**What is software? Types of Software?**

Software is a collection of computer programs that help us to perform a task.

OR

Software refers to a set of instructions, programs, or data that tell a computer or a system how to perform specific tasks or functions.

It's essentially the digital component that enables computers, smartphones, tablets, and other electronic devices to carry out a wide range of operations, from basic tasks like arithmetic calculations to complex processes like running applications, managing databases, and controlling hardware components.

Software is used to control a computer. There are different types of software that can run on a computer: system software, utility software, and application software.

**1. System software**

* System software is essential for the operation of computer hardware and provides a platform for running application software.
* It manages computer resources and facilitates communication between hardware components and application programs.
* If you think of software as being in layers, the system software is the bottom layer: it sits between the hardware and the application software.
* Operating systems like Windows, macOS, Android and iOS are examples of system software.

## 2. Utility software

* Utility software designed to perform specific tasks to enhance the performance, efficiency, and functionality of computer systems.
* Utility software is part of the system software and performs specific tasks to keep the computer running.
* Utility software typically focuses on system maintenance, optimization, security, and management.
* Utility software is always running in the background. Examples of utility software are security and optimisation programs.
* Security programs include anti-virus software that scans and removes viruses. Most computers will include some sort of anti-virus software, but you can add your own.
* These tools are typically installed as part of the operating system.
* They have access to the hard drive to keep it tidy.

## 3. Application software

* Application software is designed to perform specific tasks or functions for end-users.
* This is everything else! Anything that is not an operating system or a utility is an application or app.
* So, a word processor, spreadsheet, web browser, and graphics software are all examples of application software, and they can do many specific tasks.
* You can remove and add applications on your computer using the operating system.
* Application software like a word processor regularly directs the operating system to load and save files from and to the hard drive.
* Testing mainly focus on application software

**4.Programming Software**

* Programming software, also known as development software or coding tools, refers to a category of software applications designed to assist programmers and developers in writing, testing, debugging, and maintaining software code.
* These tools provide an integrated development environment (IDE) or a set of functionalities that streamline the software development process and facilitate efficient coding practices.
* Programming software typically includes features such as code editing, syntax highlighting, debugging, version control, and project management capabilities.

**What is software testing?**

Software testing is a part of software development process. Software testing is an activity to detect and identify the defect present in the software. The aim of software testing is to delivered a quality product to the client within the given time and cost. The primary goal of software testing is to identify defects, errors, or bugs in the software and to verify that it behaves as expected. Software testing check the software weather it meets with the user requirement or not.

**What is software quality?**

Software quality refers to the degree to which a software product meets specified requirements, fulfils user expectations, and satisfies stakeholder needs. Software quality is a multidimensional concept that encompasses both functional and non-functional aspects of the software.

Parameters of Software Quality.

* Bug Free
* Delivered on time
* Within budget
* Meet requirements and expectations
* Maintainable (After installing the software in the customers environment they should able to work with that and if there is any small bug or issues then they should be able to resolved themselves and the software should be user friendly)

**Project Vs Product**

**Project:** If software is developed for specific customer based on the requirement then it is called project. A project is a temporary endeavour undertaken to create a unique product, service, or result.

**Product**: If software is developed for multiple customers based on market requirement then it is called Product. A product hand, is a tangible or intangible item that is created or produced as a result of a project or ongoing development efforts.

**Characteristics of Project**

* Temporary
* Unique
* Defined Scope
* Resources
* Project Management

**Characteristics of Product**

* Ongoing
* Marketed
* Customer Value
* Maintenance
* Product Management

**Why do we need testing?**

Because we have to release the quality product. This is main reason of doing testing. Testing helps catch mistakes early so you can fix them and deliver a better product that people will enjoy using. It should not have any bug it should work according to customers requirement

**Error, Bug & Failure**

**Error:** Error is nothing but the human mistakes. If we do any mistakes that is called as error. We can say it as an incorrect human action. (developer)

**Bug:** Deviation of expecting and actual result. Something is not working according to our requirement. (tester)

**Failure:** Deviation identified by end user by giving an action. Bug that found on customer side after delivering the product. (end user or client)

**Why the software has bugs?**

Software has bugs because making software is really complicated! In software, there are millions of tiny "pieces" (lines of code) that all need to work together perfectly. With so many parts, it's easy to miss something or make a mistake. Bugs happen when these mistakes cause the software to behave differently than expected.

**Reasons for occurring bug:**

* Miscommunication or no communication (between developers and testers, developer is developing something and tester is testing something else)
* Software complexity
* Programming errors (by developers)
* Changing requirements
* Lack of skilled testers

**SDLC**

SDLC is Software development life cycle. It is a process use by software industry to design, develop and test the software. SDLC is a step by step process which we have to follow to develop , test and delivered the software to the customer.

**Requirement Analysis:** Requirement analysis is like figuring out exactly what needs to be done before starting a project. It's about understanding what the software needs to do, making a list of everything it should have, and creating a plan to make it happen.

**Design:** The design phase in the SDLC is where we decide what the software will look like and how it will work. It's like creating a detailed map or plan before building the software.

**Development:** The development phase in the SDLC is where the actual building of the software happens. In simple terms, the development phase is where the software is built according to the plans laid out in the design phase.

**Testing:** The testing phase in the Software Development Life Cycle (SDLC) is where we check if the software works correctly. We test different parts of the software to find any problems or "bugs," and then we fix them to make sure the software runs smoothly and check its working according to the requirement.

**Maintenance:** The maintenance phase is like taking care of the software to ensure it continues to run smoothly and meets the needs of users over time.

**STLC**

**Waterfall Model**

Waterfall Model is the very old and traditional model use to develop the software. When the software industry started the first and initial model, they have started with waterfall model. It is also called as linear model.



In this model each and every activity will be documented it means every phase has been documented and this documentation will play a very important role. In waterfall model each phase needs some input and each phase produce one output and the output of previous phase becomes the input to the next phase. Every phase in depend on the previous phase of the model. We can not change the requirement in the middle of any phase.

**Requirement Analysis**: In requirement analysis we will talk to the customer, customer in the sense normally business people like product manager, project manager, these people will be involved in these communications. They will collect the requirement from the customer and once they get all the requirement, they prepare some kind of document that is called SRS document. In the requirement analysis phase, we gathered the requirement from the customer and we will prepare a document (here document part is very important).

**System design:** Once the document is created System Design is start. Designers are present in this phase and they create a design based on the requirement. They will divide the entire document in high level modules and low-level modules. In design phase based on the requirement document the designers will prepare the design document.

**Implementation:** The developer will develop the software based on the design document. In this phase base on the design document coding and implementation is start.

**Testing:** After completion of software the tester will test the software and compare the functionality of software with the requirement mentioned in the SRS document.

