SOFTWARE TESTING TOOLS

About Our Curriculum:

Career Benefits:

There are many career benefits on completing this Software Testing Tools training and its concepts from **Quality Thought** which are mentioned as below: The software testing training has been designed and developed very sincerely to meet the expectation of the professionals and the students who are curious to learn software testing. In the current job market, there are various open positions for software testers. Some organizations look for the testers who are having experience in manual testing while some of them need the testers who are having hands-on experience in automation testing. Our Software Testing Course Offline/Online will train you on both the approach of testing: Manual and Automation [Selenium with Java]. After completing this software testing training, you will be able to work as a software tester who is amply capable to handle the entire software testing process. You will always get preference among the other testers as you will be knowing several testing methodologies after finishing the course.

Goals:

This course is mainly designed for beginners and professionals who want to endorse their skills in manual and automation testing along with which you will be explored to the process of testing and different type of testing applications. The main goal of this course is to successfully advance your career in the manual and automation testing domain.

Objectives:

The objective of this course almost involves all the main concepts related to testing. In this course you will master in testing tools, manual testing concepts from scratch along with real-time execution, different types of testing methods, STLC, SDLC, AGILE, Defect Life Cycle, Jira, Mobile App Testing. Selenium IDE concepts from scratch along with its installation, understanding the components and commands, selenium web driver for managing web elements, etc. The main and important section in this course is advanced UI techniques and some hands-on projects to implement these topics at the end of each section.

Course Highlights:

This course has brief and consistent topics covered which are important to meet the goals and objectives of any organization for the beginners and professionals to get the job and understand the technology properly. Here are the concepts are given below that are covered throughout this course:

In this course, manual testing from requirements analysis, test scenarios preparation, test case preparation, test data preparation, test execution, result analysis, defect reporting, tracking,

retesting and SDLC, STLC, DLC life cycles with agile model scrum framework and Jira tool and also you will be trained for interview questions related to this topic. In automation testing (regression testing) first offal giving complete and strong knowledge on Core java with complete OOPS concepts, introduced to the selenium IDE along with installation, Selenium web driver for managing web elements. In this course, you will get to learn a skill with selenium in this section it will cover from basics to advanced concepts along with working of locators, customization of Xpath in locators, how to create TestNg with examples and also you will be trained for interview questions related to this topic.

Project:

In this course, there few projects for each section so that the students can understand the concepts and implement those concepts easily by the knowledge given during this course. These projects are mainly to understand case studies for different fields.

Target Audience:

Students of Engineering in Computers or any Bachelors: Any student of BTech or B.E in engineering or any other bachelor's degree or Master Degree can choose this course. This Software Testing Tools training is recommended to any learner who is very interested in learning. This course is also designed to advance or elevate the career of the learners or to secure any job opportunity as a Manual and Automation Test Engineer or Selenium Tester. Software Testing training to become a Test Manual and Automation Engineer or QA Engineer or Test Architect in any Testing based applications or Test Automation related QA roles in the larger MNCs or organizations.

TABLE OF CONTENTS

Manual Testing Course Syllabus:

Module1: Software Testing Fundamentals

Module 2: Software Testing Life Cycle–(STLC)

Module 3: Test Design

- +Project Management plan (PMP)
- -Architecture of Current Project
- +Test Scenarios
- Test Scenario Entry and Exit Criteria
- Test Scenario Template
- Test Scenarios Identification
- Writing Test Scenarios for application
- +LAB Checklist
- Creation and working with Folder Structure
- +Test Cases
- Test cases Entry and Exit Criteria
- Test cases Template
- Test cases Identification
- Writing Test cases for application
- Good Test Case design steps
- Test Case Design Techniques
- Boundary Value Analysis
- Equivalence Class Partitioning
- State Transition
- Decision Table
- Test Case Execution
- Build Release process
- Testing Vs Debugging
- +Status Reports Process
- Daily Status Report
- Daily Defect Report
- Weekly Status Report
- Re-Testing Status Report
- +Test Closure
- Test Metrics
- Test Summary Reports
- Requirements Traceability Matrix (RTM)
- When testing need to be stopped

Module 4: Testing Methods

- Methods
- Of Testing
- +White Box Testing
- Unit Testing
- Integration Testing
- +Black Box Testing
- System Testing
- User Acceptance Testing
- Alpha Testing
- Beta Testing
- +Functional Testing Types
- Smoke Testing / Sanity Testing
- Formal Testing
- Ad-hoc Testing
- Re-Testing
- Regression Testing
- Static Testing
- Dynamic Testing
- System Integration Testing
- End-to-End Testing
- Exploratory Testing
- Monkey Testing
- +Non-Functional Testing Types
- UI Testing
- Usability Testing
- Security Testing
- Compatibility Testing
- Load Testing
- Performance Testing
- Globalization Testing
- Localization Testing
- Recovery Testing

Module 5: Defect management

- +Defect Life Cycle
- Defects Reporting
- Defects Reporting Template
- Defects Reporting & Re-Testing
- Defect Severity & Defect Priority
- Defects Closing
- Defect /Bug/Error/Failure
- Defects in Real Time application
- +Test/Project management Tool: JIRA
- JIRA Introduction
- Features of Jira
- Defect Reporting using Jira
- Jira Dashboard
- Creating Product backlog in JIRA
- Creating EPICS in JIRA
- Creating User Stories in JIRA
- Writing TestCases in JIRA
- Executing TestCases from JIRA
- Adding Bugs to the JIRA Project

Module 6: Agile Methodology

- +Agile Methodology
- What is agile?
- Agile Testing principles
- Scrum Introduction
- Roles of Scrum Master
- Sprint Planning
- Product Owner
- Product Backlog
- Concept of User Stories
- Writing User Stories- Examples
- Defect Backlog
- Project Backlog meeting
- Daily SCRUM

Module 7: Mobile APP Testing

- +Mobile APP Testing
- Mobile Apps Testing (Android & IOS)
- Mobile Apps Testing (Responsive)
- Mobile Testing Strategy
- Mobile Testing using Emulators

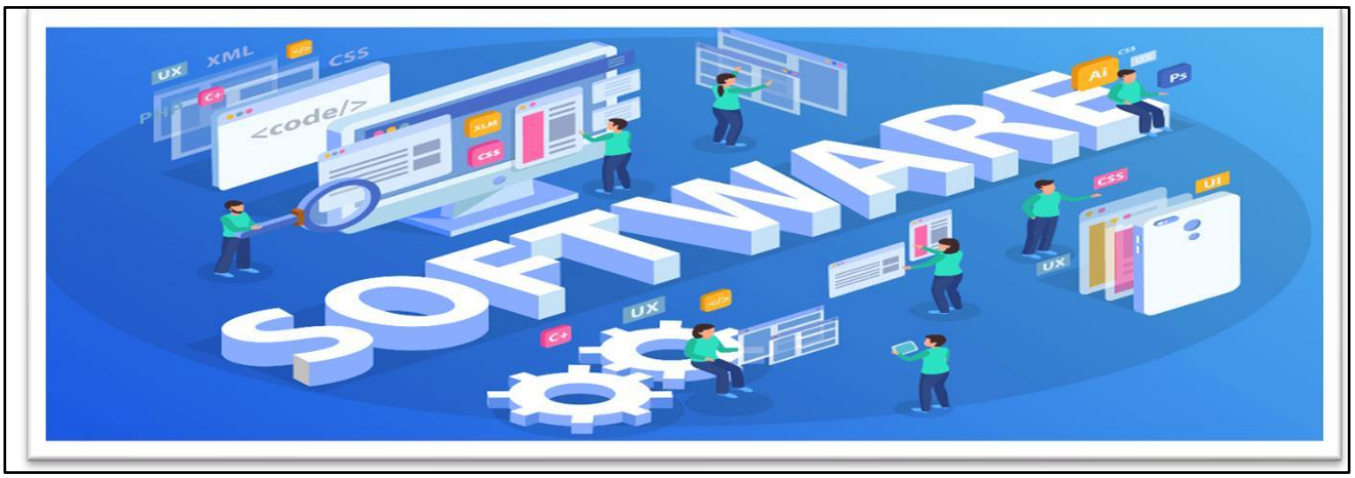
Manual Testing

Module 1: Software Testing Fundamentals:

Software Definition: Software is a set of instructions, data, or programs used to operate a computer and execute specific tasks.

The two main categories of software are application software and system software.

System software governs a computer's internal operation, mainly by operating systems and controls peripherals such as displays, printers, and storage devices. In contrast, application software guides the machine to perform user-given commands and can assume to include any program that processes a user's data.



Types of Software's:

1.System Software:
System software is a type of computer program that is designed to run a computer's hardware and application programs. A system software aids the user and the hardware to function and interact with each other. In simple words, we can say that system software is an intermediator or a middle layer between the user and the hardware because hardware understands machine language (i.e., 1 or 0) whereas user applications are work in human-readable languages like English, Hindi, German, etc. so system software converts the human-readable language into machine language and vice versa.

When you first turn on the computer, it is the system software that gets initialized and gets loaded in the memory of the system. The system software runs in the background and is not used by the end-users. The operating system is the best-known example of system software, it manages all the other computer programs.

Ex: Device drivers, Operating Systems, Servers, Utilities, etc…

2.Programing Software:
Programming software is a software which helps the programmer in developing other software. Computer programmers use programming software to write code. Programming software and

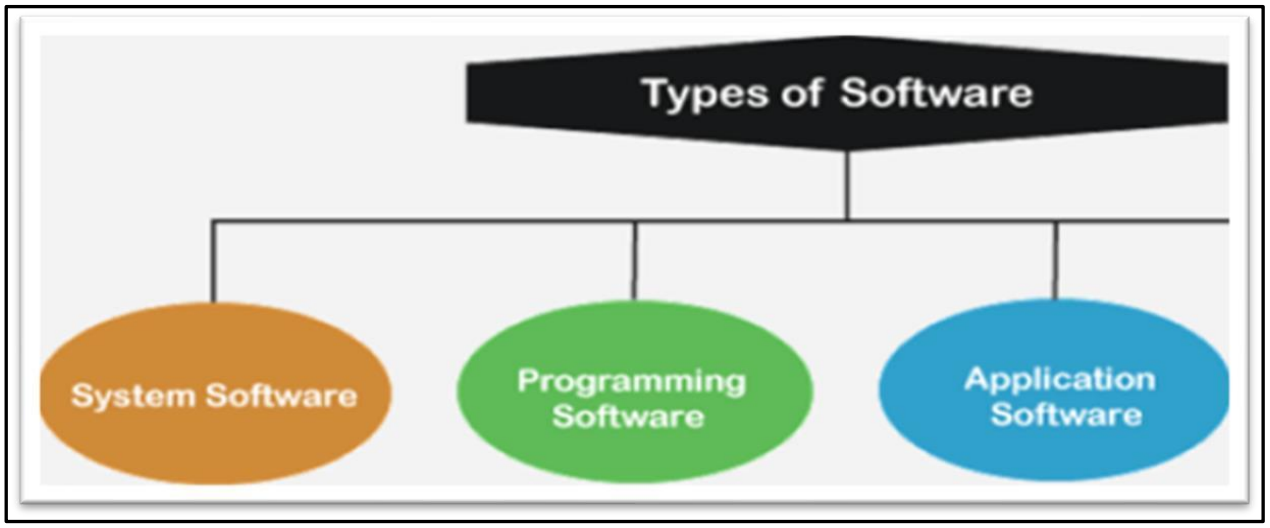
programming tools enable developers to develop, write, test and debug other software programs. Examples of programming software include assemblers, compilers, debuggers and interpreters. Integrated development environments (IDEs) are combinations of all these software. Programming software is also known as programming tool or software development tool.

Ex: Compilers, debuggers, interpreters, etc…

3. Application Software:

Application software is designed to perform a specific task for end-users. Application Software is also called simply apps. An application can be self-contained, or it can be a group of programs that run the application for the user. All the apps that we see on our mobile phones are also examples of Application Software. Examples of modern applications include office suites, graphics software, databases and database management programs, web browsers, word processors, software development tools, image editors and communication platforms.

Ex: Web Applications, Mobile Apps, Desktop Applications etc….



Risks In Software:

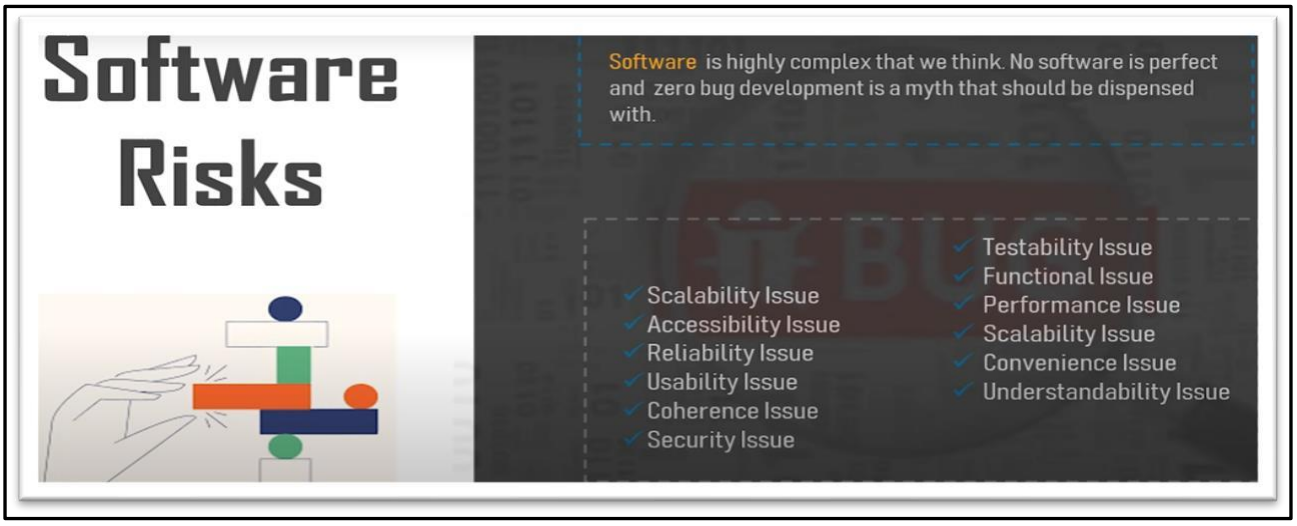
"Tomorrow problems are today's risk." Hence, a clear definition of a "risk" is a problem that could cause some loss or threaten the progress of the project, but which has not happened yet.

“Risk is an uncertain future event with a probability of occurrence and potential for loss”

Risk identification and management are the main concerns in every software project. Effective analysis of software risks will help in effective planning and assignment of work.

Are you developing any Test plan or Test strategy for your project? Have you addressed all the risks properly in your Test plan or Test strategy?

- Aggressive deadlines
- Business risks
- Categories of Risks
- Environmental risks
- Known risks
- Legal risks
- Low productivity
- Market risks
- Operational Risks
- Predictable risks
- Programmatic Risks
- Project risks:
- Schedule Risk
- Stakeholder issues
- Technical Risks
- Third-party risk
- Unpredictable risks



Software Process and benefits: A software process (also known as software methodology) is a set of related activities that leads to the production of the software. These activities may involve the development of the software from the scratch, or, modifying an existing system.

A software process is the set of activities and associated outcome that produce a software product.

There are Four main key activities:

⇒ Software specifications: The functionality of the software and constraints on its operation must be defined.
⇒ Software development: The software to meet the requirement must be produced.
⇒ Software validation: The software must be validated to ensure that it does what the customer wants.
⇒ Software evolution: The software must evolve to meet changing client needs.

Application or Software Application:

In information technology, an application, also referred to as an application program or app for short application or software that runs on your computer. It is a computer software package that performs a specific function or specific task directly for an end user or, in some cases, for

another application. An application can be self-contained or a group of programs. Depending on the activity for which it was designed, an application can manipulate text, numbers, audio, graphics, and a combination of these elements. Some application packages focus on a single task, such as word processing; others called integrated software include several applications.

Software application is basically categorized in-two types:

1. **Project** **2.Product**

1.Project: If a software application is developed for a specific customer requirement, then it is called a project. A project is a set of tasks that must be completed in order to arrive at a particular goal or outcome. It means that exact rules i.e the customer requirements must be followed. A project is temporary in that it has a defined beginning and end in time, and therefore defined scope and resources.

→ Requirements gathered from particular client.
→ End user is one.
→ Develop the application for particular client.

Examples: Banking projects like ICICI, HDFC and e-commerce projects like flipkart.com and bigbasket.com. These are specific to clients.

