**TESTING**

* Step in SDLC
* Testing is the process of checking the functionality of an application to ensure it runs as per requirements.
* Unit testing comes into picture at the developers’ level; it is the testing of a single entity (class or method). Unit testing plays a critical role in helping a software company deliver quality products to its customers.

Unit testing can be done in two ways:

| *Manual Testing* | *Automated Testing* |
| --- | --- |
| Executing test cases manually without any tool support is known as manual testing. | Taking tool support and executing the test cases by using an automation tool is known as automation testing. |
| Time-consuming and tedious: Since test cases are executed by human resources, it is very slow and tedious. | Fast: Automation runs test cases significantly faster than human resources. |
| Fast: Automation runs test cases significantly faster than human resources. | Less investment in human resources: Test cases are executed using automation tools, so less number of testers are required in automation testing. |
| Less reliable: Manual testing is less reliable, as it has to account for human errors. | More reliable: Automation tests are precise and reliable. |
| Non-programmable:No programming can be done to write sophisticated tests to fetch hidden information. | Programmable: Testers can program sophisticated tests to bring out hidden information. |

**JUnit**

JUnit is a unit testing framework for Java programming language. It plays a crucial role test-driven development, and is a family of unit testing frameworks collectively known as xUnit.

**Featuresof JUnit**

* JUnit is an open source framework, which is used for writing and running tests.
* Provides annotations to identify test methods.
* Provides assertions for testing expected results.
* Provides test runners for running tests.
* JUnit tests allow you to write codes faster, which increases quality.
* JUnit is elegantly simple.
* It is less complex and takes less time.
* JUnit tests can be run automatically and they check their own results and provide immediate feedback.
* There's no need to manually comb through a report of test results.
* JUnit tests can be organized into test suites containing test cases and even other test suites.
* JUnit shows test progress in a bar that is green if the test is running smoothly, and it turns red when a test fails.

**Requirement Traceability Matrix(RTM)**

* RTM is a document in the form of a table which maps and traces user requirements with test cases.
* It captures all requirements proposed by the client and requirement traceability in a single document, delivered at the conclusion of the Software development life cycle.
* The main purpose of Requirement Traceability Matrix is to validate that all requirements are checked via test cases such that no functionality is unchecked during Software testing.
* Created by the test engineer/ test lead

**Test Plan**

* A Test Plan can be defined as a document that defines the scope, objective, and approach to test the software application.
* The Test Plan is as I like to call a ‘super document’ that lists everything there is to know and need.
* It's a detailed doc which describes the strategy

**Test Strategy**

* Test Strategy is a set of guidelines that explain the test design and determine how testing needs to be done.
* For example: A Test Strategy includes details like “Individual modules are to be tested by the test team members”. In this case, who tests it does not matter – so it’s generic and the change in the team member does not have to be updated, keeping it static.

| TEST PLAN | TEST STRATEGY |
| --- | --- |
| It is derived from software requirement specification(SRS). | It is derived from the Business Requirement document(BRS). |
| It is prepared by the test lead or manager. | It is developed by the project manager or the Business analyst. |
| Test plan id, features to be tested, test techniques, testing tasks, features pass or fail criteria, test deliverables, responsibilities, and schedule, etc. are the components of the test plan. | Objectives and scope, documentation formats, test processes, team reporting structure, client communication strategy, etc. are the components of test strategy. |
| We can prepare a Test plan at the project level. | We can use Test strategy at multiple projects. |
| It describes how to test , when to test, who will test and what to test. | It describes what type of technique to follow and which module to test. |

**Test Cases**

* A test case is a document, which has a set of test data, preconditions, expected results and postconditions, developed for a particular test scenario in order to verify compliance against a specific requirement.
* Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution postcondition.