**Deployment:** Once the testing is completed, we will deploy the software in customers environment so deployment is nothing but we will install the software in the customers environment. They will start using the

**Maintenance:** After installing the software They will start using the software this comes under maintenance phase.

**Advantages of waterfall model:**

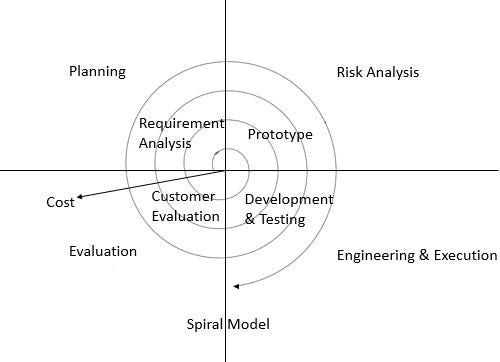
* Quality of the product will be good
* Since requirement change are not allowed , chances of finding bug will be less
* Initial investment is less since the tester are hied in the later stages
* Preferred for smaller project where requirement are feezed

**Disadvantages of waterfall model:**

* Requirement changes are not allowed
* If there is a defect in requirement that will be continued in the later phases
* Total investment is more, because the time taking for rework on defect is time consuming which leads to high investment.
* Testing will start only after coding

**Spiral Model**

Spiral model is also called as iterative model. To overcome the drawback of waterfall model spiral model is invented.



**Planning:** Here planning in the sense requirement analysis. Planning phase includes the estimating costs, schedule for iteration.

**Risk Analysis:** How much risk is involved, how many developers required, how many testers required. In simple terms its deals with analysing the risk.

**Engineering& Execution:** In this phase, the execution part is being done by engineers and developers.

**Evaluation:** In evaluation phase, the product is being assessed by the client and provided with the revert if any changes required from client side.

After evaluation phase, one version of the software is delivered to the customer, the customer will give few more requirement. Then again, we do Planning, Requirement analysis, Engineering & Execution, Evaluation and again we will be delivered version 2 of the software. If again some more requirement is added then we follow this phase again and delivered the version 3 of the software. We can add new requirement only after completion of the cycle, we cannot add new requirement in the middle of the cycle. This cycle will moved continuously, endlessly until we have completed requirement for the customer. Each and every time we will follow the all phases again and again and delivered the new version to the customer. The main feature of spiral model is to accommodate the new requirement and also accommodate the frequently changing requirement. This model is specially suited for product which are updated according to the market. Software will be release in multiple versions. So, it is also called as version control model.

**Advantages of Spiral Model:**

Testing is done in every cycle, before going to next cycle

Customer will get to use the software for every module

Requirement changes are allowed after every cycle before going to next cycle

**Disadvantages of Spiral Model:**

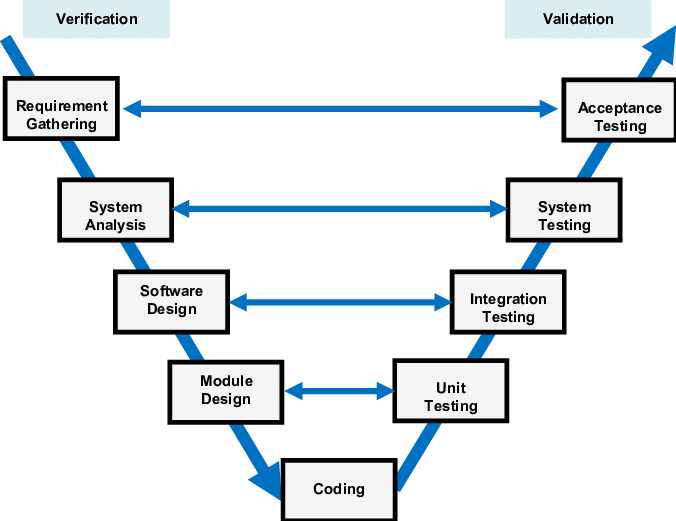
Requirements changes are not allowed in between the cycle.

Every cycle of spiral modal looks like waterfall model.

There is no testing in requirement & design phase.

**V-Model**

V-Model is verification and validation model.The main Speciality of V model is in every phase it will do testing.



**First Phase / Requirement analysis:** First phase is to collect requirement from the user. So, here we have something called BRS, CRS and URS. This is tested by Using Reviews (Static Testing).

**BRS –** Business Requirement Specification

**CRS –** Customer Requirement Specification

**URS –** User Requirement Specification

(Requirement given by customer or client) (Prepare by business unit team) (Only understand by the business people not by the technical people) (Very huge document)

**Second Phase / System Analysis:** In Second phase SRS document is created with the help of BRS, CRS and URS document. These three documents are not understood by the developer so based on this document SRS document is created. SRS document content more technical details. SRS is understood by both developer and tester.SRS is prepare by product manager. This is tested by Using Reviews (Static Testing).

**Third Phase / Software Design:** In this phase based on the SRS documents the designers will prepare HLD (High Level Design) and LLD (Low Level Design) documents. This document is prepared by the designers. HLD contains the main modules LLD contains the low-level modules This is tested by Using Reviews (Static Testing).

**Fourth phase / Module Design:** In this phase coding is started based on this document.

**Unit Testing:** We cannot develop the complete software we develop the software in different module testing of this module is called as unit testing. Developer conduct the unit testing. This is done at code level so it is called as white box testing.

**Integration Testing:** After integrating all the module which are develop in Unit testing, we do integration testing. This is also done by developers. This is done at code level so it is called as white box testing.

**System Testing:** After integrating all the modules, we do System Testing. System Testing is done by testers. They don’t need to understand the code they will just verify or test our software is working according to the customer requirements (Check the functionality). Testers are involved. It comes under Block box testing.

**User Acceptance Testing:** User and customer will do some kind of testing at their level. Here, also we don’t need to understand the code so its comes under black-box testing.

**Verification & Validation**

**Verification:** Verification check whether we are building the right product. (Check the each and every step) (Focus on documentation.)(are we building the product correctly or not)(Conduct – Before software is ready)

Verification typically involves Static testing.

* Review
* Walkthrough
* Inspection

**Validation:** Validation check whether we are building the product right. (Check whatever the software we have developed is correct or not) (Conduct after the software is ready)

Takes place after verifications are completed.

Validation typically involves Dynamic testing.

* Unit testing
* Integration Testing
* System Testing
* UAT

**Advantages:**

Testing is involved in each and every phase

**Disadvantages:**

Documentation is more.

Initial investment is more.

**Review, Walkthrough, Inspection:**

**Review:** Conduct on document to ensure correctness and completeness of the software.

Review can be done anytime by anyone there is no restriction nothing.

Only one single person can do review.

Types of review we can do

* Requirement Review
* Design Review
* Code Review
* Test Plan Review
* Test cases Review, etc

**Walkthrough:** It is also a type of review but it is an informal review (There is not be any plan).

It can be do anytime at any place. It can be done by 2 or more people or peers (two or more team members). It doesn’t have any pre-planned it can be done whenever required. Also, it doesn’t have minutes of the meet.