2.Products: If a software application developed for multiple customer requirements, then it is called a product. This is based on the general requirements i.e on our own requirements. A product is an object or system made available for consumer use; it is anything that can be offered to a market to satisfy the desire or need of a customer.

→ Requirements gathered from market survey.
→ End users are more than one.
→ Develop the application for Global clients.

Examples: Google products like Gmail, Drive (Free sources) and Oracle products like Databases etc.

Difference between Product and Project:

Product	Project
A way of answering a specific need	A way of executing plan
Long term	Short term
Focus on creating a sustainable environment	Focus on fixed constraints (scope, resources, time, budget)
No fixed end date, continuous discovery and development needed	Has a defined start and end date
Unpredictable	Appears predictable
Team stays together long term with slight changes over time	Team stays for the duration of the project

Types of Applications:

There are mainly 4 types of applications:

1. Standalone Application:

Standalone applications are also known as desktop applications or window-based applications. These are traditional software that we need to install on every machine. Examples of standalone application are Media player, antivirus, etc. AWT and Swing are used in Java for creating standalone applications.

2. Web Application:

An application that runs on the server side and creates a dynamic page is called a web application. Web application is a client server application which is run by the client. Web applications include mail, online retail sales, Wikipedia information's etc. Currently, Servlet, JSP, Struts, Spring, Hibernate, JSF, etc. technologies are used for creating web applications in Java.

3. Enterprise Application:

Enterprise application is related to middle ware applications. To enable software application and hardware systems we are using technologies and services across the enterprises. So, we are calling it as enterprise application. Enterprise applications are mainly designed for

corporate sides such as banking business systems. It has advantages like high-level security, load balancing, and clustering.

4. Mobile Application:

Mobile application is developed for run the mobile phones and tablets. An application which is created for mobile devices is called a mobile application. Currently, Android and Java ME are used for creating mobile applications.

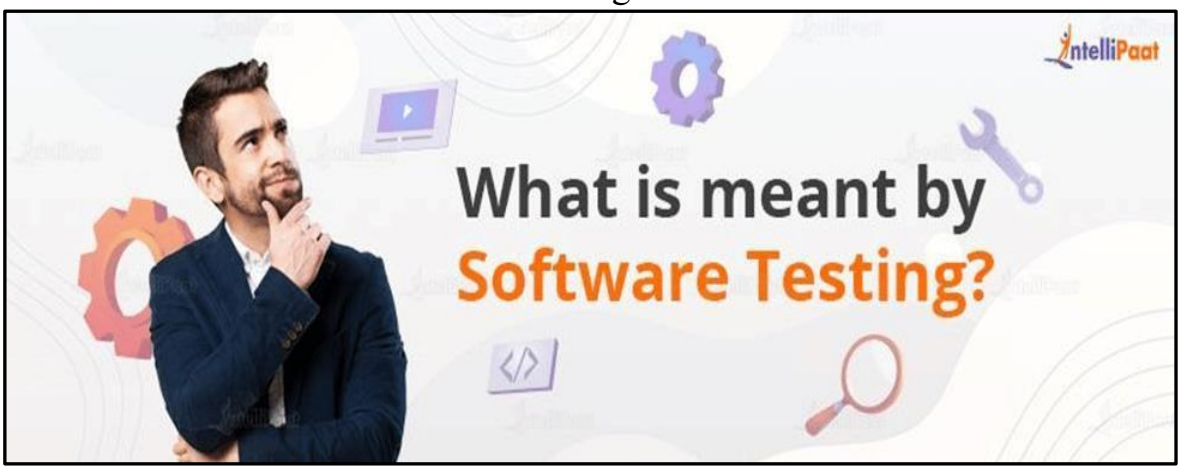
Testing: It is a process of verifying if we are developing the right product or not and validating if the developed product is right or not.

Introduction to Software Testing:

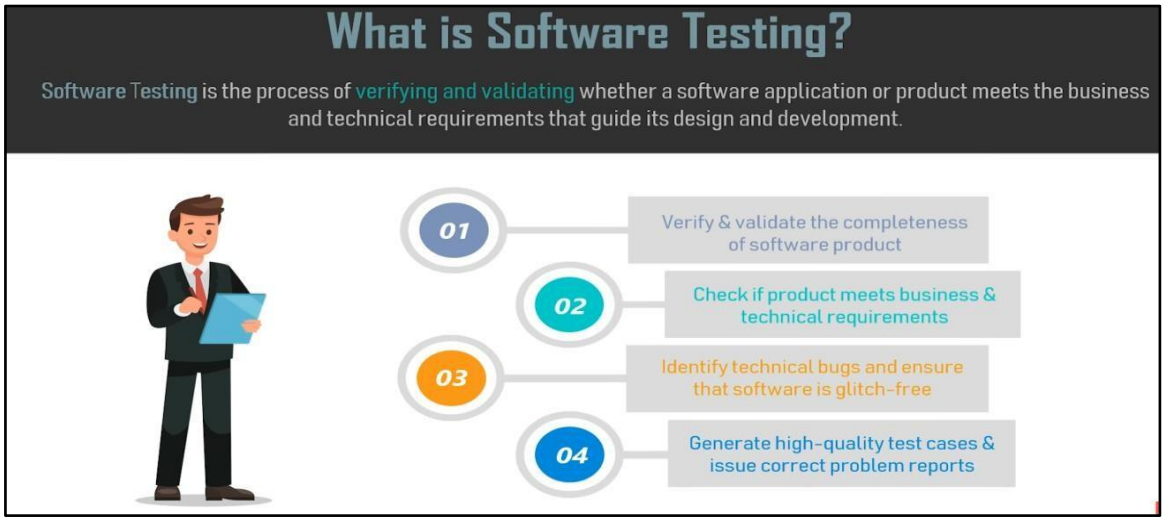
Software testing is a check activity to validate whether the actual results match with expected results and to ensure that the software system is bug free.

Definition: "To test the developers developed application/software is working according to client requirements or not".

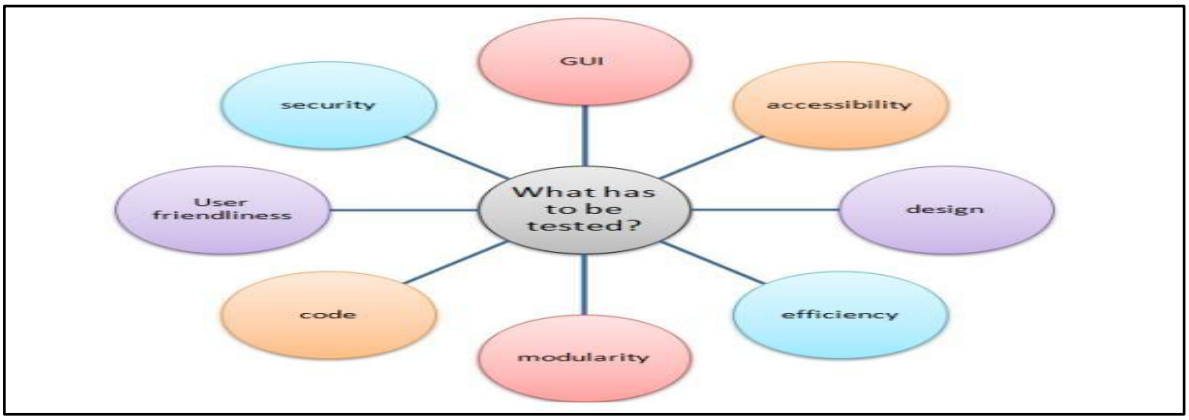
"Software testing is the process of assessing the functionality of a software program. The process checks for errors and gaps and whether the outcome of the application matches desired expectations before the software is installed and goes live".



Whenever an application comes into the market, and it is unstable or having a bug or issues or creating a problem while end-users are using it. If we don't want to face these kinds of problems, we need to perform one round of testing to make the application bug free and stable and deliver a quality product to the client, because if the application is bug free, the end-user will use the application more conveniently.



Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free. It involves execution of software/system components using manual or automated tools.



Software Testing is Important because if there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product. Properly tested software product ensures reliability, security and high performance which further results in time saving, cost effectiveness and customer satisfaction.

Why Testing Required:

Software Testing is necessary because we all make mistakes. Some of those mistakes are unimportant, but some of them are expensive or dangerous. We need to check everything and anything we produce because things can always go wrong – humans make mistakes all the time.

Since we assume that our work may have mistaken, hence we all need to check our own work. However, some mistakes come from bad assumptions and blind spots, so we might make the same mistakes when we check our own work as we made when we did it. So, we may not notice the flaws in what we have done.

"Ideally, we should get someone else to check our work because another person is more likely to spot the flaws".

By detecting errors and mistakes that affect the quality of software, these testers can ensure that software products are worthy of being sold in the market. It's also important to note that the process of software testing is generally only a company's responsibility if they are software developers creating software for the market.

"Software testing is the process of evaluating and verifying that a software product or application does what it is supposed to do. The benefits of testing include preventing bugs, reducing development costs and improving performance".

For now, let's list the important reasons as to why software testing must be considered mandatory:

- To gain customer confidence
- To check software adaptability
- To identify errors
- To avoid extra costs
- To accelerate software development
- To reduce risks
- To optimize business
- To increase the quality
-

Objectives of Testing:

- To conform whether the application developed is according to the customer requirements or not.
- Finding defects.
- To make sure that the product is error free and all problems are resolved and closed.
- Finally testing is helpful to deliver a quality and risk-free product to the customer.

Benefits of Software Testing:

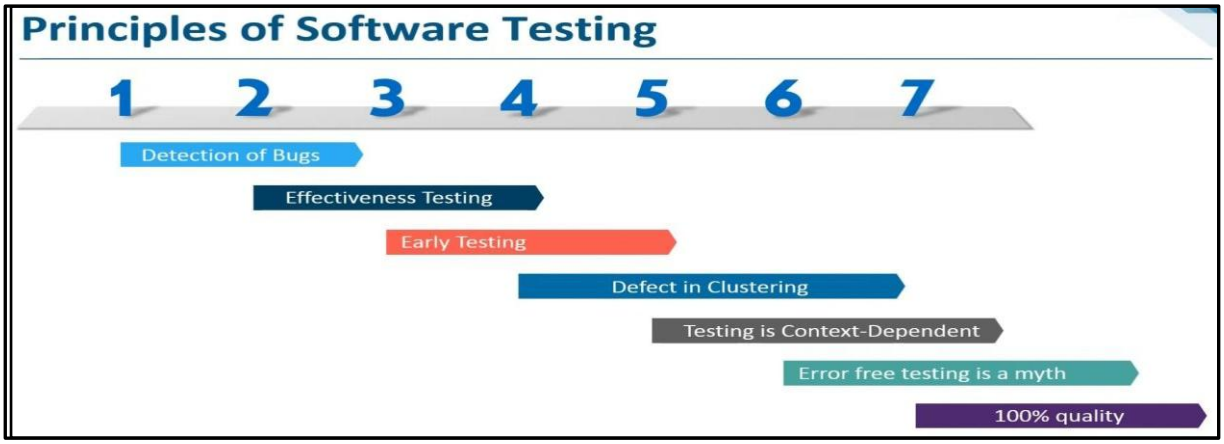
The benefits of testing include preventing bugs, reducing development costs and improving performance.

1.Cost-Effective: It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.

2.Security: It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.

3.Product quality: It is an essential requirement of any software product. Testing ensures a quality product is delivered to customers.

4.Customer Satisfaction: The main aim of any product is to give satisfaction to their customers. UI/UX Testing ensures the best user experience.



Why Software has Defects:

Unclear requirements and misinterpretation of requirements are the two major factors that cause defects in software.

Also, defects are introduced in the development stage if the exact requirements are not communicated properly to the development teams.

1. Communication problems: Miscommunication or lack of communication during various stages of the software development process.

2. Human errors: Humans are prone to errors and quite naturally, expecting the products they develop to be flawless and without errors/defects would be foolish.

3. Impractical development timeframe: Insufficient or limited resources, unrealistic release schedules, and project deadlines.

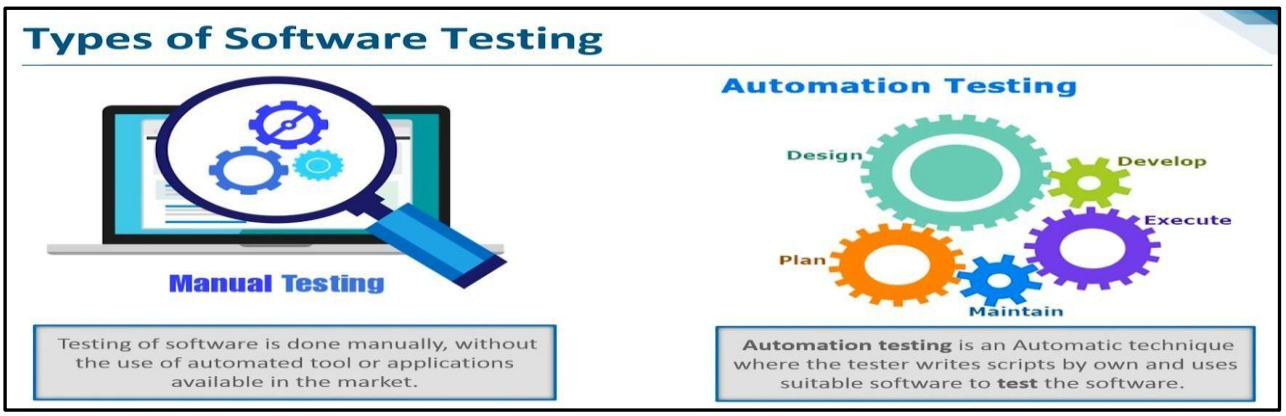
4. Lack of skilled testing: In several companies, poor testing is often the norm, which may include scarcity of skilled testers, shortcomings in the testing process that's followed, testing being taken lightly, and the process getting conducted without giving it much importance etc.

5. Last-minute changes: Just before a product release, changes made to the infrastructure, requirement, tools, platform etc can be dangerous. Actions such as testing the compatibility of your software across a variety of browsers and operating systems, database migration etc are complex things and if done in a hurry (due to a last-minute change), they may introduce bugs/ errors in the software/application.

Different ways of Software Testing:

The software testing mainly divided into two ways, which are as follows:

1. Manual Testing
2. Automation Testing



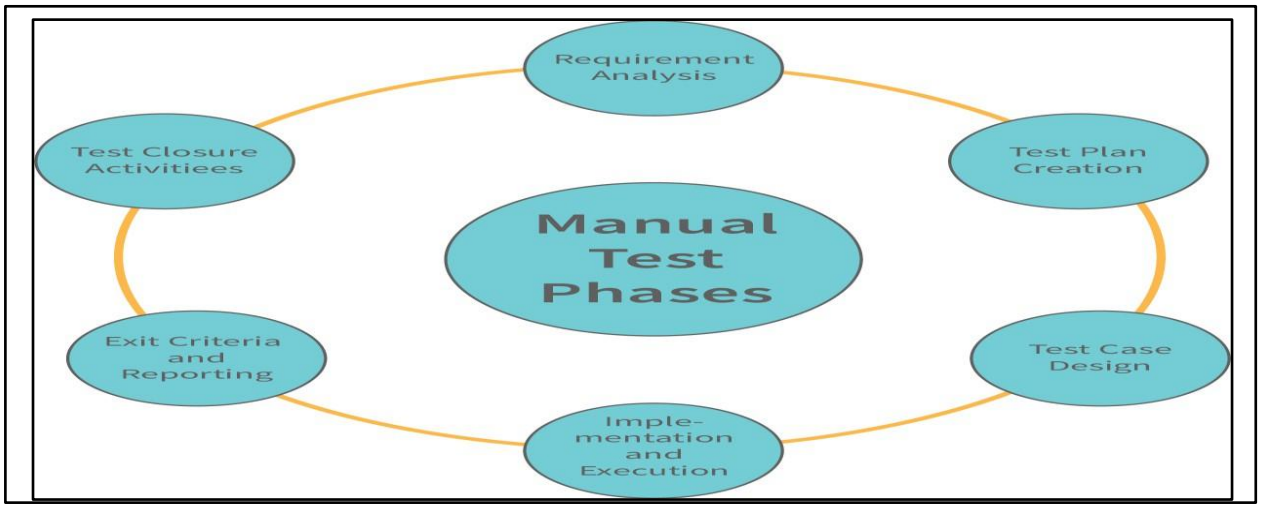
1.Manual Testing:

Manual testing is a software testing process in which test cases are executed manually without using any automated tool. All test cases executed by the tester manually according to the end user's perspective. It ensures whether the application is working, as mentioned in the requirement document or not. Manual Testing is one of the most fundamental testing processes as it can find both visible and hidden defects of the software. Test cases are planned and implemented to complete almost 100% of the software application. Test case reports are also generated manually. Manual testing is mandatory for every newly developed software before automated testing. This testing requires great efforts and time, but it gives the surety of bug-free software. Manual Testing requires knowledge of manual testing techniques but not of any automated testing tool.