**Test Script**

* Test Scripts are a line-by-line description containing the information about the system transactions that should be performed to validate the application or system under test.
* Test script should list out each step that should be taken with the expected results.

| TEST CASES | TEST SCRIPT |
| --- | --- |
| It is a step by step by procedure that is used to test an application | It is a set of instructions to test an application automatically. |
| The term Test Case is used in the manual testing environment. | The term Test Script is used in the automation testing environment(using scripting format). |
| It is developed in the form of templates. | It is developed in the form of scripting. |
| Test case template includes Test Suit ID, Test Data, Test procedure, Actual results, Expected results etc. | In Test Script we can use different commands to develop scripts. |
| Example: We need to verify the login button in an application,  The steps include:  a) Launch the application.  b) Verify if the login button is displaying or not. | Example: We want to click an image button in an application.  The script includes:  a) Click the Image Button. |

**Test Scenario**

* It is a way to define all the possible ways to test an application. It is a single statement to cover all possible ways to test an application.
* Example: Validate if a new country can be added by the Admin

**User Story**

* Requirement of the client
* User story break them into test cases
* A user story is a tool in [Agile software development](https://searchsoftwarequality.techtarget.com/definition/agile-software-development) used to capture a description of a software feature from a user's perspective. The user story describes the type of user, what they want and why. A user story helps to create a simplified description of a requirement.

**STEPS IN SDLC**

1. ***Planning and requirement analysis***

* Requirement Analysis is the most important and necessary stage in SDLC.
* The senior members of the team perform it with inputs from all the stakeholders and domain experts or SMEs in the industry.
* Planning for the quality assurance requirements and identifications of the risks associated with the projects is also done at this stage.

1. ***Defining Requirements***

* Once the requirement analysis is done, the next stage is to certainly represent and document the software requirements and get them accepted from the project stakeholders.
* This is accomplished through "SRS"- Software Requirement Specification document which contains all the product requirements to be constructed and developed during the project life cycle.

1. ***Designing the Software***

* The next phase is about to bring down all the knowledge of requirements, analysis, and design of the software project.
* This phase is the product of the last two, like inputs from the customer and requirement gathering.

1. ***Developing the project***

* In this phase of SDLC, the actual development begins, and the programming is built. The implementation of design begins concerning writing code.
* Developers have to follow the coding guidelines described by their management and programming tools like compilers, interpreters, debuggers, etc. are used to develop and implement the code.

1. ***Testing***

* After the code is generated, it is tested against the requirements to make sure that the products are solving the needs addressed and gathered during the requirements stage.
* During this stage, unit testing, integration testing, system testing, acceptance testing are done.

1. ***Deployment***

* Once the software is certified, and no bugs or errors are stated, then it is deployed.
* Then based on the assessment, the software may be released as it is or with suggested enhancement in the object segment.
* After the software is deployed, then its maintenance begins.

1. ***Maintenance***

* Once when the client starts using the developed systems, then the real issues come up and requirements to be solved from time to time.
* This procedure where the care is taken for the developed product is known as maintenance.

**SDLC MODELS**

***Waterfall model***

* Waterfall Model methodology which is also known as Liner Sequential Life Cycle Model.
* The Waterfall Model followed in the sequential order, and so project development team only moved to the next phase of development or testing if the previous step completed successfully.

***Agile model***

* Agile methodology is a practice which promotes continued interaction of development and testing during the SDLC process of any project.
* In the Agile method, the entire project is divided into small incremental builds. All of these builds are provided in iterations, and each iteration lasts from one to three weeks.

***RAD model***

* RAD or Rapid Application Development process is an adoption of the waterfall model.
* It targets developing software in a short period.
* The RAD model is based on the concept that a better system can be developed in less time by using focus groups to gather system requirements.

***Spiral Model***

* The spiral model is a risk-driven process model.
* This SDLC model helps the group to adopt elements of one or more process models like a waterfall, incremental, waterfall, etc.
* The spiral technique is a combination of rapid prototyping and concurrency in design and development activities.

***V-Model***

* In this type of SDLC model testing and the development, the step is planned in parallel.
* So, there are verification phases on the side and the validation phase on the other side. V-Model joins by Coding phase.

***Incremental Model***

* The incremental model is not a separate model. It is necessarily a series of waterfall cycles.
* The requirements are divided into groups at the start of the project. For each group, the SDLC model is followed to develop software.

***Iterative Model***

* It is a particular implementation of a software development life cycle that focuses on an initial, simplified implementation, which then progressively gains more complexity and a broader feature set until the final system is complete.