**Inspection:** Inspection is a most formal review type, most formal means we have too pre-planned. We have to send invite to the team members not only tester sometimes even developers and managers also. In which at least 3-8 people will seat in a meeting. There are 3 types of people will be involved in the meeting 1-reader (author of the document), 2-writer (who note down all the clarification and the issues), 3-moderator (organizer of the meeting) plus concerned. Inspection will have the proper schedule which will be intimated email to the consent developer or testers.

**QA. QC and QE**

P – People - QC

P – Process - QA

P – Product

**QA (Quality Analyst):** High level management people will come under the QA. QA is process related. Takes care about entire process development cycle from the beginning (requirement) to till we release the product to the customer.QA makes sure rest of the people are following the process properly or not .QA is involved in every stage of SDLC because they are taking care of the process.

**QC (Quality Control):** They are involved only during testing part The people who are working under testing they are actually comes under QC. QC is people related. They are verifying and validating the software whether it is actually working according to the customer requirement or not.

|  |  |
| --- | --- |
| QA is process related. | QC is actual testing of the software.. |
| QA focuses on building in Quality | QC focuses on testing for Quality |
| QA is preventing defect | QC is detecting defect |
| QA is process oriented | QC is product Oriented |
| QA for entire life cycle | QC for testing part in SDLC |

**QE (Quality Engineering):** QE is a person who write the code for testing the software. QE comes under QC.

**Different Level of Software testing**

The four levels of software testing are

* Unit testing
* Integration Testing
* System Testing
* UAT

**Unit Testing:** As soon as the coding starting, developers start doing unit testing.

* Unit testing is the single component or module of a software.
* Unit testing conduct on a single program or a single module.
* Unit testing is a white box testing technique.
* Unit testing is conducted by unit testers.
* Unit testing technique
  + Basis path testing (Every line of code should be executed at least once)
  + Control structure testing
* Conditional coverage
* Loop coverage
  + Mutation Testing(repetition)

**Integration Testing:** Integrating two or more modules we do integration testing.

* It is performed between two or more modules.
* It focuses on checking data communication between multiple modules.
* It is a white box testing technique.
* It is done by developers but sometimes it is also done by testers.
* Types of Integration testing techniques

1. **Incremental integration testing**

* Incrementally adding the modules and testing the data flow between the modules.
* We keep on adding new modules with the existing modules then test the data flow and communication between the modules which is called Incremental integration testing.
* There are two approaches in Incremental Integration Testing.
* **Top-down approach:** Incrementally adding the modules and testing the data flow between the modules and ensure the modules are added is the child of previous module.
* **Bottom-up approach:** Incrementally adding the modules and testing the data flow between the modules and ensure the module added is the parent of the previous module.
* **Sandwich/Hybrid approach:** Combination of top-down and bottom-up approach is called Sandwich Approach.

1. **Non-incremental integration testing**

* Adding all the modules in a single shot and test the data flow between modules.
* **Drawback:**
* We might miss data flow between some of the modules.
* If you find any defect we can’t understand the root cause of defect.

**System Testing:**

* Testing all over functionality of the application with respective client requirement.
* Understanding client requirement is very very important here.
* It is a black box testing technique.
* This testing is conducted by testing team.
* After completion of components and integration level testing, we start system testing.
* Before conducting system testing, we should know the customers requirement.
* System testing focus on the below aspects.
* User interface Testing. (GUI)
* Functional Testing
* Non-Functional Testing.
* Usability Testing.

**User Acceptance Testing:**

* UAT is done by end users and if everything will be working fine then they will accept the software.
* Sometimes testers are also involved in UAT.
* After completion of system testing UAT team conduct acceptance testing in two levels.
* **Alpha Testing:** Users come back to the company and do alpha testing
* **Beta Testing:** After installing the software in their environment, they do beta testing.

**White Box & Black Box testing**

**White Box Testing**

White Box Testing means what is there internally we can see that.

It means we are able to see the code in white box testing.

Only developers can do white box testing.

White box testing aims to verify the correctness of the internal code, identify logical errors, and ensure that all code paths are tested.

**Black Box Testing**

Black box testing means what we can verify the functionality of the application / software whether it is working according to customer requirement or not.

In black box testing we don’t need to know programming because we don’t need to know the internal logic of the program, we just need to know how the functionality is working, how the flows is working.

Black box testing aims to validate the software's functionality, ensure it meets the specified requirements, and assess how it behaves from an end-user perspective.

**Static and Dynamic Testing**

**Static Testing:** Testing the project related documents in the form of review, walkthrough, inspections.

We use Review, Walkthrough, inspection to perform static testing. In first Second and third phase of UAT we use static testing

**Dynamic Testing:** Testing the actual software. We use Unit Testing, Integration Testing, System Testing, UAT.

**System testing types**

System testing is nothing but checking the overall functionality of the application with request to customers requirement.

**There are 4 types of System testing.**

GUI Testing

Usability Testing

Functional

Non-Functional Testing

**GUI Testing**

GUI testing is a process of testing the user interface of an application.

GUI includes all the elements such as menus, checkbox, button, colours, fonts, size, icons, contents, and images.

**GUI Testing Checklist**

* Testing the size, position, width, height of the elements.
* Testing of the error messages that are getting displayed.
* Testing the different sections of the screen.
* Testing of the font whether it is readable or not.
* Testing of the screen in different resolutions with the help of zooming in and zooming out.
* Testing the alignment of the texts and other elements like icons, buttons, etc. are in proper place or not.
* Testing the colours of the fonts.
* Testing whether the image has good clarity or not.
* Testing the alignment of the images.
* Testing of the spelling.
* The user must not get frustrated while using the system interface.
* Testing whether the interface is attractive or not.
* Testing of the scrollbars according to the size of the page if any.
* Testing of the disabled fields if any.
* Testing of the size of the images.
* Testing of the headings whether it is properly aligned or not.
* Testing of the colour of the hyperlink.
* Testing UI Elements like button, textbox, text area, check box, radio buttons, drop downs, links etc.

**Usability Testing**

Usability means easiness of the application.

* During this testing validates application provided context sensitive help or not to the user.
* Checks how easily the end users are able to understand and operate the application is called usability testing.

**Functional**

Functionality is nothing but the behaviour of the application

Functional testing talks about how your feature should work.

* **Object Properties Testing:**
* Check the properties of the object present on application.
* Every element in the web page having certain number of properties.
* Example: enable, disable, focus, visible, etc
* **Database Testing:**
* Checking database operations with respect to user operations.
* Also called backend testing
* Check DML operations is working properly or not.
* Insert, update, delete, select
* It is a grey box testing technique (Both white and Black)
* **Error Handling**
* Where we have to mainly focus on error messages.
* Tester verifies the error message white performing incorrect action on the applications.
* Error messages should be readable
* It should be in simple language or user understandable language
* **Calculations/Manipulations Testing**
* Tester should verify the calculations whether they are working correctly or not.
* **Links Existence & Links Execution**
* Links Existence – where exactly the links are placed
* Links Execution – Links are navigating to the proper page or not.
* Specially done for web Application.
* There are 3 types of links

Internal Links: Same page of the link but in different sections.