How to perform Manual Testing:

⇒ First, test engineer observes all documents related to software, to select testing areas.
⇒ Tester analyses requirement documents to cover all requirements stated by the customer.
⇒ Tester develops the test cases according to the requirement document.

⇒ All test cases are executed manually by using Black box testing and white box testing.
⇒ If bugs occurred then the testing team informs the development team.
⇒ The Development team fixes bugs and handed software to the testing team for a retest.



In order to perform manual testing, what skills are required?

The following are the important manual testing skills to acquire:

- Detail-oriented and able to report test results in a professional manner.
- A strong analytical ability.
- Ability to perform technical testing.
- Familiarity with Agile methodologies.
- Plan and track the testing process.
- Knowledge of SDLC, STLC, SQL, and manual concepts.
- An understanding of test management tools, test tracking tools, and testing techniques.

Advantages of Manual Testing:

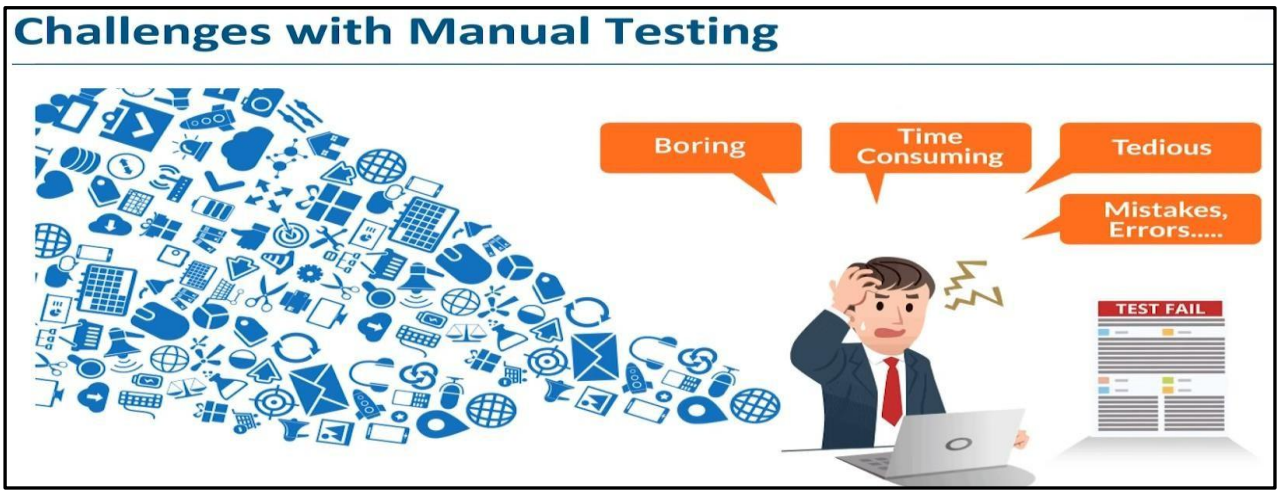
⇒ It does not require programming knowledge while using the Black box method.
⇒ It is used to test dynamically changing GUI designs.
⇒ Tester interacts with software as a real user so that they are able to discover usability and user interface issues.
⇒ It ensures that the software is a hundred percent bug-free.
⇒ It is cost-effective.
⇒ Easy to learn for new testers.

Disadvantages of Manual Testing:

⇒ It requires a large number of human resources.
⇒ It is very time-consuming.
⇒ Tester develops test cases based on their skills and experience. There is no evidence that they have covered all functions or not.

⇒ Test cases cannot be used again. Need to develop separate test cases for each new software.

⇒ Since two teams work together, sometimes it is difficult to understand each other's motives, it can mislead the process.



Manual Test Engineer:

The manual test engineer is a professional who conducts quality checks on software applications without using automation tools or scripting. In essence, the speciality involves manually checking software for errors and fixing them. Manual testers must have the appropriate skills and be able to meet the company's requirements.

2. Automation Testing:

The most significant part of Software testing is Automation testing. It uses specific tools to automate manual design test cases without any human interference. It is used to re-run the test scenarios, which were executed manually, quickly, and repeatedly. It is the most acceptable way to enhance the efficiency, productivity, and test coverage of Software testing. We cannot write the test script or perform the automation testing without understanding the programming language. In automation testing, the test automation engineer will write the test script or use the automation testing tools to execute the application. Some organizations still perform only manual testing to test the application as those companies are not fully aware of the automation testing process.

Advantages of Automation Testing:

⇒ Reusability
⇒ Consistency
⇒ Running tests anytime (24/7)
⇒ Early Bug detection
⇒ Less Human Resources
⇒ Automation testing takes less time than manual testing.
⇒ Automation Testing provides re-usability of test cases on testing of different versions of the same software.
⇒ It does not require many human resources, instead of writing test cases and testing them manually, they need an automation testing engineer to run them.
⇒ The cost of automation testing is less than manual testing because it requires a few human resources.

Disadvantages of Automation Testing:

⇒ Automation Testing requires high-level skilled testers.
⇒ It requires high-quality testing tools.
⇒ When it encounters an unsuccessful test case, the analysis of the whole event is complicated.
⇒ Test maintenance is expensive because high fee license testing equipment is necessary.

There are **3** methods that can be used for software testing:

1. White-Box Testing 2. Black-Box Testing 3. Gray Box Testing

1. White Box Testing:

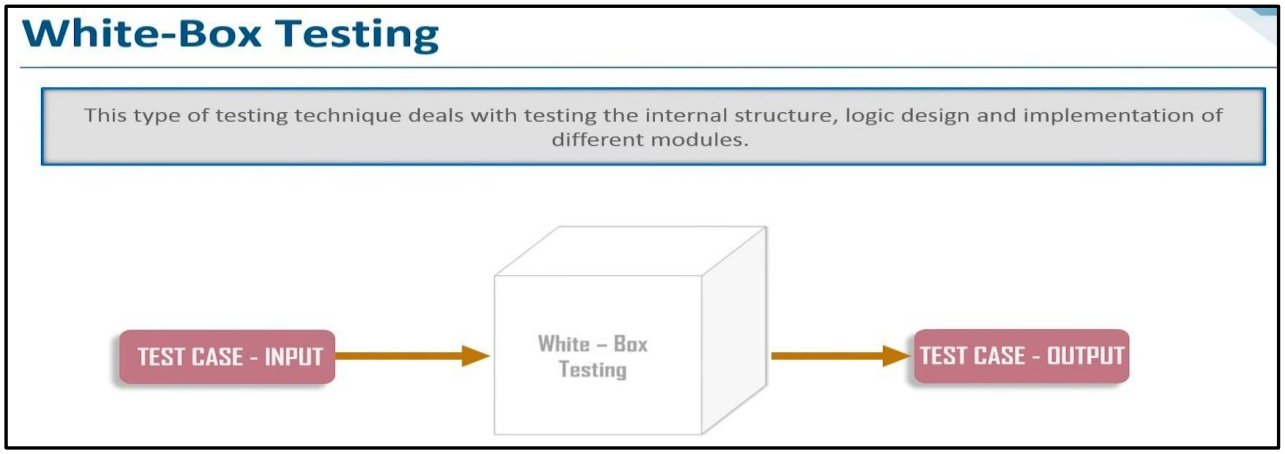
The White Box Testing is a testing technique in which the internal structure of the software is tested including the internal structure, design and coding are tested to verify input-output flow and improve design, usability, and security. Code is visible to testers, so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing, and Glass box testing. In other words, Testing conducted on the source code by developers to check if the source code is working as per the expectation or not is called white box testing.

Need of white box Testing:

⇒ As the source code is visible finding the problems and rectifying the problems is so easy for developers.
⇒ The defects that are identified in white box testing are economical to resolve.
⇒ White box testing is helpful to reduce the defects as early as possible.

⇒ It is used to ensure 100% code coverage.

Note: White box testing is also called as glass box, structural testing, and clear box testing.



Different types of white-box testing:

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

2. Black box Testing:

It is a Software Testing method that analyses the functionality of a software/application without knowing much about the internal structure/design of the item that is being tested and compares the input value with the output value. The main focus of Black Box Testing is on the functionality of the system as a whole. The term 'Behavioural Testing' is also used for Black Box Testing. Testing is conducted on the application by test engineers or by domain experts to check whether the application is working according to the customer requirements or not. In this method, tester selects a function and gives input value to examine its functionality, and checks whether the function is giving expected output or not. If the function produces correct output, then it is passed in testing, otherwise failed. The primary source of black box testing is a specification of requirements that is stated by the customer.

Need of Black Box Testing:

⇒ White box testing conducted by developer in a technical perception where as black box testing is conducted by test engineers with an end-user perception.

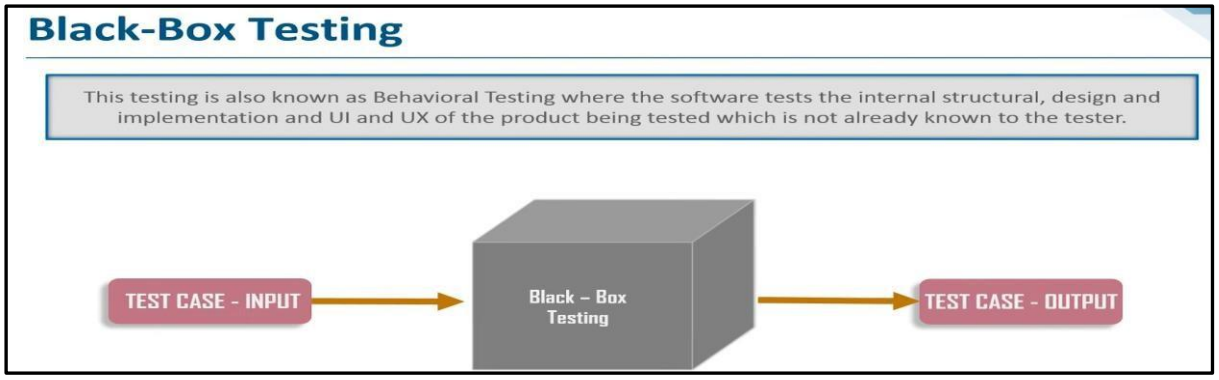
⇒ Programmers will conduct white box testing in a positive perception whereas tester will conduct black box testing with a negative perception. This will provide a chance of finding more defects

⇒ The more defects you identify, you will achieve more results in a quality system.

⇒ White box testing will not cover non-functional areas as non-functional requirements. It is very important for live production and is covered under black box testing.

⇒ The objective of white box testing is 100% coverage whereas the objective of black box testing is 100% customer business requirement coverage.

⇒ Black Box Testing = System Testing + User Accepting Testing. It is called a Requirement Based Testing (or) Specification Based Testing.

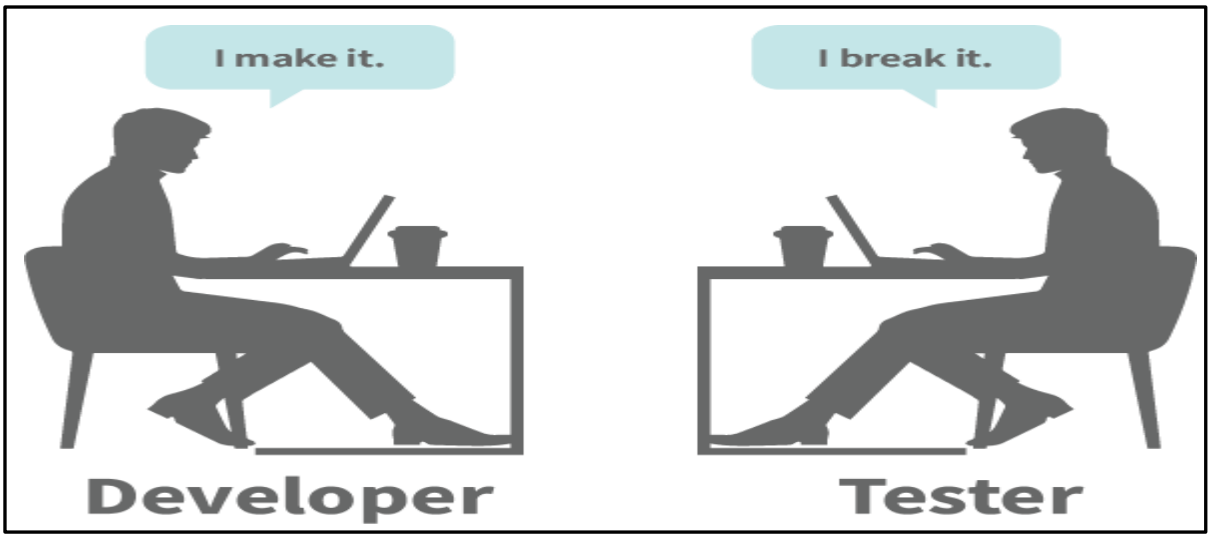


Software Developer and Test Engineer:

Software Developer: Software developers create computer programs and solve technical problems using their creativity, analytical thinking and problem-solving skills. They also develop computer games and other digital architecture. Using their skills, they create internal programmers that help a business be more efficient or create software which their company can sell in the open market. A software developer writes code to build online structures, environments and applications. As a result, mastery over one or more programming language is necessary for this job role. They perform the following tasks:

- Develop and test software based on client's specifications
- Upgrade existing software
- Document their work for future references
- Perform quality assurance on applications they develop or upgrade

Software Test Engineer: A software tester is an individual that tests software for bugs, errors, defects or any problem that can affect the performance of computer software or an application. Software testers are part of a software development team and perform functional and non-functional testing of software using manual and automated software testing techniques. A software tester primarily performs software quality testing procedures on software. They generally have strong grasp over software quality testing tools and techniques, along with some level of software development knowledge/experience. The software tester ensures that the software performs as expected both functionally and non-functionally.



Software Developer vs Test Engineer:

Developer	Test Engineer
Software developers write and maintain source code for computer programming.	A software tester is responsible for identifying the quality, correctness, and completeness of software.
Basically, it involves developing software through successive phases in an orderly manner.	This test evaluates how well a software application works.
They solve problems quickly, reduce costs, increase flexibility, and improve the quality of business.	By reporting problems as soon as possible, they help save money, provide security, and ensure the quality of the software.
Additionally, they provide suggestions on how to improve software applications.	Not only do they find bugs, but they also identify their root causes, so bugs can be permanently fixed.
Coding skills, time management skills, and programming skills are all essential for developers.	An ideal tester should be well versed in the system being developed, possess good communication skills, be a critical thinker, etc.
As part of the software development process, they mostly focus on the requirements of the users.	In testing software applications, they are primarily concerned with the behavior of the end-user.
Its responsibility is to create individual software applications.	Its responsibility is to evaluate individual software programs.

Types of Black Box Testing:

The following are the two categories of black box testing:

1. **Functional Testing**
2. **Non-Functional Testing**

1. **Functional Testing:**

Functional testing is a kind of black-box testing that is performed to confirm that the functionality of an application or system is behaving as expected. This testing type deals with the functional requirements or specifications of an application. Here, different actions or functions of the system are being tested by providing the input and comparing the actual output with the expected output.

For example, when we test a Dropdown list, we click on it and verify if it expands and all the expected values are showing in the list.