***Prototypical Model***

* The prototyping model starts with the requirements gathering. The developer and the user meet and define the purpose of the software, identify the needs, etc.
* A 'quick design' is then created. This design focuses on those aspects of the software that will be visible to the user. It then leads to the development of a prototype. The customer then checks the prototype, and any modifications or changes that are needed are made to the prototype.

**Bug and Defect**

* Bugs are issues found during the testing lifecycle. Issues in code can cause bugs.
* Defects are issues found in the production environment, and may be a deviation from the requirement. They can also be found by a developer while unit-testing.

**Risk**

* In software testing Risks are the possible problems that might endanger the objectives of the project stakeholders. It is the possibility of a negative or undesirable outcome. A risk is something that has not happened yet and it may never happen; it is a potential problem.

**Entry and Exit criteria**

* Entry Criteria: Entry Criteria gives the prerequisite items that must be completed before testing can begin.
* Exit Criteria: Exit Criteria defines the items that must be completed before testing can be concluded

**TYPES OF TESTING**

### ***Accessibility Testing***

### Accessibility testing is the practice of ensuring your mobile and web apps are working and usable for users without and with disabilities such as vision impairment, hearing disabilities, and other physical or cognitive conditions.

### ***Acceptance Testing***

### Acceptance testing ensures that the end-user (customers) can achieve the goals set in the business requirements, which determines whether the software is acceptable for delivery or not. It is also known as user acceptance testing (UAT).

### ***Unit Testing***

Unit testing is the process of checking small pieces of code to ensure that the individual parts of a program work properly on their own, speeding up testing strategies and reducing wasted tests.

### ***White Box Testing***

White box testing involves testing the product's underlying structure, architecture, and code to validate input-output flow and enhance design, usability, and security.

### ***Black Box Testing***

### Black box testing involves testing against a system where the code and paths are invisible.

### ***End to End Testing***

End to end testing is a technique that tests the application’s workflow from beginning to end to make sure everything functions as expected.

### ***Functional Testing***

Functional testing checks an application, website, or system to ensure it’s doing exactly what it’s supposed to be doing.

### ***Interactive Testing***

Also known as manual testing, interactive testing enables testers to create and facilitate manual tests for those who do not use automation and collect results from external tests.

### ***Integration Testing***

Integration testing ensures that an entire, integrated system meets a set of requirements. It is performed in an integrated hardware and software environment to ensure that the entire system functions properly.

### ***Load Testing***

This type of non-functional software testing process determines how the software application behaves while being accessed by multiple users simultaneously.

### ***Non Functional Testing***

Non functional testing verifies the readiness of a system according to nonfunctional parameters (performance, accessibility, UX, etc.) which are never addressed by functional testing.

### ***Performance Testing***

Performance testing examines the speed, stability, reliability, scalability, and resource usage of a software application under a specified workload.

### ***Regression Testing***

Regression testing is performed to determine if code modifications break an application or consume resources.

### ***Sanity Testing***

Performed after bug fixes, sanity testing determines that the bugs are fixed and that no further issues are introduced to these changes.

### ***Security Testing***

Security testing unveils the vulnerabilities of the system to ensure that the software system and application are free from any threats or risks. These tests aim to find any potential flaws and weaknesses in the software system that could lead to a loss of data, revenue, or reputation per employees or outsides of a company.

### ***Single User Performance Testing***

Single user performance testing checks that the application under test performs fine according to specified threshold without any system load. This benchmark can be then used to define a realistic threshold when the system is under load.

### ***Smoke Testing***

This type of software testing validates the stability of a software application, it is performed on the initial software build to ensure that the critical functions of the program are working.

### ***Stress Testing***

Stress testing is a software testing activity that tests beyond normal operational capacity to test the results.