External Links: Link will navigate to some other page.

Brokens Links: It will not navigate to anywhere else, It doesn’t have target page.

* **Cookies & Sessions**
* Cookies – Cookies are temporary files created by browser while browsing the pages through internet.
* Sessions are time slot created by the server. (Made for security purpose)
* Session will be expired after some time. (If you are idle for some time)
* Specially done for web Application.

**Non-Functional Testing**

Once the functionalities are stable non-functional testing will be conducted.

It mainly focusses on customer expectations.

It is not easy special people required for doing non-functional testing.

* **Performance Testing**: Speed of the application

How well my application is responding

* **Load Testing**

Gradually increasing the load on the application slowly then check the speed of the application. (Load is nothing but the number of users)

* **Stress Testing**

Suddenly increase and decrease the load of the application then check the speed of the application.

* **Volume Testing**

How much data is able to handle by the application.

* **Security Testing:** How secure our application.
* Authentication: Users are valid or not.
* Authorization/Access control: Permissions of the valid users.
* Security Testing is done by security testing specialist.
* **Recovery Testing:** Check the system change abnormal to normal.

If somehow data is loss, then user should be able to recover the data.

* **Compatibility Testing:**
* Forward Compatibility:
* Backward Compatibility:
* Hardware Compatibility:
* **Installation Testing**
* Check screen is clear to understand.
* Screen navigation.
* Simple or not.
* Non-tech user can also able to install.
* When Un-installation is done all the files related to that software is uninstalled.
* **Sanitation/Garbage Testing:**

If any application provides extra feature or functionality then we consider them as bug.

**Functional vs Non-Functional Testing**

|  |  |
| --- | --- |
| **Functional Testing** | **Non-functional Testing** |
| Validates functionality of Software. | Verify the performance, security, reliability of the software. |
| Functionality describes what software does. | Non-Functionality describes how software works. |
| Concentrates on user requirement. | Concentrates on user expectation |
| Functional testing takes place before Non-functional testing. | Non-Functional testing performed after finishing Functional testing. |

**Re-Testing & Regression Testing**

**Re-Testing:** After fixing the bug check whether it is fix or not.

**Regression Testing:** After fixing the defect check whether it affect existing functionality or not is called as Regression testing. Regression testing can be done in 3 different ways.

There are three types of regression testing

**Unit Regression testing:** Testing only changes/ modification done by developers.

**Regional Regression Testing:** Testing the modified module along with impacted module.

**Full regression Testing:** Testing the main features and remaining part of the applications.

**Sanity and Smoke Testing:**

**Smoke Testing:**

* Basic functionality testing.
* Whether the particular software is installed or not.
* Check the basic navigation are working or not. (when open the URL getting the home page or not)
* Can be done by tester as well as developer
* After successfully completion of smoke testing only me move further

**Sanity Testing:**

* All the high-level functionality
* Can be done by tester only
* Sanity testing can only perform after smoke testing
* If smoke testing fail, we can’t do sanity testing

|  |  |
| --- | --- |
| Smoke Testing | Sanity Testing |
| Smoke Test is done to make sure the build we received from the development team is testable/stable or not | Sanity Test is done during the release phase to check for the main functionalities of the application without going deeper. |
| Smoke Testing is performed by both Developers and Testers | Sanity Testing is performed by Testers alone |
| Smoke Testing, build may be either stable or unstable | Sanity Testing, build is relatively stable |
| It is done on initial builds. | It is done on stable builds. |
| It is a part of basic testing. | It is a part of regression testing. |
| Usually, it is done every time there is a new build release. | It is planned when there is no enough time to do in- depth testing. |

**Exploratory Testing**

* Sometime It happen that the application is ready but the document of that application is not ready or to documents are not available. We have to check the application by exploring its functionality.
* We have to explore the application, understand completely and test it.
* Understand the application, identify all possible scenarios, document it then use it for testing.
* We do exploratory testing when the Application ready but there is no requirement.
* Test Engineer will do exploratory testing when there is no requirement.

**Drawbacks:**

* You might misunderstand any feature as a bug (or) any bug as a feature since you do not have requirement.
* Time consuming
* If there is any bug in application, you will never know about it.

**Adhoc Testing**

* Here also we don’t have any document, any requirement, we don’t have anything in our hand we just have an application.
* Here we have some knowledge about the application due to experience. (We have a knowledge about login page)
* Based on our knowledge we perform ADHOC Testing.
* It is performed on specific module.
* Testing application randomly without any test cases or any business requirement document.
* Adhoc testing is an informal testing type with an aim to break the system.
* Tester should have knowledge of application even thou he doesn't have requirements/test cases.

This testing is usually an unplanned activity.

**Monkey Testing**

* Here also we check the application randomly
* In adhoc testing we should have basic knowledge about the application but here we don’t have any knowledge about the application we check the application randomly.
* Testing application randomly without any test cases or any business requirement document.
* Adhoc testing is an informal testing type with an aim to break the system.
* Tester do not have knowledge of application
* Suitable for gaming applications.

|  |  |  |
| --- | --- | --- |
| **Adhoc Testing** | **Monkey Testing** | **Exploratory Testing** |
| No Documentation | No Documentation | No Documentation |
| No Plan | No Plan | No Plan |
| Informal testing | Informal testing | Informal testing |
| Tester should know Application functionality | Testers doesn't know Application functionality | Testers doesn't know Application functionality |
| Random Testing | Random Testing | Random Testing |
| Intension is to break the application/find out corner defects | Intension is to break the application/find out corner defects | Intension is to learn or explore functionality of application |
| Any Applications | Gaming Applications | Applications |

**Positive and Negative Testing**

**Positive Testing:**

* Testing the application with valid inputs is called as Positive Testing.
* It checks whether an application behaves as expected with positive inputs.
* For example -

Enter Only Numbers

99999

Positive Testing

* There is a text box in an application which can accept only numbers. Entering values up to 99999 will be acceptable by the system and any other values apart from this should not be acceptable.
* To do positive testing, set the valid input values from 0 to 99999 and check whether the system is accepting the values.

**Negative Testing:**

* Testing the application with invalid inputs is called as Negative Testing.
* It checks whether an application behaves as expected with the negative inputs.
* For example -

Enter Only Numbers

abcdef

Negative Testing

* Negative testing can be performed by entering characters A to Z or from a to z. Either software system should not accept the values or else it should throw an error message for these invalid data inputs.

**Requirement:**

* For Example, if a text box is listed as a feature and in FRS it is mentioned as Text box accepts 6-20 characters and only alphabets.

**Positive Test Cases:**

* Textbox accepts 6 characters.
* Textbox accepts up to 20 chars length.
* Textbox accepts any value in between 6-20 chars length.
* Textbox accepts all alphabets.

**Negative Test Cases:**

* Textbox should not accept less than 6 chars.
* Textbox should not accept chars more than 20 chars.
* Textbox should not accept special characters.
* Textbox should not accept numerical.