Types of Functional Testing:

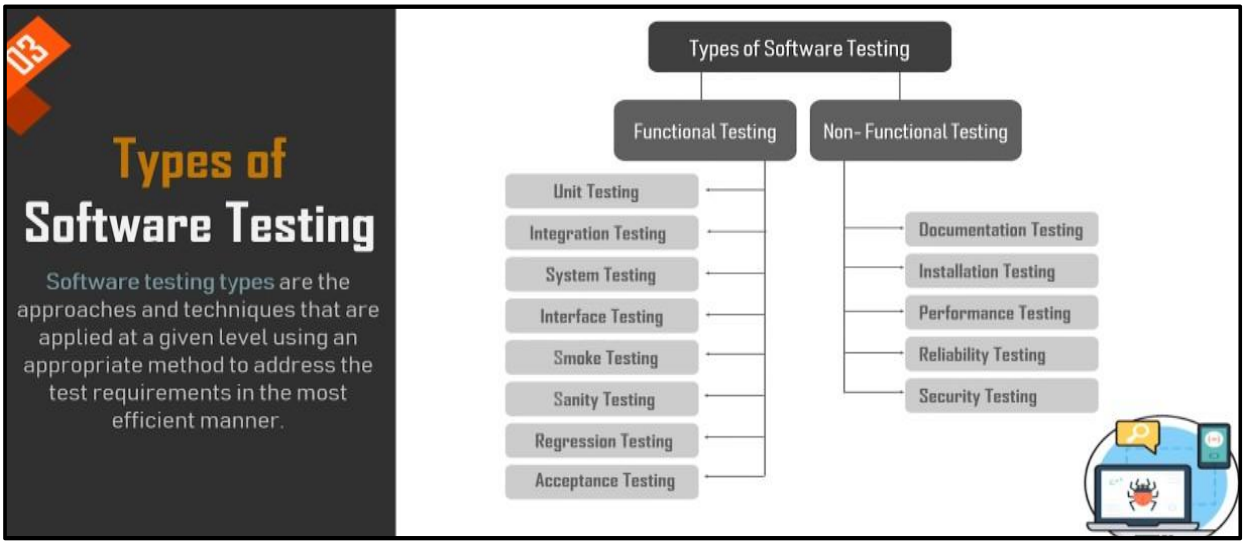
1. Smoke Testing
2. Sanity Testing
3. Integration Testing
4. System Testing
5. Regression Testing
6. User Acceptance Testing

2. Non-Functional Testing:

It focuses on the software's performance, usability, and scalability. In Non-Functional Testing to be tested to improve the quality and performance of the application.

Types of Non-Functional Testing:

1. Usability Testing
2. Load Testing
3. Performance Testing
4. Compatibility Testing
5. Stress Testing
6. Scalability Testing



Difference Between Functional and Non-Functional Testing:

Functional Testing	Non-Functional Testing
It tests 'What' the product does. It checks the operations and actions of an application.	It checks the behavior of an application.
Functional testing is done based on the business requirement.	Non- functional testing is done based on the customer expectation and Performance requirement.
It tests whether the actual result is working according to the expected result.	It checks the response time, and speed of the software under specific conditions.
It is carried out manually. Example: Black box testing method.	It is more feasible to test using automated tools. Example: Load runner.
Customer feedback helps in reducing the risk factors of the product.	Customer feedback is more valuable for non- functional testing as it helps to improve and lets the tester to know the expectation of the customer.
It tests as per the customer requirements.	It tests as per customer expectations.
It is testing the functionality of the software.	It is testing the performance of the functionality of the software.
Functional testing has the following types: ⇒ Unit testing ⇒ Integration testing ⇒ System Testing ⇒ Acceptance Testing	Non-functional testing includes: ⇒ Performance testing ⇒ Load Testing ⇒ Stress testing ⇒ Volume testing ⇒ Security testing ⇒ Installation testing ⇒ Recovery testing

Difference Between White Box Testing and Black Box Testing:

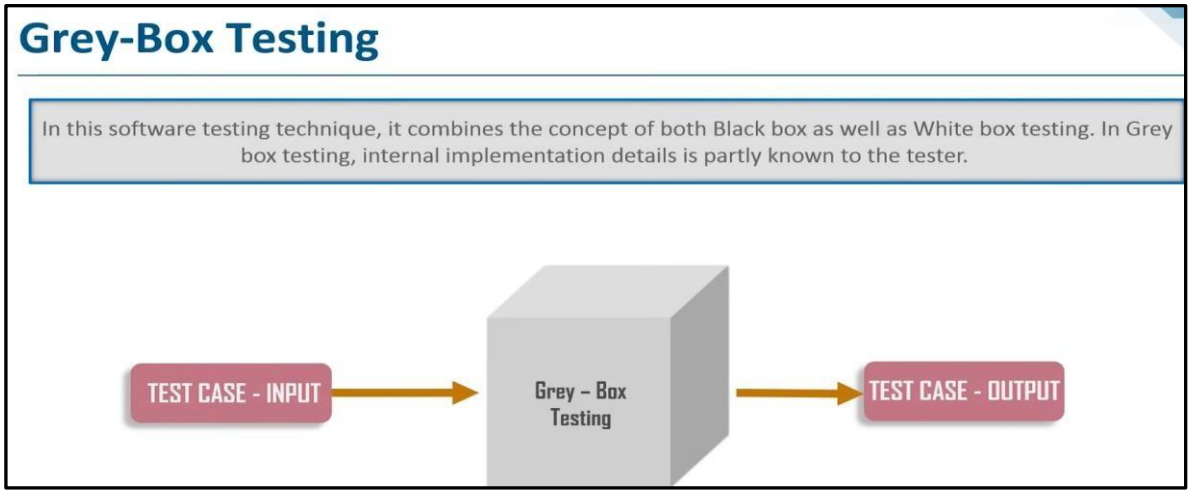
Black Box Testing	White Box Testing
It is a testing method without having knowledge about the actual code or internal structure of the application.	It is a testing method having knowledge about the actual code and internal structure of the application.
The main objective of this testing is to test the Functionality / Behavior of the application.	The main objective is to test the infrastructure of the application.
Some test techniques include Boundary Value Analysis, Equivalence Partitioning, Error Guessing etc.	Some testing techniques include Conditional Testing, Data Flow Testing, Loop Testing etc.
This is a higher-level testing such as functional testing.	This type of testing is performed at a lower level of testing such as Unit Testing, Integration Testing.

It concentrates on the functionality of the system under test.	It concentrates on the actual code – program and its syntax's.
Black box testing requires Requirement specification to test.	White Box testing requires Design documents with data flow diagrams, flowcharts etc.
Black box testing is done by the testers.	White box testing is done by Developers or testers with programming knowledge.
This is performed by professional Software Testers.	This is the responsibility of the Software Developers.
Main focus of the tester is on how the application is working.	Main focus will be on how the application is built.

3. Gray Box Testing:

Gray Box Testing is a software testing technique which is a combination of Black Box Testing technique and White Box Testing technique. In Black Box Testing technique, tester is unknown to the internal structure of the item being tested and in White Box Testing the internal structure is known to tester. The internal structure is partially known in Gray Box Testing. This includes access to internal data structures and algorithms for purpose of designing the test cases. It is a combination of black box and white box testing. Gray-box testing is beneficial because it takes the straightforward technique of black-box testing and combines it with the code-targeted systems in white-box testing.

Regression Testing: To check whether a change in the previous version of the tested application introduced any new error.



Verification and Validation:

Verification and Validation is the process of investigating that a software system satisfies specifications and standards and it fulfils the required purpose.

Verification:

"Are we building the system, right? It's a process of checking if the system is well- engineered. It is also called static testing".

Verification is the process of checking that a software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfils the requirements that we have. Verification is Static Testing. It is a static analysis technique. Here, testing is done without executing the code. Examples include – Reviews, Inspection, and walkthrough.

Activities involved in Verification:

1. Inspections
2. Reviews
3. Walkthroughs
4. Desk-checking

Verification Strategies:

1. Requirements Review: The study and discussions of the computer system requirements to ensure they meet stated user needs and are feasible.

Deliverable: Reviewed statement of requirements (Approved SRS)

2. Design Review: The study and discussion of the computer system design to ensure it will support the system requirements.

Deliverable: Approved High level design (HLD) and Low-level Design (LLD) includes DB, UI and UML diagrams.

3. Code Walkthrough: Informal analysis of the program source code to find defects and verify coding techniques.

Deliverable: Software ready for initial testing by the developer.

4. Code Review: Formal analysis of the program source code to find defects as defined by meeting system design specification

Deliverable: Software ready for testing by the testing team.

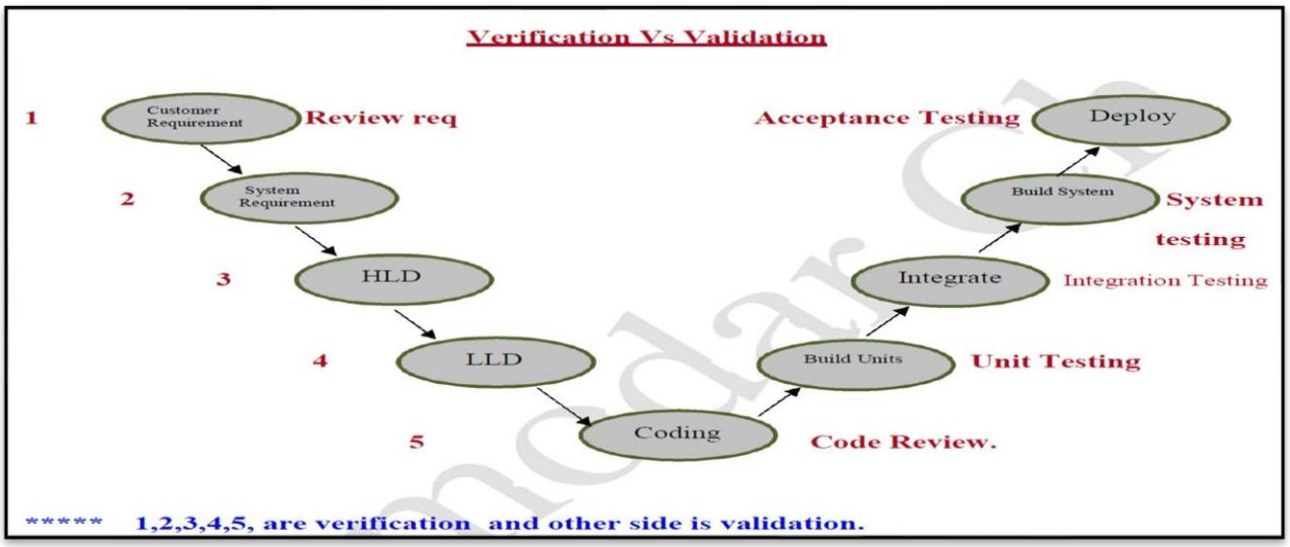
Validation:

"Are we building the right system? It is a process of checking if system meets the customer's actual requirements. It is also called dynamic testing".

Validation is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product i.e., it checks what we are developing is the right product. it is validation of actual and expected product. Validation is the Dynamic Testing. It is a dynamic analysis technique where testing is done by executing the code. Examples include functional and non-functional testing techniques.

Activities involved in validation:

1. Black box Testing
2. White box Testing
3. Unit Testing
4. Integration Testing



Differences between Verification and Validation:

Verification	Validation
We check whether we are developing the right product or not.	We check whether the developed product is right.
Verification is also known as static testing.	Validation is also known as dynamic testing.
Verification includes different methods like Inspections, Reviews, and Walkthroughs.	Validation includes testing like functional testing, system testing, integration, and User acceptance testing.
It is a process of checking the work-products (not the final product) of a development cycle to decide whether the product meets the specified requirements.	It is a process of checking the software during or at the end of the development cycle to decide whether the software follow the specified business requirements.
Quality assurance comes under verification testing.	Quality control comes under validation testing.
The execution of code does not happen in the verification testing.	In validation testing, the execution of code happens.

In verification testing, we can find the bugs early in the development phase of the product.	In the validation testing, we can find those bugs, which are not caught in the verification process.
Verification testing is executed by the Quality assurance team to make sure that the product is developed according to customers' requirements.	Validation testing is executed by the testing team to test the application.
In this type of testing, we can verify that the inputs follow the outputs or not.	In this type of testing, we can validate that the user accepts the product or not.
Verification is done before the validation testing.	After verification testing, validation testing takes place.

Testing Roles and Responsibilities:

Software Testing Roles:

1. Software Test Engineer [STE]
2. Senior Software Test Engineer [SSTE]
3. Automation Test Engineer
4. Test Lead [TL]
5. Test Manager [TM]

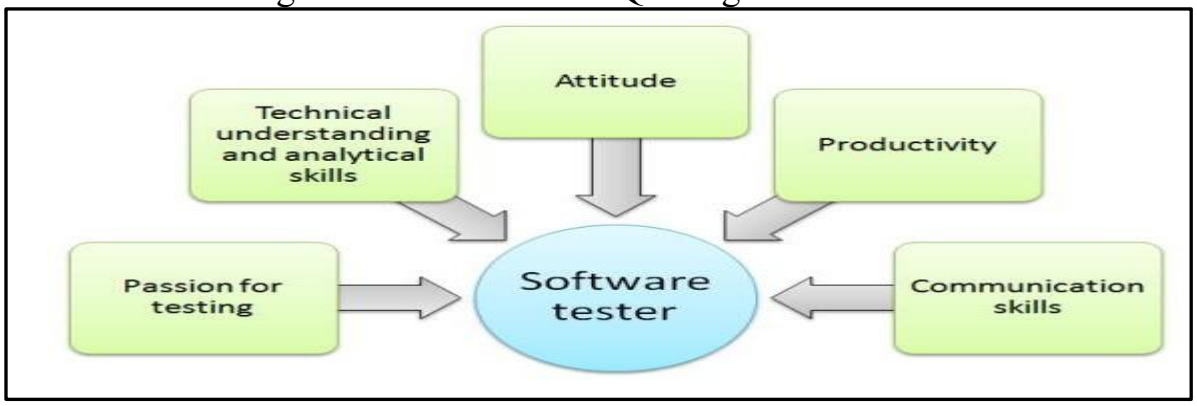
Note: In some organization's role is also called as Designation.

Software Test Engineer [STE]:

A software tester's primary responsibility is to perform manual testing on software applications to ensure the product quality fits everyone's expectations. Testers will raise defects when they find errors in the application and when the implementation derives from the specified product requirements.

Software tester activities may vary significantly depending on the project and test team size. While in large QA teams, test leads can distribute the work so that testers only need to focus on test execution and raise defects. A software testing role may include more activities such as test creation and test planning and test status reporting in small QA teams.

Note: Software Test Engineer is also called as QA Engineer.



Software Test Engineer Responsibilities:

⇒ Understanding the requirements and functional specifications of the application.
⇒ Identifying the required test scenarios for the project.
⇒ Designing and preparing Test cases to validate the application.
⇒ Execute Test cases to valid the application.
⇒ Log Test results (How many test cases pass or fail).
⇒ Defect reporting and tracking.
⇒ Retest fixed defects of previous build.
⇒ Performed various type of testing assigned by Test Lead (Functionality, Usability, User Interface and compatibility) etc.,
⇒ Reports to Test Lead about status of the assigned tasks.
- ❖ Daily status report
- ❖ Daily defect report
- ❖ Weekly status report
- ❖ Retesting Report
- ❖ Other assigned tasks if any

⇒ Participated in regular team meetings by test lead and test manager.
⇒ Creating automation scripts for regression testing.
⇒ Provides enhancements to project-based on End user prospective.
⇒ Provides recommendation on whether or not the application is ready for production.

Senior Software Test Engineer [SSTE]:

Senior software test engineers are responsible for ensuring that the products their company produces are of the highest quality possible. Senior software test engineers may also be tasked with developing new testing methods or strategies as the company they work for grows and changes over time.

Senior Software Test Engineer Responsibilities:

- ➢ Same as test engineer responsibilities.

+

- ➢ He is participated in review of test scenarios, test cases and defects.

- Sometimes he is involved in preparation of test plan also.
- Whenever Test lead on vacation, he will lead the team (Backup for Test lead).

Test Lead [TL]:

A test lead's primary responsibility is to plan and coordinate the activities for a team of testers. The test lead may also collaborate in team activities and test execution.

Test lead activities include mentoring testers and managing testing requirements. The test lead is often responsible for configuring and preparing QA environments for testing.

Test Lead Responsibilities:

- ✓ Task Preparation.
- ✓ Task allocation to team members.
- ✓ Training team members.
- ✓ Team management.
- ✓ Bug/Defect reviews.
- ✓ Test scenarios and Test case development for new enhancements.
- ✓ Test scenarios and Test case reviews.
- ✓ Preparation of Build summary report
- ✓ Conducting meeting within team members.
- ✓ MOM preparation and conduction.
- ✓ Coordinates overall test team activities.
- ✓ Adding test deliverables to configuration repository.
- ✓ Test plan preparation.

Test Manager [TM]:

A test manager's primary responsibility is to plan and coordinate the team of testers and test leads. The role of the test manager and test lead is often very similar. In a broad sense, a test manager can accumulate more responsibility than a test lead. A test manager may have more than one QA team to manage, each headed by a test lead.

Test manager activities include mentoring testers and test leads. Activities also include managing testing requirements and choosing the test management tools to be used.