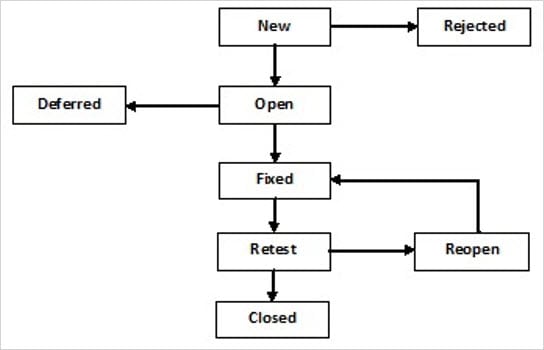
**Test Management**

* Test management most commonly refers to the activity of managing a testing process.
* A test management tool is [software](https://en.wikipedia.org/wiki/Software) used to manage [tests](https://en.wikipedia.org/wiki/Test_case) (automated or manual) that have been previously specified by a test procedure. It is often associated with [automation](https://en.wikipedia.org/wiki/Automation) software.

**Risk Analysis**

* Risk is the probability of occurrence of an undesirable event.
* Risk Analysis in Software Engineering is the process of analyzing the risks associated with your [Testing](https://www.guru99.com/software-testing.html) Project.

**Defects life cycle**



**Defects States**

1. ***New:***

This is the first state of a defect in the Defect Life Cycle. When any new defect is found, it falls in a ‘New’ state, and validations & testing are performed on this defect in the later stages of the Defect Life Cycle.

1. ***Assigned:***

In this stage, a newly created defect is assigned to the development team to work on the defect. This is assigned by the project lead or the manager of the testing team to a developer.

1. ***Open:***

Here, the developer starts the process of analyzing the defect and works on fixing it, if required.If the developer feels that the defect is not appropriate then it may get transferred to any of the below four states namely Duplicate, Deferred, Rejected, or Not a Bug-based upon a specific reason. We will discuss these four states in a while.

1. ***Fixed:***

When the developer finishes the task of fixing a defect by making the required changes then he can mark the status of the defect as “Fixed”.

1. ***Pending Retest:***

After fixing the defect, the developer assigns the defect to the tester to retest the defect at their end, and until the tester works on retesting the defect, the state of the defect remains in “Pending Retest”.

1. ***Retest:***

At this point, the tester starts the task of retesting the defect to verify if the defect is fixed accurately by the developer as per the requirements or not.

1. ***Reopen:***

If any issue persists in the defect, then it will be assigned to the developer again for testing and the status of the defect gets changed to ‘Reopen’.

1. ***Verified:***

If the tester does not find any issue in the defect after being assigned to the developer for retesting and he feels that if the defect has been fixed accurately then the status of the defect gets assigned to ‘Verified’.

1. ***Closed:***

When the defect does not exist any longer, then the tester changes the status of the defect to “Closed”.

**QA/QE**

* ***QA: Quality Analyst –***
  + One who ensures/maintains the quality of a product by executing on CodeScience’s quality procedures.
  + One who see the end software is working good with the requirements of the final product.
* ***QE: Quality Engineer –*** 
  + one who automates quality procedures to minimize manual testing efforts.
  + One who import the right artifacts to perform routine quality checks and create pretentious measures that helps to mitigate defects.
  + He will minimise the manual testing effects

**Test Pyramid**

* Test Pyramid is a framework that can help both developers and QAs create high-quality software.
* It reduces the time required for developers to identify if a change they introduced breaks the code.
* It can also be helpful in building a more reliable test suite. It is also referred to as the Test automation pyramid.