**End-to-End testing**

* Testing all over the functionalities of the system including data integration among all the modules is called as end-to-end Testing.
* This will cover every flow in the application.
* Example: If we take the example of email then we check after composing the email when we sent the email then the sent email must be visible in sent email menu and when we delete the email form the sent mail then it must be visible in recycle bin.

**Localization and Globalization testing**

**Globalization Testing:**

* Performed to ensure the system or software application can run in any cultural or local environment.
* Different aspects of the software application are tested to ensure that it supports every language and different attributes.
* It tests the different currency formats; mobile number formats and address formats are supported by the application.
* For example, Facebook.com supports many of the languages and it can be accessed by people of different countries. Hence it is a globalized product.

**Localization Testing:**

* Performed to check system or software application for a specific geographical and cultural
* environment.
* Localized product only supports the specific kind of language and is usable only in specific region.
* Its testes the specific currency format, mobile number format and address format is working properly or not.
* For example, Baidu.com supports only the Chinese language and can be accessed only by people of few countries. Hence it is a localized product.

**Test Design Techniques**

* It is used to prepare data for Testing.
* It is basically use for

1. Reduce the data
2. More coverage

* There are 5 types of test design technique

1. Equivalent class partitioning
2. Boundary Value analysis
3. Decision table-based technique
4. State transition
5. Error Guessing

**Equivalent class partitioning:**

* Partition data into various classes and we can select data according to class then test.
* It reduces the number of tests cases and saves time for testing.
* Check the values and classify the data into multiple class
* Example: If a text-box allow input of 1 to 500 number then we divide the input in 2 types valid input and invalid input. The number between 1 to 500 comes under valid input and number less than 1 or number greater than 500 comes under invalid input.

**Boundary Value analysis:**

In this technique we mainly focus on the boundary of the value.

It is used to check the boundary of input

In this technique we will only check the boundary of the values

Example: Enter a age between 18 to 35

For this we will only check the following conditions

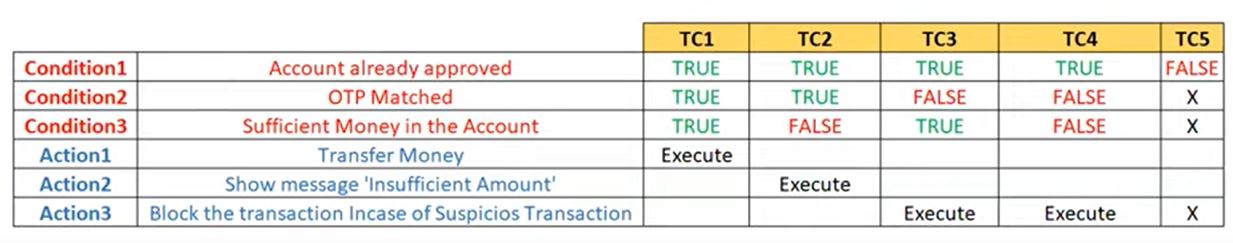
Min = 18 (Pass) Max = 35 (Pass)

Min – 1 = 17 (Fail) Max – 1 = 34 (Pass)

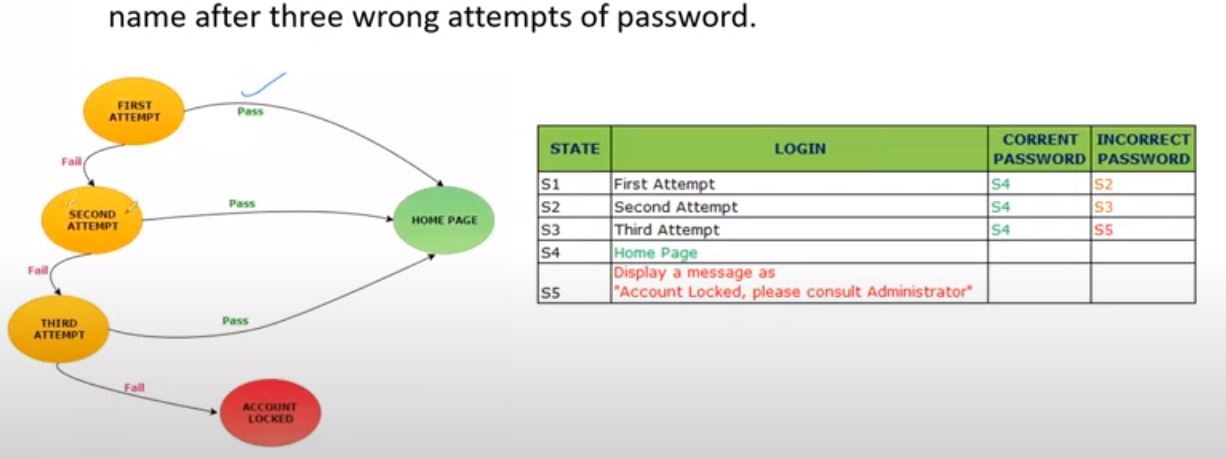
Min + 1 = 19 (Pass) Max + 1 = 36 (Fail)

**Decision Table testing:**

* This technique is used when we have different conditions.
* In this technique we have to perform different action based on different conditions.
* If we have a greater number of actions then we user decision table technique.
* Decision Table is also called as Cause-Effect Table.
* This technique will be used if we have more conditions and corresponding actions.
* In Decision table technique, we deal with combinations of inputs.
* To identify the test cases with decision table, we consider conditions and actions.
* Take an example of transferring money online to an account which is already added and approved.



**State Transition:**

* In this technique input will be changes base open the status.
* This technique is used to check the behaviour of the application.
* In State Transition technique changes in input conditions change the state of the Application.
* This testing technique allows the tester to test the behaviour of an AUT.
* The tester can perform this action by entering various input conditions in a sequence.
* In State transition technique, the testing team provides positive as well as negative input test values for evaluating the system behaviour.

**Error Guessing:**

* Error guessing is one of the testing techniques used to find bugs in a software application based on tester's prior experience.
* In Error guessing we don't follow any specific rules.
* It depends on Tester Analytical skills and experience.
* Some of the examples are:
* Submitting a form without entering values.
* Entering invalid values such as entering alphabets in the numeric field.

.