Test Manager Responsibilities:

- ⇒ Project and Test plan preparation.
- ⇒ Effort estimates.
- ⇒ Project management.
- ⇒ Training plan- Identify training needs and resources, skill gapes and initiate training [Internal and External].
- ⇒ Project Configurations.
- ⇒ Weekly summary report.

⇒ Client communications.
⇒ MOM's.
⇒ Managing support activities.
⇒ Provides regular status updates to core team.
⇒ Schedule meetings with development team and Testing team.

Automation Test Engineer:

An automation tester's primary responsibility is to prepare automated testing on software applications to ensure quality consistency and reduce manual testing efforts. Automation testers automate stable test cases to run regularly to guarantee code changes do not affect stable features with new bugs.

Automation tester activities may vary between using testing frameworks with user-friendly interfaces for test automation, to scripting and programming tests or testing tools for QA processes. It is not uncommon that automation testers participate in manual testing to improve the QA team's overall productivity and gain experience with the test scenarios before creating the automated test case.

Automation Test Engineer Responsibilities:

⇒ Review design and user documentation, project specifications, and requirements.
⇒ Design and execute automated test cases.
⇒ Prepare test environment and test data.
⇒ Track and report all defects from automation runs.
⇒ Provide test execution statistics.
⇒ Provide test summary reports.

Sno	Name	Role	Responsibilities
1	Krishna N	Test Lead	Test Planning, guidance, Monitoring and Test control
2	Venkat Rao P	Sr. Tester	Test Data Collection, Generating Test Scenarios.
3	Rakesh K	Tester	Test Case Documentation, Test execution, defect reporting and tracking for Admin module.
4	Althaf SK	Tester	Test Case Documentation, Test execution, defect reporting and tracking for Personal Banking module.
5	Kiran M	Tester	Test Case Documentation, Test execution, defect reporting and tracking for Corporate Banking module.

Real Time Project Execution-Expected Project Team:

⇒ Client and CEO (IT Company)
⇒ Vice President (VP)-Delivery Manager
⇒ Project Manager -Project Execution
⇒ Project Lead
⇒ Business Team
⇒ Technical Architects -Design HLD, LLD
⇒ Development Lead
⇒ Developers -Coding
⇒ Test Manager -Test Management
⇒ Test Lead
⇒ Test Engineers -Validation
⇒ SCM (Software Configuration Management Team) -Project Repository
⇒ SQA (Software Quality Assurance Team) -Process and Audits
⇒ CCB (Change Control Board) -Change Management
⇒ Technical Writers -Project Documentation and Help Guide

Software Company and Departments:

⇒ HR Development (Recruitments, Training and Relieving)
⇒ Finance Department (Invoices, Billing, Payroll and Tax)
⇒ NetOps (Network Operations) (System Admin)
⇒ Admin (Operations and Office Maintenance)
⇒ Engineering Department (Design and Development)
⇒ Quality Assurance Department (SQA and SQC-Testing)
⇒ Sales and Marketing (Proposals and SOW)

Testing Terminology:

- ✓ PMP: Project Management Plan
- ✓ BRS: Business Requirement Specification
- ✓ SRS: System Requirement Specification
- ✓ FRS: Functional Requirement specification
- ✓ HLD: High Level Design
- ✓ LLD: Low Level Design
- ✓ LOC: Lines of Code
- ✓ RTM: Requirement Traceability Matrix
- ✓ SSTE: Senior Software Test Engineer
- ✓ ATE: Automation Test Engineer
- ✓ TL: Test Lead
- ✓ TP: Test Plan

- ✓ TP: Test Plan
- ✓ TS: Test Suite
- ✓ TS: Test Scenario
- ✓ TC: Test Case
- ✓ CR: Change Request
- ✓ MOM: Minutes of Meeting
- ✓ DSR: Daily Status Report
- ✓ DDR: Daily Defect Report
- ✓ WSR: Weekly Status Report
- ✓ DRT: Daily Re-testing Report
- ✓ STE: Software Test Engineer
- ✓ TM: Test Manager
- ✓ PM: Project Manager

- ✓ BA: Business Analyst
- ✓ CTO: Chief Technology Officer
- ✓ SCMP: Software Configuration Management Plan
- ✓ SEPG: Software Engineering Process Group
- ✓ SQA: Software Quality Assurance
- ✓ SQC: Software Quality Control
- ✓ NC: Non-Conformance
- ✓ E2E: End to End
- ✓ OID: Organizational Innovation and Deployment
- ✓ ECP: Equivalence Class Partitioning
- ✓ BVA: Boundary Value Analysis
- ✓ VSS: Visual Source Safe
- ✓ QMS: Quality Management System
- ✓ QPM: Quality Project Management
- ✓ QIP: Quality Improvement Plan
- ✓ CAR: Casual Analysis and Resolution
- ✓ DAR: Decision Analysis and Resolution
- ✓ KT: Knowledge Transfer
- ✓ KPA: Key Process Area
- ✓ KRA: Key Result Area
- ✓ CCB: Change Control Board
- ✓ SOW: Statement of Work
- ✓ ATM: Appraisal Team Members
- ✓ ISTQB: International Software Testing Qualifications Board
- ✓ IEEE 829: Electrical and Electronics Engineers

- ✓ White-box Testing
- ✓ Black-box Testing
- ✓ Unit Testing
- ✓ Functional Testing
- ✓ Exhaustive Testing
- ✓ Expected result
- ✓ Formal review
- ✓ Informal review
- ✓ Gray-box testing
- ✓ Load testing
- ✓ Installation test
- ✓ Integration testing
- ✓ Module testing
- ✓ Naming standard
- ✓ Outcome
- ✓ Debugging
- ✓ Defect Report
- ✓ Severity
- ✓ Professional tester
- ✓ Regression testing
- ✓ Release
- ✓ Re-testing
- ✓ Retrospective meeting
- ✓ End-to-End[E2E] Testing
- ✓ Regression Testing
- ✓ Entry Criteria
- ✓ Exploratory testing
- ✓ Smoke testing
- ✓ Functional testing
- ✓ Positive testing
- ✓ Negative testing
- ✓ Non-functional testing
- ✓ Static testing
- ✓ Stress testing
- ✓ Performance testing
- ✓ Postconditions
- ✓ Preconditions
- ✓ Quality
- ✓ Review
- ✓ Risk
- ✓ Scrum
- ✓ Test automation
- ✓ Test environment
- ✓ Exit Criteria
- ✓ Error
- ✓ Failure
- ✓ Structural testing
- ✓ Security testing
- ✓ Retesting
- ✓ Selenium
- ✓ Test data
- ✓ Open source
- ✓ Prerequisites
- ✓ Priority
- ✓ Record and playback tool
- ✓ Test execution
- ✓ Test level
- ✓ Test log
- ✓ Test object
- ✓ Test report

- ✓ Test script
- ✓ Test strategy
- ✓ Validation
- ✓ Verification
- ✓ Versioning

Software Configuration Management:

⇒ The process of identifying, organizing and controlling changes to the software during the development and maintenance phase.

⇒ A methodology to control and manage a software development project.

Purpose of SCM:

Establish and maintain the integrity of work products.

Here, we have PMP, BRS, SRS, FRS, HLD, LLD, TP, TS, TC, LOC, MOM, Reports etc… These all are called Project deliverables or Project artifacts or Work products or Work items.

SCM Needs:

⇒ Multiple people have to work on software that is changing.

⇒ Projects delivering several releases (Builds)

⇒ Software must run on different machines and operating systems (Rapid evolution of software and hardware).

Problems resulting from poor Configuration Management:

⇒ Can't roll back to previous subsystem

⇒ One changed overwrites another

⇒ Which code change belongs to which version.

⇒ Faults which were fixed Re-appear.

⇒ Tests worked perfectly on old versions.

Configuration Management Using:

- Multiple project members – concurrent access
- Orderly control of change
- View difference
- Rollback changes
- Release Management
- History
- One of the key process areas in SEI CMMI Level2.

How it is accomplished SCM:

All these goals are accomplished through "Version Control"

Version Control:

⇒ Commonly used in engineering and software development.

⇒ Changes identified by incrementing an associated number.

Version Control done in Two ways:

1. Manually

2. Automation

1. Manually: In company server, administrator creates the project folder and he can provide the access permissions to all users [Project members].

2. Automation: Doing with the help of automation tools.

Open Source: Free

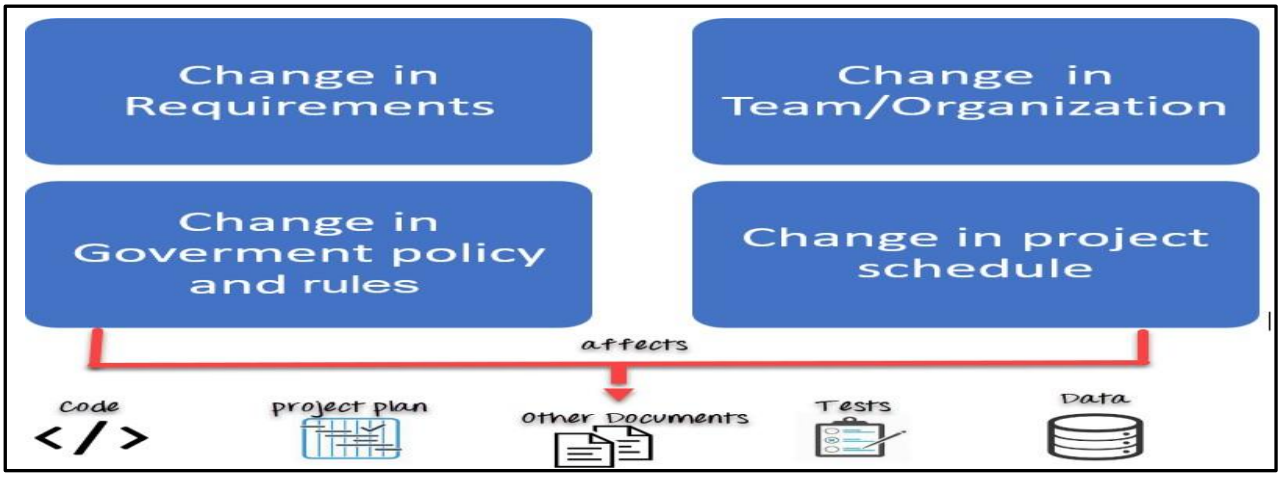
CVS: http://cvshome.org

Major Activities:

⇒ Configuration planning & Step.
⇒ Configuration identification.
⇒ Configuration baseline.
⇒ Change Management.
⇒ Configuration release control.
⇒ Configuration Audit.
⇒ Control of customer property.

Sample list of Software configuration items:

⇒ Management plans (Project Plan, Test Plan, etc.,)
⇒ Specification (Requirements, Design, Test cases, etc.,)
⇒ Customer documentation (Implementation, Manuals, User Manuals, Online help files)
⇒ Source code (Java, .Net, PHP, VB, etc.,)
⇒ Executable code (exe's)
⇒ Libraries (Packages, %includes. Files, API'S, DLL'S, etc.)
⇒ Databases (Data being processed, test data, etc.)
⇒ Production documentation.



Vocabulary in SCM:

1. **Repository:** A server were the files (Project deliverables) are stored.
2. **Check- Out:** Copies a working copy from the repository.
3. **Check – In:** Copy the changes on the local files to the directory (The VSS takes care of changes since the last synchronization).
4. **Undo checks out [uncheck out]:** The "Undo check out" command does just that, the file returns to the status it had before it was checked out. This can be done even if changes have been made to the local copy of course, these do not go into the database.

MS VSS (Visual Source Safe) or Team Foundation Service [TFS]:

It is commercial software from Microsoft. Microsoft Visual SourceSafe is a file-level version

control system that permits many types of organizations to work on several project versions at the same time. This capability is particularly beneficial in a software development environment, where it is used in maintaining parallel code versions. However, the product can also be used to maintain files for any other type of team. Available in two types: i.e.,

1) Server 2) Client

⇒ Test Engineer responsibilities in configuration management i.e., VSS6.0.
⇒ Copy files from VSS i.e., checkout files from repository.
⇒ Check in files after completion of your work.

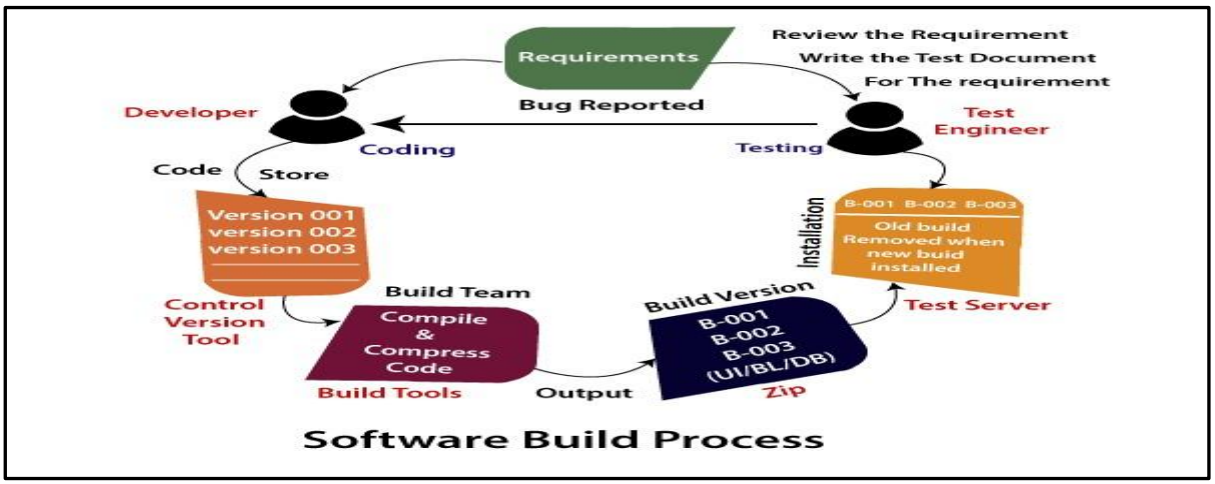
Change Control Board:

Change Control is the process that a company uses to document, identify and authorize changes to an IT environment. It reduces the chances of unauthorized alterations, disruption and errors in the system. Whenever any new or different changes are requested for the system, especially by stakeholders, it is neither optional nor ignorable. It has to be implemented without affecting other components of the system. A change control board is sometimes referred to as a change review board. It's a group of people from the project team that meets regularly to consider changes to the project. A change control board looks at change requests which are then reviewed in detail. The changes differ from the baseline requirements set at the approval of the project plan. Once the change control board has approved a change, the change must be managed.

Build Release process:

Build is generally a software or an application ready for testing. Developers prepare a software and then give to testers for testing. It is a general term which refers to an application which is going to be tested. Developers can prepare a full application or add a new feature to the existing application. Then that application along with new feature is called as BUILD which is tested by testers. Build is something which has a working piece of code need to be tested. One of the most important steps of a software build is the compilation process, where source code files are converted into executable code. The process of building software is usually managed by a build tool. Builds are created when a certain point in development has been reached or the code

has been deemed ready for implementation, either for testing or outright release. The software build is an activity to translate the human-readable source code into an efficient executable program.



Types of Software Build:

1. **Full Build:** A full build which performs a build from scratch. It treats all resources in a project as if they have never been seen by the build server/tool.
2. **Incremental Build:** An incremental build which uses a "last build state," maintained internally by the build server/tool, to do an optimized build based on the changes in the project since the last build. Since incremental builds only rebuild what needs to be, they are usually much faster than full builds and use fewer resources.

