**SDLC -** software life cycle development - is a process of software development with high quality and low cost in shortest time period



Development models,

**-Waterfall**  - phase must be completed before the next phase can begin and there is no overlapping in the phases.

**-Iterative** - iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed

**-V-model** - execution of processes happens in a sequential manner in a V-shape. It is also known as Verification and Validation model

**-Agile** -combination of iterative and incremental process models.Agile Methods break the product into small incremental builds. These builds are provided in iterations.

**STLC :** Software Testing Life Cycle (STLC) is a sequence of specific activities conducted during the testing process to ensure software quality goals are met



**Methods of Testing**

**Black Box Testing** - is done without knowing the internal codes and structure of the program. The testing is done from the customer’s point of view and the tester knows only about the inputs and the expected outputs of the application.

**White Box Testing -** It is the testing method in which internal codes & structure of the software is known to the tester. Tester should have better understanding of the source code.

**Gray Box Testing -** Tester have partially knowledge on the internal codes and structure of the program.

**Testing types/Methodologies**

**Functional Testing** - Validating each functions of the software application against the requirement specification. Its a type of Black box testing.

**Types:**

**Unit Testing** - individual units or components of a software are tested.Mostly done by Developer team

**Integration testing -** individual units / components are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.

**System testing** - testing the complete and fully integrated software product. Each and every functionalities will be tested.

**User Acceptance Testing (UAT) -** is a type of testing performed by the end user or the client to verify/accept the software application before moving to the production environment.

**Smoke Testing -** is a type of testing where critical functionalities will be tested after the software build, to ensure whether the latest built is stable or not. It is executed “before” any detailed functional or regression tests are executed on the software build.

**Sanity testing** - performed after receiving a software build, with minor changes in code, or functionality, to ensure that the bugs have been fixed and no further issues are introduced due to these changes.

**Regression testing** - is testing the existing software application to make sure that a new change made on the application hasn't broken any existing functionality. Also called as Impact testing.

**End-to-End Testing -** is a methodology used to test whether the flow of an application is performing as designed from start to finish.

**Non-functional testing** - is a type of testing to check non-functional aspects (performance, Security, reliability, etc.) of a software application.

**Performance Testing** - evaluating the performance of the software application under particular load. By using this stability of the software is evaluated.

**Load testing** - testing the application by constantly and steadily increasing the load on the application until it reaches the threshold limit.

**Stress testing** - testing the application using load beyond the threshold limit, in order to find the behaviour of the application in extreme condition. Its a type of negative testing.

**Security Testing** - is a type of Software Testing that uncovers vulnerabilities of the application and determines that the data and resources of the application are protected from possible intruders.

Other misc types

**Adhoc testing:** is unplanned testing. It doesn’t have any specific approach defined neither it has any documentation associated with it. Adhoc testing is entirely informal, and the only important factor is the knowledge and insight of the tester. Only suitable for Manual testing.

**Exploratory testing:** performed when the documentation of the test is poor, and we have a short time for execution. It requires creativity of the tester and also the tester’s product knowledge.Only suitable for Manual testing.

**Levels of Testing:**

-Unit Testing

-Integration Testing

-System Testing

-Acceptance Testing

**Static Testing or Verification** - is a form of software testing where the actual program or application is not used. It is done without execution of the code. Static testing is done to avoid errors at an early stage of development. Eg- Code review, Requirement and design specification review, Walkthroughs. It is also called Verification and its a type of white box testing.

**Dynamic testing or Validation -** is done on the actual application to ensure that the specified requirement or not. It will be carried out after the code deployment with the execution the code. It will be done at the later stage of development. It involves functional and non-functional testing and comes under validation.

**Error** - is a human action that produces an incorrect result. The mistakes made by programmer is known as an ‘Error’.

**Bug -** is the result of a coding Error in the program

**Defect -** is a deviation from the Requirements. A defect does not necessarily mean there is a bug in the code, it could be a function that was not implemented but defined in the requirements of the software.

**Failure** is a deviation of the software from its intended purpose. It is reported by client or end user.

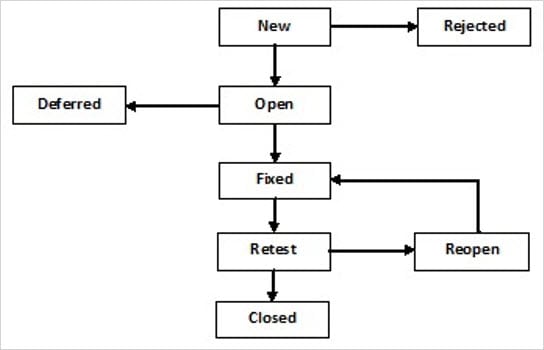
**Severity** is a parameter to denote the impact of a particular defect on the software. It is determined by **Test Engineer**.

**Types -** Critical, Major, Minor, Low (spelling mistakes on the web page).

**Priority** is a parameter to decide the order in which defects should be fixed. It is determined by **Product owner or Client**.

**Types** - High, Medium, Low.

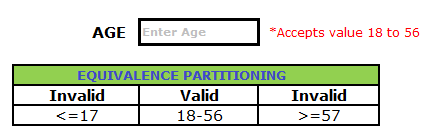
**Defect Life cycle:**



**Defect rejection ratio:** (No. of defects rejected/ total no. of defects raised) X 100

**Defect leakage ratio:** (No. of defect missed/total defects of software) X 100

**Equivalence Partitioning** is also known as Equivalence Class Partitioning. In equivalence partitioning, inputs to the software or system are divided into groups that are expected to exhibit similar behavior



**Boundary value analysis (BVA)** is based on testing the boundary values of valid and invalid partitions. The Behavior at the edge of each equivalence partition is more likely to be incorrect than the behavior within the partition, so boundaries are an area where testing is likely to yield defects.