*This test automation pyramid operates at three levels:*

* ***Unit tests:-***

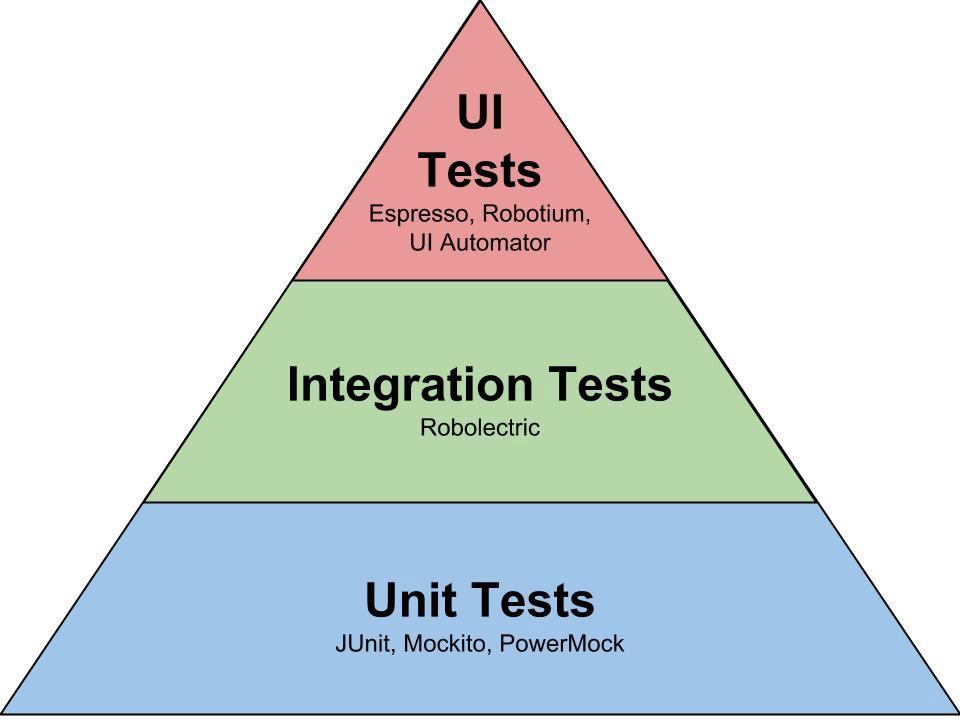
Unit tests form the base of the testing pyramid. They test individual components or functionalities to validate that it works as expected in isolated conditions. It is important to run a number of scenarios in unit tests – happy path, error handling, etc.

* ***Integration tests:-***

Unit tests verify small pieces of a codebase. However, in order to test how this code interacts with other code (that form the entire software), integration tests need to be run. Essentially, these are tests that validate the interaction of a piece of code with external components. These components can range from databases, external services (APIs) and the like.

* ***UI /End-to-End Test:-***

At the top of the pyramid are the UI tests. These ensure that the entire application is functioning as required. End-to-end tests do exactly what the name suggests: test that the application is working flawlessly from start to finish.



**Verification and Validation**

* *Verification* in Software Testing is a process of checking documents, design, code, and program in order to check if the software has been built according to the requirements or not.
* The main goal of the verification process is to ensure quality of software application, design, architecture etc.
* The verification process involves activities like reviews, walk-throughs and inspection.
* *Validation* in Software Engineering is a dynamic mechanism of testing and validating if the software product actually meets the exact needs of the customer or not.
* The process helps to ensure that the software fulfills the desired use in an appropriate environment.
* The validation process involves activities like unit testing, integration testing, system testing and user acceptance testing.

**Requirements Testing**

* Requirements testing is done to clarify whether project requirements are feasible or not in terms of time, resources and budget.
* Many bugs emerge in software because of incompleteness, inaccuracy and ambiguities in functional requirements.
* That’s why it is highly important to test requirements and eliminate ambiguities before you start to develop a project.

*Types of Requirements Testing*

* Implicit Requirement Testing
* Explicit Requirement Testing
* Latent Requirement Testing

**Static Testing**

* Static Testing is done without executing the application
* It is done to avoid errors at an early stage of development as it is easier to identify the errors and solve the errors.
* It also helps finding errors that may not be found by Dynamic Testing.

**Difference between Alpha testing and Beta testing**

| ALPHA TESTING | BETA TESTING |
| --- | --- |
| It is a type of software testing performed to identify bugs before releasing the product | Performed by user in real environment |
| Involves both black box and white box testing | Commonly uses black box testing |
| Alpha testing is performed by testers who are usually internal employees of the organization. | Beta testing is performed by clients who are not part of the organization. |
| Alpha testing is performed at the developer's site. | Beta testing is performed at the end-user of the product. |
| Reliability and security testing are not checked in alpha testing. | Reliability, security and robustness are checked during beta testing. |
| Alpha testing ensures the quality of the product before forwarding to beta testing. | Beta testing also concentrates on the quality of the product but collects users input on the product and ensures that the product is ready for real time users. |
| Alpha testing may require a long execution cycle. | Beta testing requires only a few weeks of execution. |