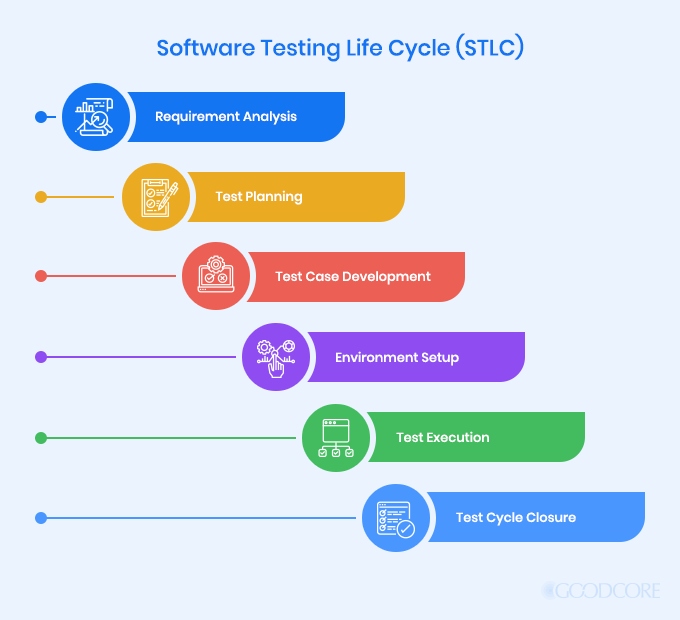
**STLC**

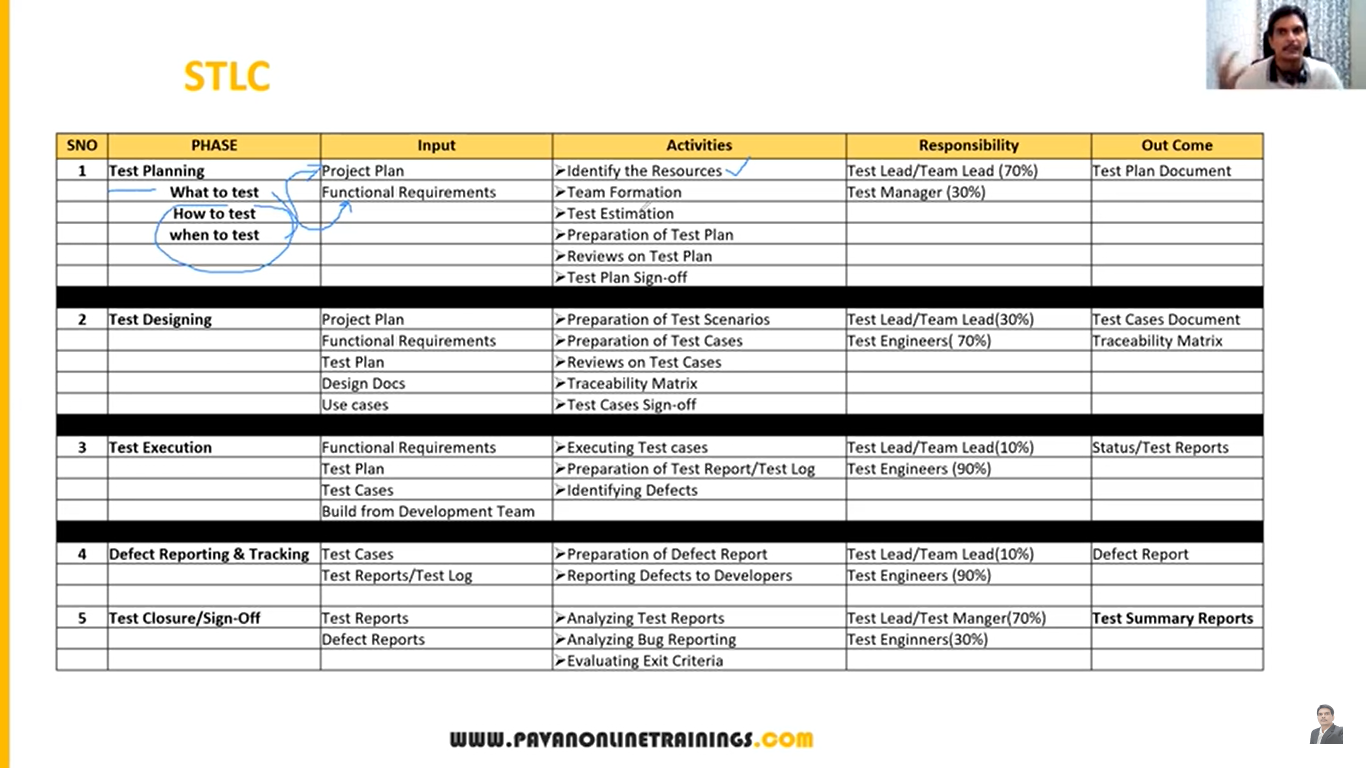
**STLC:**

STLC in the one of the phases in SDLC that deals with the testing part of the software.

Just like SDLC, STLC also consists of some phases which are as follows.

1. Requirement analysis
2. Test Planning
3. Test Designing
4. Test Execution/ Environment setup
5. Defect/ Bug reporting and tracking
6. Text closure





**Contents in Test plan**

A Test plan is a document that describe the test scope, test strategy, test objective, schedule, deliverables and resource required to perform testing for a software product.

**Test plan template contents:**

* **Overview** (What is the project why we are preparing test plan document)
* **Scope** (Test environment like what to test and what not to test)
* Inclusions (What we are going to test)
* Test Environments (In which environment we are going to test)
* Exclusions (What we are not going to test)
* **Test Strategy** (what type of testing we are going to conduct like smoke, sanity, etc)
* **Defect Reporting Procedure** (how we are going to report the bug)
* **Roles/Responsibilities**
* **Test Schedule** (when we have to conduct what type of testing like end date & start date of every testing)
* **Test Deliverables** (each and every phase of testing we are going to delivered certain document checking which type of document we are delivering in each phase)
* **Pricing** (how much pricing it will be take)
* **Entry and Exit Criteria** (when we have to start testing when we have to end testing)
* **Suspension and Resumption Criteria** (some time we stop testing like something is breaking, something is not working then after recovery we start testing so in which case we have to stop testing and we have to start testing)
* **Tools** (what king of tool we are going to testing also even manual tool like excel document, automation testing tools)
* **Risks and Mitigations** (what is the risk and what is solution for that particular risk)
* **Approvals**

**Use cases, Test scenarios and Test case**

**Use Case:**

* Use case describes the requirement.
* This will help us to understand the requirement more clear.
* Use case is like a picture or data flow by which we can understand the requirement more clear
* Use case contains THREE Items.
* **Actor**, which is the user, which can be a single person or a group of people, interacting with a process.
* **Action,** which is to reach the final outcome
* **Goal/Outcome**, which is the successful user outcome.
* Tester will not write use cases
* Project manager, product owner, or who written the SRS document will write the test cases.
* It describes the functional document
* It is prepared by BA

**Test Scenario:**

* A possible area to be tested (What to test)
* Where we have to conduct testing.

**Test Case:**

* Step by step actions to be performed to validate functionality of AUT (How to test).
* Test case contains test steps, expected result & actual result.
* For each scenario we write multiple test cases
* It describes the test step, procedure, prepared by test engineers.

**Test Suite:**

It is a group of test cases which belongs to same category.

**Content in Test case**

A test case is a set of actions executed to validate particular feature or functionality of your software application.

**Test case content**

* Test Case ID
* Test Case Title
* Description
* Pre-condition
* Priority (PO, P1, P2, P3) - order
* Requirement ID
* Steps/Actions
* Expected Result
* Actual Result
* Test data

**RTM**

**What is RTM (Requirement Traceability Matrix)?**

RTM describes the mapping of Requirement's with the Test cases.

