts is not possible
* **Agile methodology:**
* It is a continuous iteration of development & testing. Agile means in the form of sprints, each sprint spans up to around 2 weeks.
* Continuous iteration of development & Testing
* Methodology is flexible
* Incremental approach
* Testing occur concurrently with development
* Changing is possible at any time

Testing Types:

**Manual Testing:**

* Include testing of the Software manually
* Tester identifies any un-expected behaviour or bug without using any automated tool.
* There are different stages for manual testing like unit testing, Integration testing, System testing and User Acceptance testing.
* Testers use test plan, test cases or test scenarios to test the Software to ensure the completeness of testing.

**Automation Testing:**

* Tester writes scripts and uses another software to test the software.
* Used to re-run the test scenarios that were performed manually, quickly and repeatedly.

Testing Methods:

**Black Box Testing:**

* Technique of testing without having any knowledge of the interior workings of the application.
* 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.
* Testing is done without the knowledge of the internal structure of program or application
* Main goal- test the behaviour of software.
* Focused on external or end user perspective.
* Less time consumption

**White Box Testing:**

* Detailed investigation of internal logic and structure of the code.
* Testing is done with the knowledge of internal structure of program
* Main goal- test the internal operation of software.
* Focused on code, structure, condition paths & branches.
* More time consumption

Levels of Testing:

**Functional Testing**

This is a type of black box testing that is based on the specifications of the software that is to be tested.

* Unit Testing:
* This type of testing is performed by the developers before the setup is handed over to the testing team to formally execute the test cases.
* Unit testing is performed by the respective developers on the *individual units* of source code assigned areas.
* The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.
* Limitations of Unit Testing- Testing cannot catch each and every bug in an application, it is impossible to evaluate every execution path in every software application, There is a limit to the number of scenarios and test data that the developer can use to verify the source code.
* **Integration Testing:**
* The testing of *combined parts* of an application to determine if they function correctly together is Integration testing.
* There are two methods of doing Integration Testing Bottom-up Integration testing and Top Down Integration testing.
* **System Testing:**
* Once all the components are integrated, the application as a whole is tested rigorously to see that it meets Quality Standards.
* This type of testing is performed by a specialized testing team.
* **Regression Testing:**
* Whenever a change in a software application is made it is quite possible that other areas within the application have been affected by this change.
* To verify that a fixed bug hasn’t resulted in another functionality or business rule violation is Regression testing.
* The intent of Regression testing is to ensure that a change, such as a bug fix did not result in another fault being uncovered in the application.
* **Acceptance Testing:**
* Check whether the application meets the intended specifications and satisfies the client’s requirements.
* The QA team will have a set of pre written scenarios and Test Cases that will be used to test the application.
* 2 stages-Alpha & Beeta testing.
* Alpha Testing: Test each user journey and confirm they work as intended- It always done by developers or tester at the software development site. Done before the software release where it will address & fix bugs and correct any minor issue
* Beta Testing: Test how the software perform in the real world scenario-It always performed by the customers at their own site. User provide feedback & result to the product team for in co-operate in the next version.

**Non Functional Testing**

Non-functional testing of Software involves testing the Software from the requirements which are non-functional in nature related but important a well such as performance, security, user interface etc.

* Performance Testing:
* It is mostly used to identify any performance issues rather than finding the bugs in software.
* Performance testing is considered as one of the important and mandatory testing type in terms of some aspects such as Speed (i.e. Response Time, data rendering and accessing), Capacity, Stability and Scalability.
* 2 types- Load testing and Stress testing.
* Load testing- A process of testing the behaviour of the Software by applying maximum load in terms of Software accessing and manipulating large input data. This type of testing identifies the maximum capacity of Software and its behaviour at peak time.
* Stress testing- This testing type includes the testing of Software behaviour under abnormal conditions. Taking away the resources, applying load beyond the actual load limit. The main intent of this testing is to identify the breaking point of software.
* **Static Testing**
* Static Testing is a type of a Software Testing method which is performed to check the defects in software without actually executing the code of the software application.
* Static testing includes activities like code Review, Walkthrough, etc.
* A test strategy is an outline that describes the testing approach of software development cycle.
* It gives a set of guidelines that explains test design and determines how testing needs to be done.

**Requirement Testing:**

* Requirement will be based on userstories
* All test case, Test senarios, Test data are included in requirements
* Requirement Testing is base on the requirement provided by the client

**Types of Requirement Testing:**

* Explicit requirements-It is a first type of requirements, found mostly in documents. It includes the things you wrote down
* Implicit requirements- It is the second type of requirements. It includes the things that users are going to expect that were not captured explicitly. Eg: performance, security, usability, availability etc...
* Latent requirement- It represents the behaviour that users do not expect based on their previous experiences but which will make them like the software more. Eg: when I transfer money from one account to another they shows the transaction is successful, which I didn’t expect but I am delight.