Error, Defect, Bug and Failure:

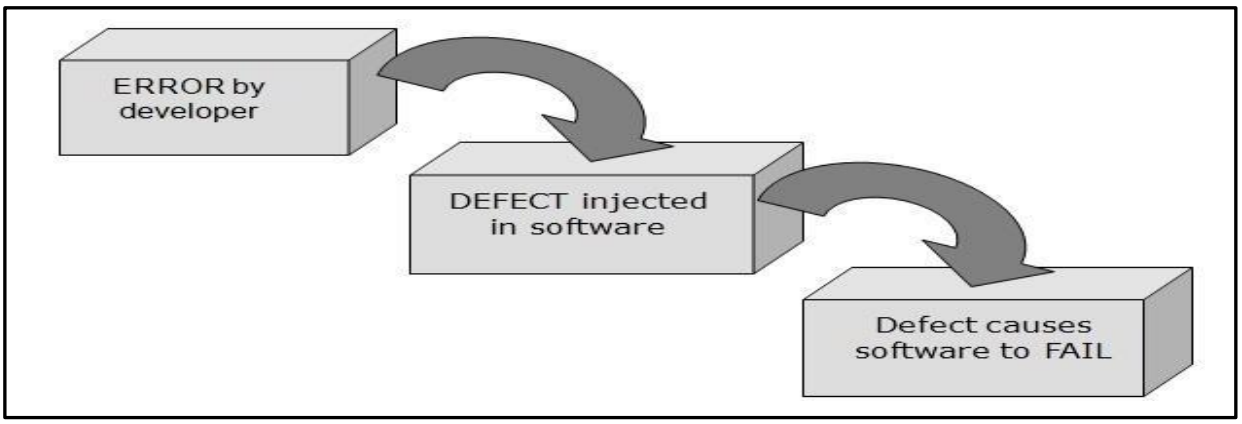
Error: An error is a mistake, misunderstanding, or misconception, on the part of a software developer. The category of developers includes software engineers, programmers, analysts, and testers. For example, a developer may misunderstand a design notation, or a programmer might type a variable name incorrectly – leads to an error. An error normally arises in software, it leads to a change the functionality of the program.

Defect: A defect is a variance between expected results and actual results, detected by the developer after the product goes live. The defect is an error found AFTER the application goes into production. In simple terms, it refers to several troubles with the software products, with its external behavior, or with its internal features. In other words, "Deviation from the expected behavior to the actual behavior of the system is called defect" but not all defects are bugs.

Bug: Developer accepted defect by reporting test engineer is called bug. A bug produces unexpected results or causes a system to behave unexpectedly. In short, it is any behaviour or result that a program or system gets but it was not designed to do". A bug is an unexpected flaw or imperfection that could be permanent.

Failure: The deviation identified by end-user while using the system is called a failure. For example, in a bank application if the Amount Transfer module is not working for end-users

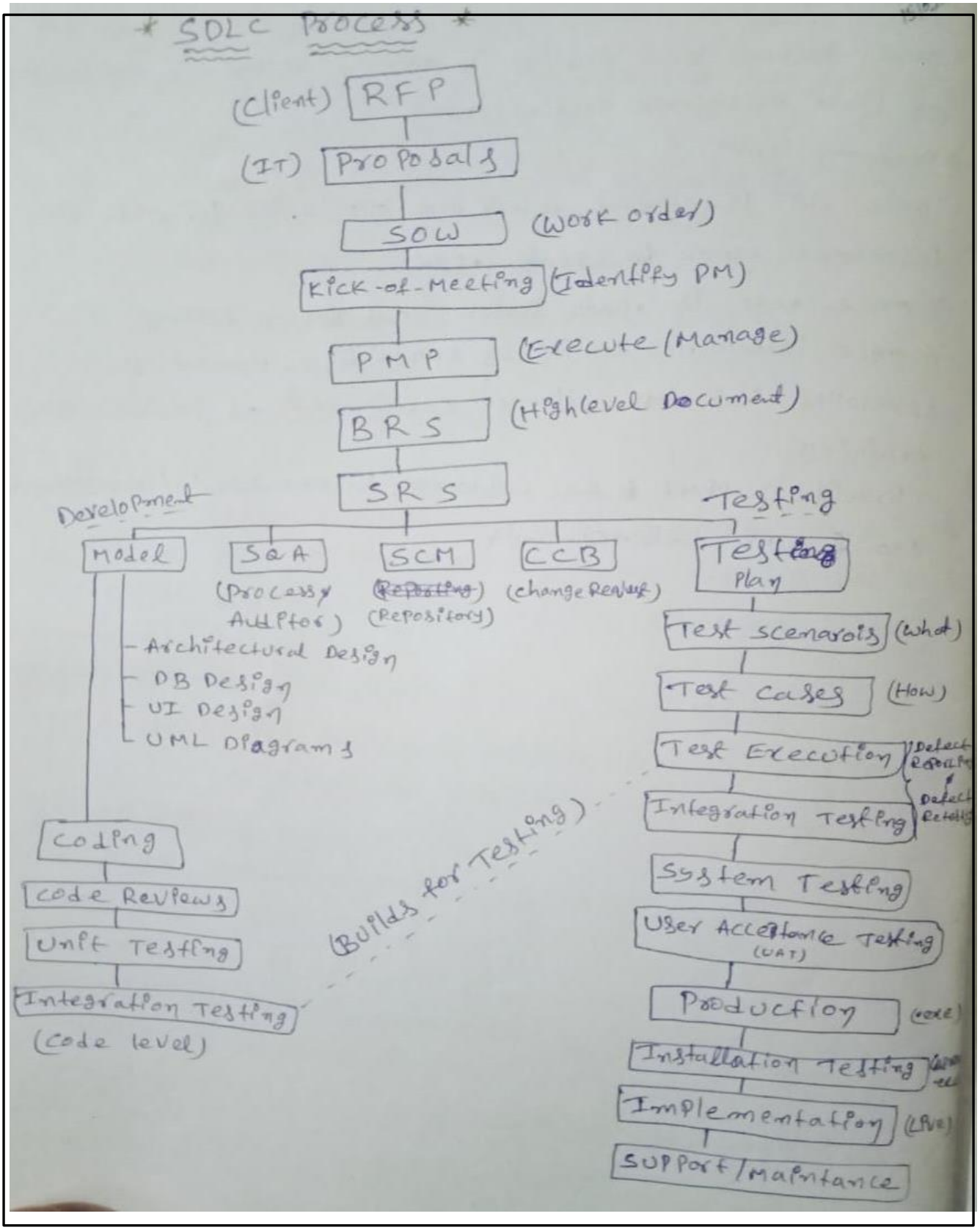
when the end-user tries to transfer money, submit button is not working. Hence, this is a failure.



Software Companies Work Procedure (In Realtime):

Request for Proposal (RFP):

⇒ Flipkart will prepare and distribute the RFP to selected Vendors (Software Companies).
⇒ S/W Companies "Sales and Marketing team" respond the RFP with their PROPOSALS. (About the IT Company, Technical solution, Delivery schedule, Budget, Time line and Deliverables etc...)
⇒ If Flipkart likes proposal from any Vendor (IT Company), then they will sign statement of work (SOW) with that Vendor (work order).
⇒ Conduct Project kick-off meeting (It's a high-level meeting between the senior Managers, CEO, and Project Managers).
⇒ Identify Project Manager.
⇒ Project Manager will prepare Project Management plan (PMP)
⇒ After high level meeting with customer, Business team or technical team will release Business Requirement specification (BRS).
⇒ BRS is a high-level document containing project requirements from customer.
⇒ Person should be a Domain expert / Functional Expert, he is eligible to Convert the BRS into SRS (S/W requirement specification).



QA and Testing:

⇒ Identity standards, Guidelines, procedures, checklists and Templates required to develop the project.

⇒ Review and approval of SRS (Mapping SRS with BRS)- (Business Analyst) BA

⇒ Once SRS is approved, create baseline of SRS (SRS 1.0v).

Development:

Based on approved SRS, software Architect will design model of the project (Blue Print or Prototype)

Model Contains:

1. Architectural Design (Modules, subsystems etc.)
2. Database Design
3. User Interface Design
4. UML Diagrams

- Use case
- Class
- objects
- Sequences etc...

Review and approval of Design Documents - Mapping Design documents with Requirements.

QA and Testing:

⇒ Test Manager/Test Lead Will Prepare Test Plan for project
⇒ Review and approval of Test Plan.
⇒ Test Engineers will identify the Test scenarios for project (what is to be tested)
⇒ Review and approval of Test scenarios.
⇒ Test Engineers will prepare Test cases for all approved test scenarios (How "what is to be tested").
⇒ Review and approval of Test Cases

Development:

⇒ Based on approved Design documents, Developers are going to Convert the logic specified in the design documents into coding of Per chosen programming language and they will release code (Loc).
⇒ Perform code reviews.
⇒ Once code is approved, perform UNIT level Testing.
⇒ Once UNIT level testing is approved, developers will integrate the UNITS and release the BUILDS for testing.

QA and Testing:

⇒ Test Case Execution (perform) Integration level testing)
⇒ Test Results with Pass or Fail.
⇒ Defect Reporting of Failed Test Cases
⇒ Defect status Tracking and Retesting of fixed defects
⇒ Perform Regression testing.
⇒ Once Integration level testing is approved, Perform system level testing.

Development:

⇒ Developers are going to fix the defects which are reported by Testing Department

QA and Testing:

⇒ Once system level testing is approved, release the application for User Acceptance Testing (UAT).
⇒ Perform UAT.
⇒ Once UAT is approved, release the application for production department team to create .exe.
⇒ Once .exe is ready, Perform Installation testing.
⇒ Once installation testing is approved, Implement the developed s/w into customer environment and provide user training.
⇒ Provide support to the customer, for any defects /issues/questions and future Enhancements.

Software Engineering:

S/W engineering is a technological and managerial both displains concerned on a development and maintenance of a s/w projects or products on-time and within cost estimates.

⇒ There are 3 important factors in s/w development is

1. Quality
2. Budget (cost)
3. Schedule (Time)

“Developing a quality s/w within budget and within cycle time".

One of the important characteristics of s/w engineering is "Re-usability".

Advantage of Re-usability:

1. Productivity will be increased.
2. Shorten s/w development time
3. Reduce s/w development and maintenance cost.

Software Process:

⇒ A sequence of steps Performed for a given Purpose.
⇒ A set of Activities, methods and transformations, practices that people use to develop and maintain a system.

If companies are not following process properly -[Dis-advantages]

⇒ Commitments Consistently missed
⇒ Late delivery
⇒ Last minute crunches.
⇒ Quality problems.
⇒ Too much rework.
⇒ Functions do not work correctly.
⇒ Customer complaints after delivery.
⇒ People Frustrated (End-users)

Benefits of Process:

⇒ Quality
⇒ Consistency
⇒ Traceability.
⇒ Early defect removal (verification)
⇒ Reduced re-works and rejections
⇒ Control of cost
⇒ On time

Software Development Life Cycle (SDLC):

Software Development Life Cycle is a systematic approach to develop software. It is a Process followed by Software Developers and Software Testing is an integral part of Software Development, so it is also important for Software Testers. Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test software. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates. ISO/IEC 12207 is an international standard for software life-cycle processes. It defines all the tasks required for developing and maintaining software.

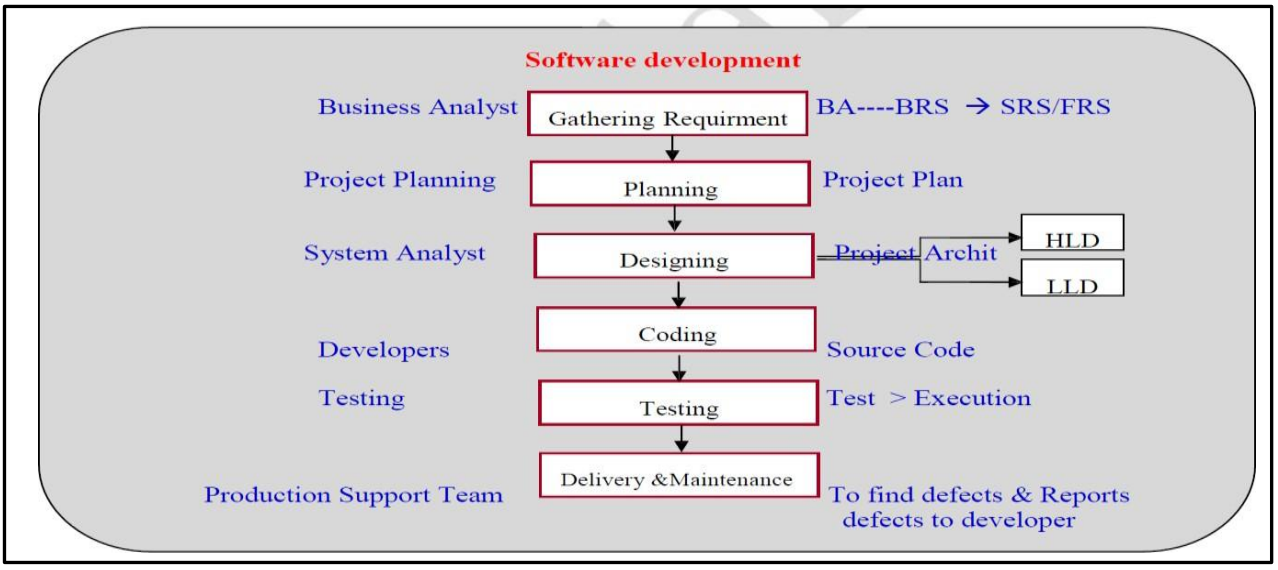
Phases of Software Development Life Cycle:

6 Phases of SDLC

1. **Requirement Gathering**
2. **Analysis**
3. **Design**
4. **Coding / Development**
5. **Testing**

6. **Deployment & Maintenance**

Note: These phases may vary form one organization to another, but purpose is almost all same, that is "Develop and Maintain Quality Software".



1. Requirement Gathering:

⇒ Requirement Gathering is the most important phase in software development life cycle, Business Analyst collects the requirements from the Customer/Client as per the client's business needs and documents the requirements in the Business Requirement Specification and provides the same to Development Team.
Note: Document name may vary from one Organization to another, some examples are Customer Requirement Specification (CRS), Business Requirement Document (BRD) etc…

⇒ Suppose Our Planned Software is not intended for a single customer and the software product for multiple customers then Business Analyst or Business Team collects Requirements from the Market and also evaluate Other similar products in the Market

⇒ Key Role in this phase is Business Analyst and Outcome of the phase is "Business Requirement Specification"

2. Analysis:

⇒ Once the Requirement Gathering is done the next step is to define and document the product requirements and get them approved by the customer.

⇒ This is done through SRS (Software Requirement Specification) document.

⇒ SRS consists of all the product requirements to be designed and developed during the project life cycle.

⇒ Key people involved in this phase are Project Manager, Business Analyst and Senior members of the Team.

⇒ The outcome of this phase is Software Requirement Specification.

3. Design:

⇒ In Design phase Senior Developers and Architects, they give the architecture of the software product to be developed.

⇒ It has two steps one is HLD (High Level Design) or Global Design and another is LLD (Low Level Design) or Detailed Design.

⇒ High Level Design (HLD) is the overall system design, covers the system architecture and database design. It describes the relation between various modules and functions of the system.

⇒ Low Level Design (LLD) is the detailed system design, covers how each and every feature in the product should work and how every component should work.

⇒ The outcome of this phase is High Level Document and Low-Level Document which works as an input to the next phase Coding.

4. Coding / Development:

⇒ Developers (seniors, juniors and fresher) involved in this phase, this is the phase where we start building the software and start writing the code for the product.

⇒ The outcome of this phase is Source Code Document (SCD) and the developed product.

5. Testing:

⇒ Once the software is complete then it is deployed in the testing environment. The testing team starts testing (either test the software manually or using automated test tools depends on process defined in STLC)

⇒ Testing is done to verify that the entire application works according to the customer requirement.

⇒ During this phase, Testing team may find defects which they communicate to developers, the development team fixes the defect and send back to Testing for a re-test.

⇒ This process continues until the software is Stable, and working according to the business needs of that system.

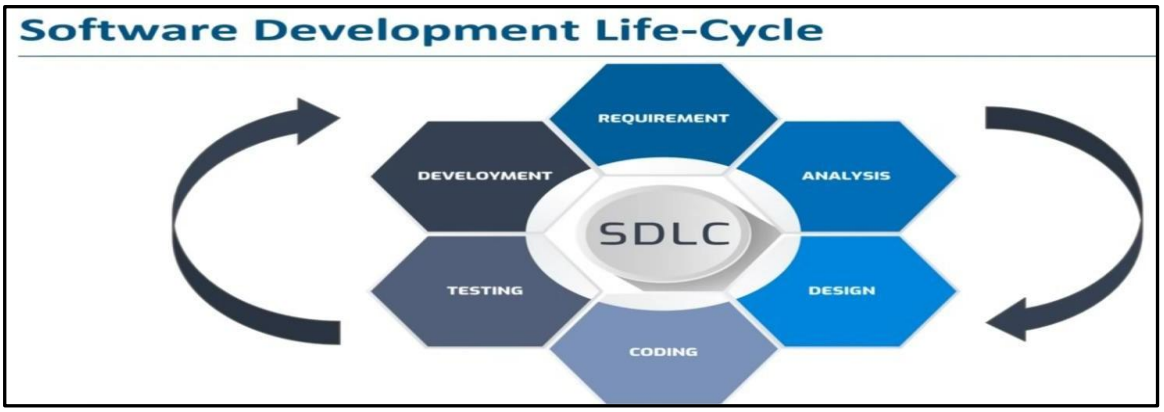
6. Deployment & Maintenance:

⇒ After successful testing, the product is delivered (deployed to the customer for their use), Deployment is done by the Deployment/Implementation engineers and once when the customers start using the developed system then the actual problems will come up and needs to be solved from time to time.