The main purpose of RTM is to see that all test cases are covered so that no functionality should miss while doing Software testing.

Tester will prepare it

**Requirement Traceability Matrix - Parameters include**

Requirement ID

Req Description

Test case ID's

**Test environment and Execution**

**Test environment:**

* Test Environment is a platform specially build for test case execution on the software product.
* It is created by integrating the required software and hardware along with proper network configurations.
* Test environment simulates production/real time environment.
* Another name of test environment is Test Bed.
* Where the tester use the software in there environment. This environment must be similar to the customers environment

**Test Execution:**

* During this phase test team will carry out the testing based on the test plans and the test cases prepared.

**Entry Criteria:** Test cases, Test Data & Test Plan

**Activities:**

* Test cases are executed based on the test planning.
* Status of test cases are marked, like Passed, Failed, Blocked, Run, and others.
* Documentation of test results and log defects for failed cases is done.
* All the blocked and failed test cases are assigned bug ids.
* Retesting once the defects are fixed.
* Defects are tracked till closure.

**Deliverables**: Provides defect and test case execution report with completed results.

**Guidelines for test execution:**

* The Build being deployed to the QA environment is the most important part of the test execution cycle.
* Test execution is done in Quality Assurance (QA) environment.
* Test execution happens in multiple cycles.
* Test execution phase consists Executing the test cases + test scripts (if automation).

**Defect/Bug**

* Any mismatched functionality found in a application is called as Defect/Bug/Issue.
* Difference between expected and actual outcome.
* During Test Execution Test engineers are reporting mismatches as defects to developers through templates or using tools.
* Defect Reporting Tools:
* Clear Quest
* DevTrack (Bug tracking & defect reporting tool)
* Jira
* Quality Center
* Bug Jilla etc. (Bug tracking & defect reporting tool)

**Defect Reporting**

**Defect\_ID** - Unique identification number for the defect.

**Defect Description** - Detailed description of the defect including information about the module in which defect was found.

**Version** - Version of the application in which defect was found.

**Steps** - Detailed steps along with screenshots with which the developer can reproduce the defects.

**Date Raised** - Date when the defect is raised

**Reference**- where you Provide reference to the documents like. requirements, design, architecture or may be even screenshots of the error to help understand the defect

**Detected By** - Name/ID of the tester who raised the defect

**Status** - Status of the defect, more on this later

**Fixed by** - Name/ID of the developer who fixed it

**Date Closed** - Date when the defect is closed

**Severity** which describes the impact of the defect on the application

**Priority** which is related to defect fixing urgency. Severity Priority could be High/Medium/Low based on the impact urgency at which the defect should be fixed respectively

**Defect Classification**

Defect is categories in two types

1. Severity
2. Priority

**Severity**

Severity describes the seriousness of defect and how much impact on Business workflow.

Decided by tester

**Defect severity can be categorized into four class**

1. **Blocker** (Show stopper): This defect indicates nothing can proceed further.
   * Ex: Application crashed, Login Not worked
2. **Critical**: The main/basic functionality is not working. Customer business workflow is broken. They cannot proceed further.

Ex1: Fund transfer is not working in net banking

Ex2: Ordering product in ecommerce application is not working.

1. **Major**: It causes some undesirable behaviour, but the feature/application is still functional.
   * Ex1: After sending email there is no confirm message

Ex2: After booking cab there is no confirmation.

1. **Minor**: It won't cause any major break-down of the system

Ex: Look and feel issues, spellings, alignments.

**Priority**

Priority describes the importance of defect.

Defect Priority states the order in which a defect should be fixed.

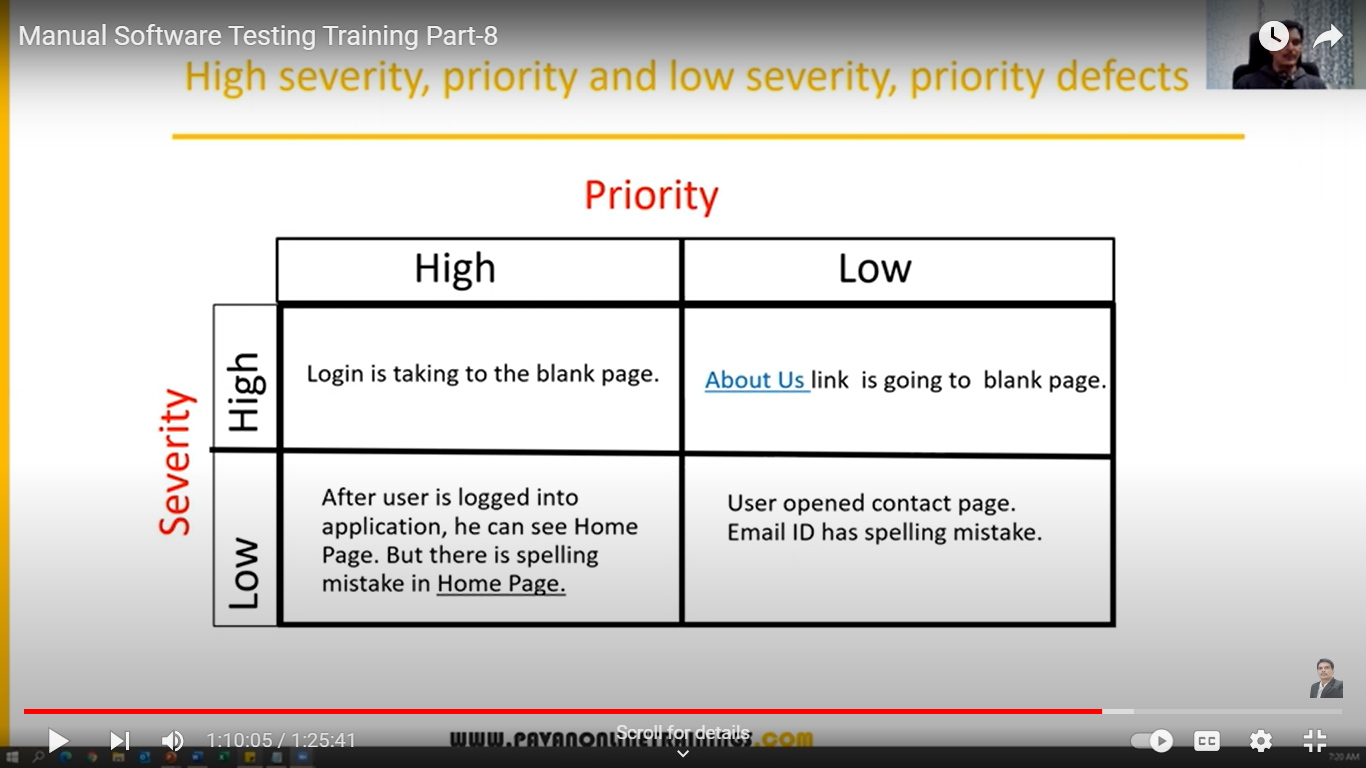
Decided by developer

**Defect priority can be categorized into three class**

**PO** (High): The defect must be resolved immediately as it affects the system severely and cannot be used until it is fixed.