⇒ Fixing the issues found by the customer comes in the maintenance phase.

⇒ 100% testing is not possible – because, the way testers test the product is different from the way customers use the product.

⇒ Maintenance should be done as per SLA (Service Level Agreement)



SDLC Models:

we have various SDLC Models in the IT Industry, Waterfall Model, V Model, Spiral Model and Agile Development Models etc...,

Software Development process varies from one SDLC Model to another.

Software Development Life Cycle Models:

There are various software development life cycle models defined and designed which are followed during the software development process. These models are also referred as Software Development Process Models".

1. **Waterfall Model**
2. **V Model**
3. **Spiral Model**
4. **Agile Model**

1. Waterfall Model:

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases.

In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

Phases of Waterfall Model:

a) **Requirements Gathering:** This first step is also the most important, because it involves gathering information about what the customer needs and defining, in the clearest possible terms, the problem that the product is expected to solve.

b) **Analysis (Software Requirements):** In this phase Business Requirements are converted as Software Requirements.

c) **Design:** In this phase Global and Detailed design can be produced based on Software Requirements.

d) **Coding:** This step consists of actually constructing the product as per the design specification(s) developed in the previous step.

Typically, this step is performed by a development team consisting of programmers, interface designers and other specialists, using tools such as compilers, debuggers, interpreters and media editors.

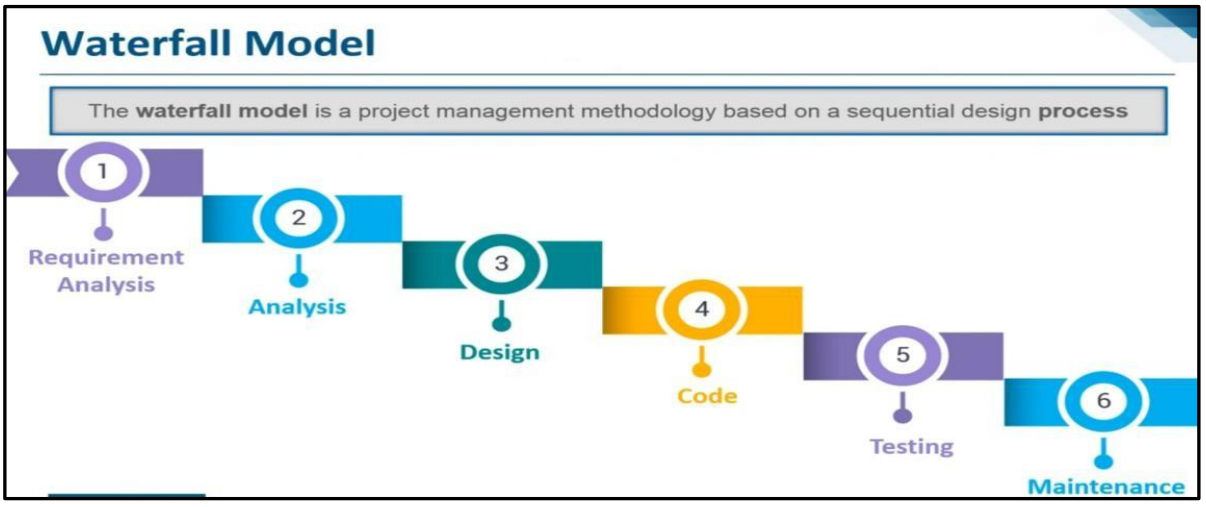
The output of this step is one or more product components, built according to a pre-defined coding standard and debugged, tested and integrated to satisfy the system architecture requirements.

e) Testing: In this stage, System will be tested by testers, if the find any mismatch they report defects.
Developers /Programmers fix the defects and then testers close defects by performing confirmation testing (Regression Testing).

f) Release & Maintenance: Release team (consists of a few developers, testers, and tech-support people etc…) install software in Customer environment and they consider below factors;

Correct & Complete installation, User Management, Services Management, Coexistence with other software, Handling of Input & Output devices, and Handling of secondary storage devices etc…

Maintenance team process Customer issues based on service agreements.



Advantages of Waterfall Model:

a) Simple and easy to use

b) Easy to manage due to the rigidity of the model- each phase has specific deliverables and a review process.

c) Phases are processed and completed one at a time.

d) Works well for smaller projects where requirements are very well understood.

Disadvantages of Waterfall Model:

a) No working software is produced until late during the life cycle.

b) High amount of risk and uncertainty.

c) Poor model for complex and object-oriented projects.

d) Poor model for Long and ongoing projects.

e) Poor Model where requirements are at a moderate to high risk of changing.

2. V Model:

It is Verification & Validation model, known as V Model, in this model all development phases can be integrated with Testing phases.

The V-model illustrates how testing activities can be integrated into each phase of the software development life cycle.

V Model was inaugurated in order to avoid drawbacks in Waterfall model and its main focus on multiple stages of testing.

Multiple stages of Testing avoid defects multiplication.

Development Phases Integration with Testing Phases:

a) User Requirements Vs Acceptance Testing:

Business Analyst category people gather requirements and the document the requirements, after documentation Reviews, Meetings like verification will take place in order get correct & Complete Requirements.

End Users conduct Acceptance Testing using Business / User Requirements.

b) Software Requirements Vs System Testing:

Development Manager/Tech Manager converts User Requirements as Software Requirements and Reviews, Meetings like verification methods will be performed on Software Requirements, after Verification Project manager provides Approval.

Independent testers generate test cases from Software Requirements in order to perform System Testing

c) Global Design Vs Integration Testing:

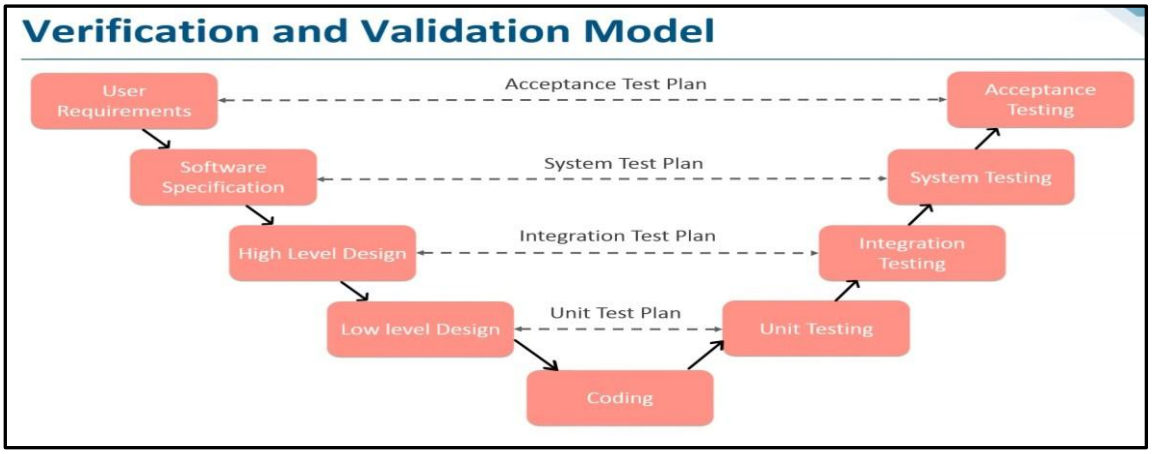
System Architect / senior developer creates Global design, Informal Review/ Walk through /

Technical Review / Inspection like Verification methods will be applied on Design documents.

Developers perform Integration Testing based on Software Global Design.

d) Detailed Design Vs Unit / Component Testing:

Developers perform Unit /Component Testing based on Software Detailed Design.



Advantages of V Model:

a) Tester role will take place in the requirement phase itself.

b) Multiple stages of Testing available so that Defects multiplication can be reduced.

c) Can be used for any type of requirements.

d) Due to Multiple stages of Testing and Multiple teams' involvement Quality can be improved.

e) The V Model Supports wide range of development methodologies such as Structured and Object-oriented systems development.

f) The V Model supports tailoring.

Disadvantages of V Model:

a) It an expensive model than Waterfall model, needs lot of resources, budget and time.

b) Co-ordination and Maintenance are difficult.

c) Adoption of changes in Requirements and Adding New Requirements at middle of the process are difficult.

d) It needs an established process for proper implementation.

3.Spiral Model:

The spiral model is similar to the incremental model, with more importance placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation.

A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spiral, starting in the planning phase, requirements are gathered and risk

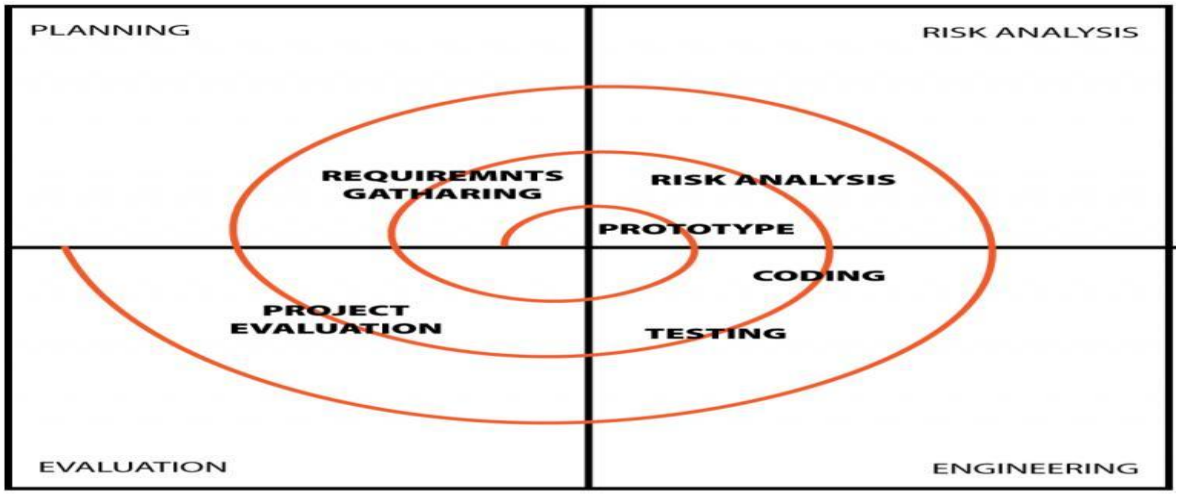
is estimated. It's one of the software development models like Waterfall, Agile, V-Model.

Planning Phase: Requirements are gathered during the planning phase. Requirements like 'BRS' that is 'Business Requirement Specifications' and 'SRS' that is 'System Requirement specifications.

Risk Analysis: In the risk analysis phase, a process is undertaken to identify risk and alternate solutions. A prototype is produced at the end of the risk analysis phase. If any risk is found during the risk analysis, then alternate solutions are suggested and implemented.

Engineering Phase: In this phase software is developed, along with testing at the end of the phase. Hence in this phase the development and testing are done.

Evaluation phase: This phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral.



Advantages of Spiral Model:

⇒ Good for large and mission-critical projects.
⇒ Additional functionality or changes can be done at a later stage
⇒ Cost estimation becomes easy as the prototype building is done in small fragments
⇒ Continuous or repeated development helps in risk management
⇒ Development is fast and features are added in a systematic way in Spiral development
⇒ There is always a space for customer feedback

Disadvantages of Spiral model:

⇒ Can be a costly model to use.
⇒ Risk analysis requires highly specific expertise.
⇒ Project's success is highly dependent on the risk analysis phase.
⇒ Doesn't work well for smaller projects.
⇒ Risk of not meeting the schedule or budget

4. Agile Model (Incremental Model):

⇒ In Agile Model project is divided into various sprints.

⇒ Each sprint contains High-Priority Requirements.
⇒ The time period for sprint is typically 2-4weeks.
⇒ In an Agile model, daily screen meetings with team to share status and potential issues.
⇒ Each sprint is released to customers.
⇒ Used for critical applications.

****

Module 2: Software Testing Life Cycle–(STLC):

BRS Document:

BRS stands for a business requirement specification. A BRS documentation is designed as a collaboration between the business analyst interaction with the customers. It is always developed by Business Analyst. BRS is a high-level document and contains problems from customers' existing system. I n BRS, we define what exactly customer wants. The BRS document includes the business rules, the project's scope, in-detail client's requirements and the scope of the project, and the client's requirements in detail. A BRS includes all the requirements requested by a client. Person should be domain expert or functional expert he will be eligible to convert the BRS to SRS.

SRS Document:

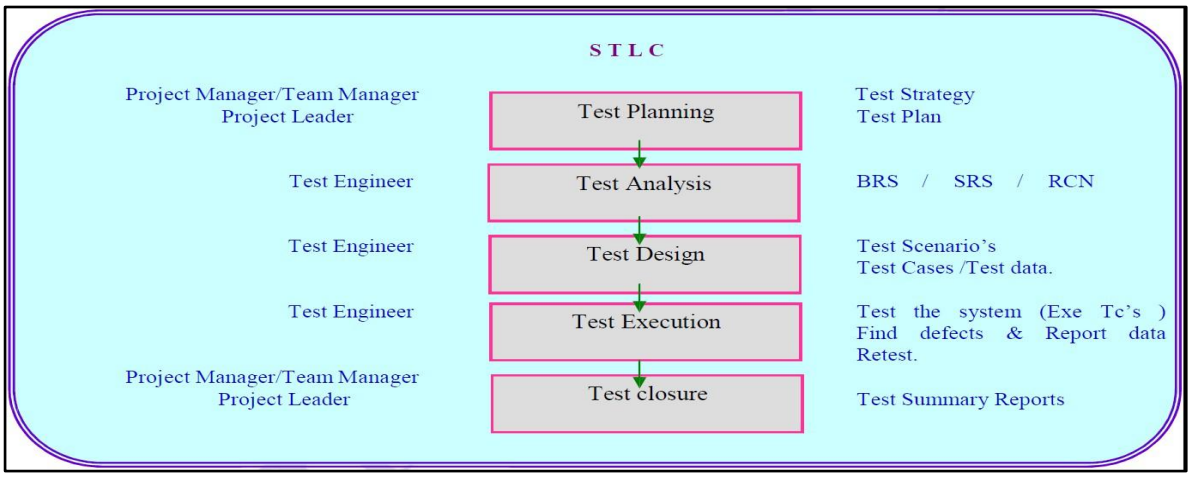
An SRS document is extracted from the Business Requirements Specification Document with the help of the business analysts and the project manager. It is developed by System Analyst. And it is also known as User Requirement Specifications. An SRS includes both functional and non-functional requirements and uses cases as well. SRS document is used to covert the customer information into a detailed document, which can easily understand by the developers and the test engineers. The SRS document is not a project plan. Therefore, this document works as a guide for system architecture and development.

FRS Document:

FRS stands for "Functional Requirement Specification" (FRS). It is always developed by developers and engineers. In FRS, we describe the particular functionalities of every single page in detail from start to end. The FRS document is very descriptive and elaborates on all the functional requirements. The ultimate goal of an FRS is to satisfy all the requirements listed in the SRS and BRS documents. It specifies screens, menus, dialogs and so on....