**P1** (Medium): It can wait until a new versions/build is created

**P2** (Low): Developer can fix it in later releases.



Low priority-Low severity - A spelling mistake in a page not frequently navigated by users.

Low priority-High severity - Application crashing in some very corner case.

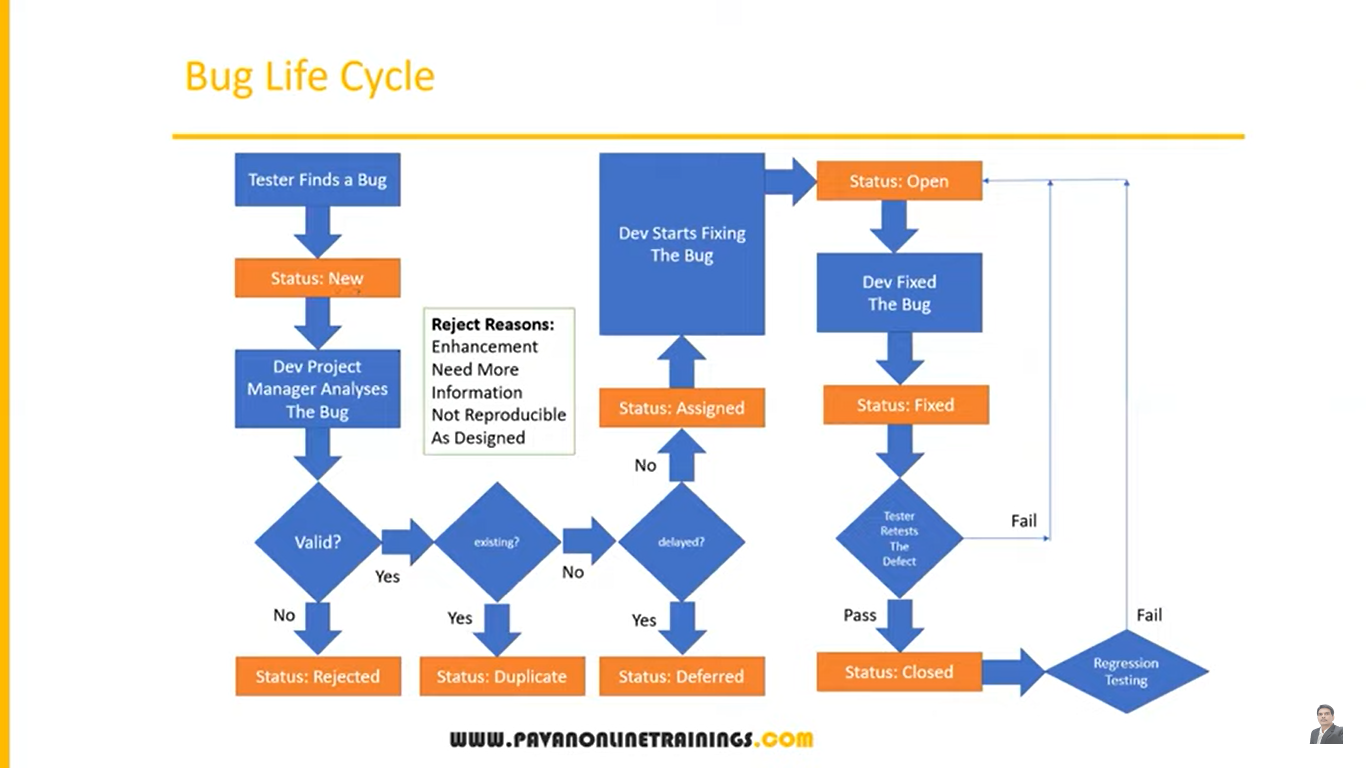
High priority-Low severity - Slight change in logo color or spelling mistake in company name.

High priority-High severity - Issue with login functionality. (user is not able to login to the application)

High Severity- Low Priority - Web page not found when user clicks on a link (user does not visit that page generally)

Low Priority- Low Severity - Any cosmetic or spelling issues which is within a paragraph or in the page

**Defect Life Cycle**



**Test Closure**

**Activities**

Evaluate cycle completion criteria based on Time, Test coverage, Cost, Software, Critical Business Objectives, Quality

Prepare test metrics based on the above parameters.

Document the learning out of the project

Prepare Test summary report

Qualitative and quantitative reporting of quality of the work product to the customer.

Test result analysis to find out the defect distribution by type and severity.

**Deliverables**

Test Closure report

Test metrics

**Test Metrics**

Whenever we do some task we have to measure that we have to trace that

Whatever task we have done we have to make its matrix

We have to calculate matrix in % formulas

**Require data for test matrix**

* No. Of Requirements
* Avg. No. of Test Cases written Per Requirement
* Total No. of Test Cases written for all Requirement
* Total No. Of test cases Executed
* No. of Test Cases Passed
* No. of Test Cases Failed
* No. of Test cases Blocked
* No. Of Test Cases Un Executed
* Total No. Of Defects Identified
* Critical Defects Count
* Higher Defects Count
* Medium Defects Count
* Low Defects Count
* Customer Defects
* No. of defects found in UAT

**Formulas for finding Test Matrix**

* % of Test cases Executed: No. of Test cases executed / Total No. of Test cases written) \* 100
* % of test cases NOT executed: (No. of Test cases NOT executed/Total No. of Test cases written) \* 100
* % Test cases passed (No. of Test cases Passed /Total Test cases executed) \* 100
* % Test cases failed (No. of Test cases failed / Total Test cases executed) \* 100
* %Test cases blocked (No. of test cases blocked / Total Test cases executed) \* 100
* Defect Density: Number of defects identified per requirement/s
* No. of defects found / Size (No. of requirements)
* Defect Removal Efficiency (DRE):
* (A/A+B) \* 100
* (Fixed Defects / (Fixed Defects + Missed defects)) \* 100
* A- Defects identified during testing/ Fixed Defects
* B- Defects identified by the customer/Missed defects
* Defect Leakage:
* (No. of defects found in UAT / No. of defects found in Testing) \* 100
* Defect Rejection Ratio:
* (No. of defect rejected /Total No. of defects raised) \* 100
* Defect Age: Fixed Date-Reported date
* Customer satisfaction = No. of complaints per Period of time

**QA Responsibilities**

* Understanding the requirements and functional specifications of the application.
* Identifying required Test Scenario's.
* Designing Test Cases to validate application.
* Setting up Test Environment (Test Bed)
* Execute Test Cases to valid application
* Log Test results (How many tests cases pass/fail).
* Defect reporting and tracking.
* Retest fixed defects of previous build
* Perform various types of testings in application.
* Reports to Test Lead about the status of assigned tasks
* Participated in regular team meetings.
* Creating automation scripts.
* Provides recommendation on whether or not the application / system is ready for production.