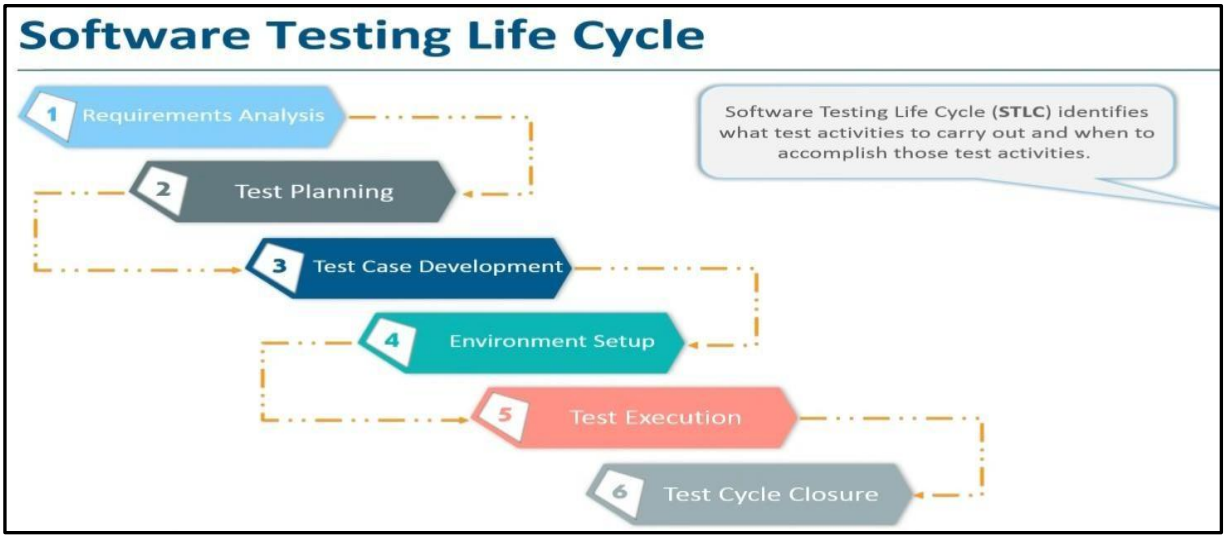
Software Testing Life Cycle (STLC):

STLC stands for Software Testing Life Cycle. Software Testing Life Cycle (STLC) is a sequence of different activities performed during the software testing process to ensure software quality goals are met. STLC involves both verification and validation activities.



STLC Phases:

1. Requirement Analysis
2. Test Plan
3. Test Scenarios (What)
4. Test Cases (How)
5. Test Environment Setup
6. Test Execution
7. Test Results (Pass/Fail)
8. Defect Reporting
9. Defect Tracking
10. Defect Re-Testing
11. Defect Closing
12. Regression Testing
13. Test Release (Closure)

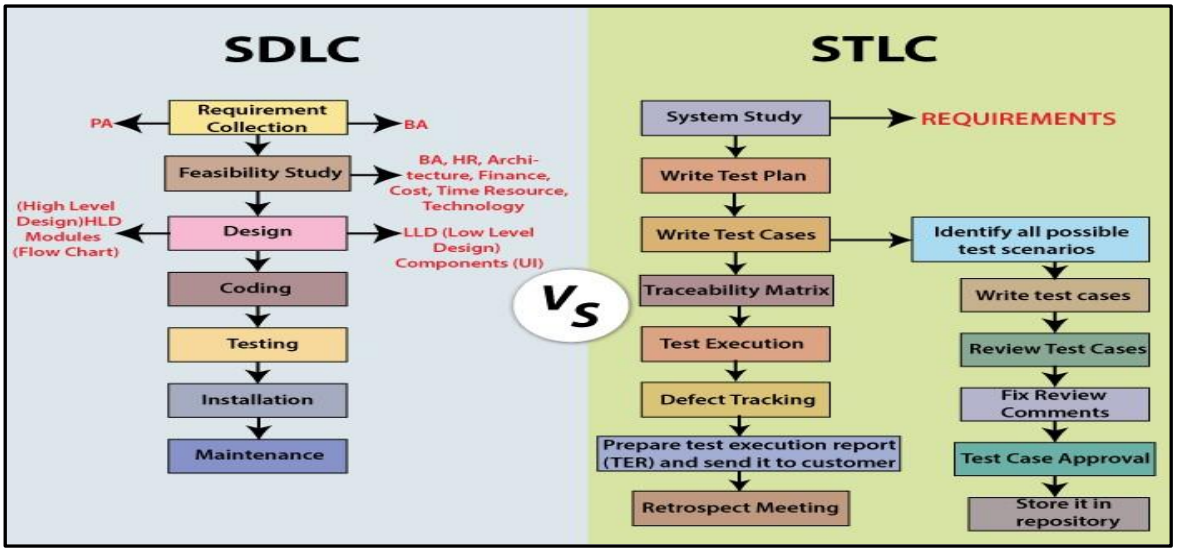


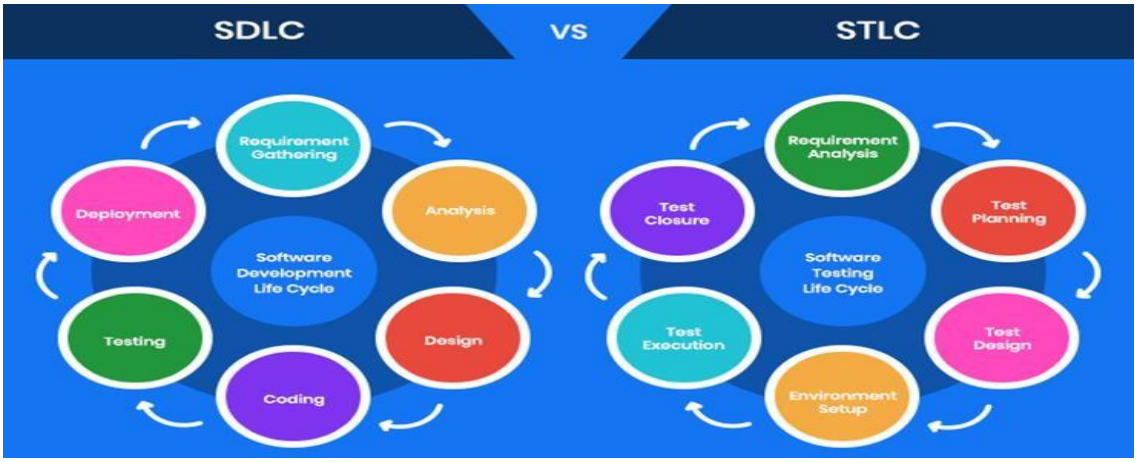
⇒ Requirement Analysis − When the SRD is ready and shared with the stakeholders, the testing team starts high level analysis concerning the AUT (Application under Test).

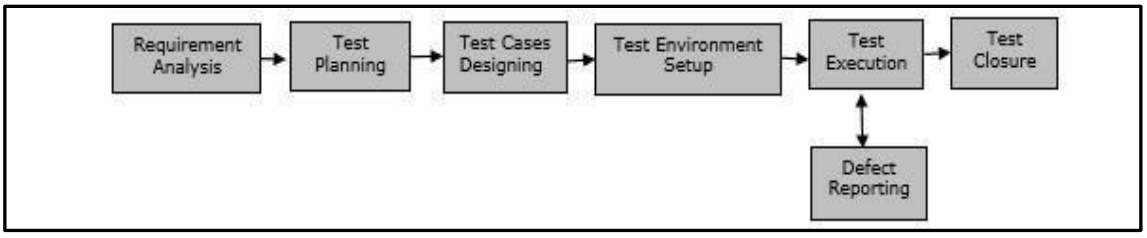
⇒ Test Planning − Test Team plans the strategy and approach.
⇒ Test Case Designing − Develop the test cases based on scope and criteria.
⇒ Test Environment Setup − When integrated environment is ready to validate the product.
⇒ Test Execution − Real-time validation of product and finding bugs.
⇒ Test Closure − Once testing is completed, matrix, reports, results are documented.

SNO	PHASE	Input	Activities	Responsibility	Out Come
1	Test Planning	Project Plan	➢Identify the Resources	Test Lead/Team Lead (70%)	Test Plan Document
	What to test	Functional Requirements	➢Team Formation	Test Manager (30%)	
	How to test		➢Test Estimation		
	when to test		➢Preparation of Test Plan		
			➢Reviews on Test Plan		
			➢Test Plan Sign-off		
2	Test Designing	Project Plan	➢Preparation of Test Scenarios	Test Lead/Team Lead(30%)	Test Cases Document
		Functional Requirements	➢Preparation of Test Cases	Test Engineers(70%)	Traceability Matrix
		Test Plan	➢Reviews on Test Cases		
		Design Docs	➢Traceability Matrix		
		Use cases	➢Test Cases Sign-off		
3	Test Execution	Functional Requirements	➢Executing Test cases	Test Lead/Team Lead(10%)	Status/Test Reports
		Test Plan	➢Preparation of Test Report/Test Log	Test Engineers (90%)	
		Test Cases	➢Identifying Defects		
		Build from Development Team			
4	Defect Reporting & Tracking	Test Cases	➢Preparation of Defect Report	Test Lead/Team Lead(10%)	Defect Report
		Test Reports/Test Log	➢Reporting Defects to Developers	Test Engineers (90%)	
5	Test Closure/Sign-Off	Test Reports	➢Analyzing Test Reports	Test Lead/Test Manger(70%)	Test Summary Reports
		Defect Reports	➢Analyzing Bug Reporting	Test Enginners(30%)	
			➢Evaluating Exit Criteria		

STLC vs SDLC:







Project/Product Life Cycle:

The project life cycle includes the steps required for project managers to successfully manage a project from start to finish. The Project Lifecycle is the sequence of phases through which a project progresses. It includes initiation, planning, execution, and closure. Learn more. The project life cycle is a 4-step framework designed to help project managers guide their projects successfully from start to finish. The purpose of the project life cycle is to create an easy-to-follow framework to guide projects.

4 Stages of the Project Life Cycle:

1. The project initiation stage:

This is the start of the project. The initiation stage of the project management life cycle is when you meet with clients and stakeholders to understand their goals, motivations, and hopes for the project. It may involve many sub-activities including: a feasibility study, identifying the scope, identifying deliverables, identifying project stakeholders, developing a business case, creating a statement of work, and possibly initial costs, price, and timeline for work to be done. Understand the goals, priorities, deadlines, and risks of the project

⇒ Identifying project objectives and deliverables
⇒ Outlining project risks, dependencies, constraints, and priorities
⇒ Establishing project scope based on deadlines and available resources
⇒ Submitting a project proposal for approval (our proposal maker can help you with that)

2. The project planning stage:

Once the project is approved from the initiation phase, it moves into the project planning stage

of the project life cycle. This phase involves creating a project plan, including the tasks, schedule, resources, and constraints on the project. The budget for the project is also created in this phase. In addition, risk should be expected and identified at this stage, as well as mitigation plans. Outline the tasks and timeline required to execute on the project.

⇒ Translating your proposal into a series of actionable tasks and scheduling them in a project roadmap
⇒ Documenting processes or workflows that your team will use (you could try using a process infographic for this)
⇒ Creating measurable short-term goals from high-level project goals
⇒ Addressing potential issues that could derail your roadmap

3. The project execution stage:

The project execution stage is the true start of the project, when you carry out all of the tasks and activities you mapped out in the planning stage. This is where the majority of the project work takes place, and it requires constant monitoring. Task owners begin work and the project manager oversees that tasks are done in a timely manner and workflow continues smoothly. Monitoring and Controlling (managing the work and financials) are a big part of this phase, as issues will always arise and require quick adjustments as the project progresses. Turn your plan into action and monitor project performance.

⇒ Monitor and control the execution process, reviewing the quality of the team's output
⇒ Adjust and update tasks, goals, and deadlines to meet changing conditions
⇒ Communicate between your team and the project stakeholders

Note: Create status reports to communicate execution progress throughout the project management process

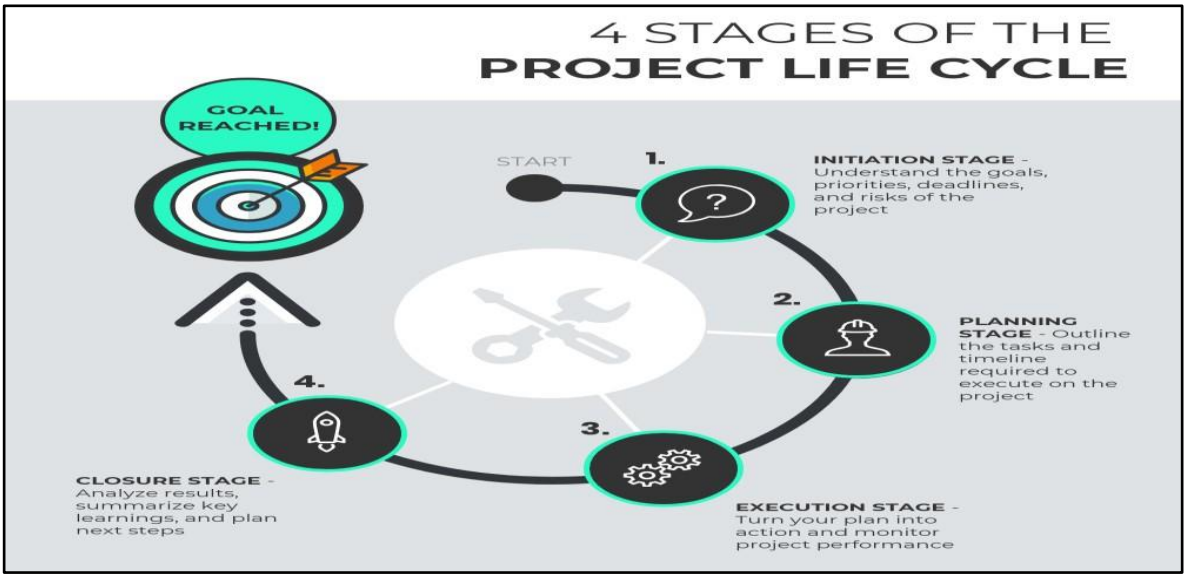
4. The project closure stage:

Once the team has completed all the tasks, and the project owner signs off that all deliverables are complete, the project is closed. Any documentation is handed over to the project owner and if required to an ongoing maintenance organization. The project is then analysed for performance to determine whether the project's goals were met (tasks completed, on time and on budget). Analyze results, summarize key learnings, and plan next steps. Once you've achieved your project goals and the results have been signed off on by your stakeholders, it's time for the project closure stage.

⇒ Hand off deliverables
⇒ Release team members and project resources
⇒ Analyze project performance in a project retrospective

Note: A project retrospective meeting is as much about reviewing the success of the project as it is about extracting learnings that can apply to future projects. Projects will never go without obstacles, and there will always be things to learn that will ease the progress of other projects.

⇒ Another duty of a project manager in the project closure phase can be to analyze the performance of the team, based on the quality of their work and how well they were able to meet deadlines.



Project Management Plan (PMP):

The Project manager creates the project management plan for project. A project management plan is a set of documents that outline the how, when and what-ifs of a project's execution. A project management plan is a formal document that defines how a project is going to be carried out. It outlines the scope, goals, budget, timeline, and deliverables of a project, and it's essential for keeping a project on track. There are no shortcuts to a thorough understanding of your project than through a well-written, well-structured project plan document. The project management plan outlines the scope, budget, goals, timeline, and project deliverables. The project management plan provides essential project information and can be used to introduce project members to the project. Purpose of project management plan is to manage the project/To execute project.

Entry criteria to prepare PMP:

⇒ RFP
⇒ Sow or Work order
⇒ Project plan guidelines
⇒ Project plan template

Exit criteria to prepare PMP:

⇒ Project plan should be reviewed and approved.

The main aspects in PMP:

⇒ Scope Management
⇒ Schedule Management
⇒ Quality Management

⇒ Resource Management like people, tools and other.
⇒ Communication Management
⇒ Project change Management (CCB)
⇒ Risk Management
⇒ Data Management
⇒ SDLC model selection (Agile)

Test Plan:

A Test Plan is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product. The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager.

A well-crafted test plan is a dynamic document that changes according to progressions in the project and stays current at all times. It is the point of reference, based on which testing activities are executed and coordinated among a QA team.

The test plan is also shared with Business Analysts, Project Managers, Dev teams, and anyone else associated with the project, this mainly offers transparency into QA activities so that all stakeholders know how the software will be tested. Test Plan Template may vary from one company to another and One Project to another but the Objective is the same.

Why are Test Plans important?

They help individuals outside the QA teams (developers, business managers, customer-facing teams) understand exactly how the website or app will be tested. They offer a clear guide for QA engineers to conduct their testing activities. They detail aspects such as test scope, test estimation, strategy, and so on. Collating all this information into a single document makes it easier to review by management personnel or to re-use for other projects.

What If There is No Test Plan?

Below are some of the situations that may occur if there is no test plan in place:

⇒ Misunderstanding roles and responsibilities.
⇒ The test team will have no clear objective.
⇒ No surety when the process ends.
⇒ Undefined test scope may confuse testers.

Components of a Test Plan:

1. Introduction:

Describe the purpose of this Software Test Plan, If it links in with other plans (for example, Project Plan or Master Test Plan) then identify the level to which this plan belongs.

2. Scope: Details the objectives of the particular project. Identify the scope of this Software Test Plan in relation to the overall project plan that it relates to.

3. Test Plan Identifier:

The Test Plan Identifier is just a type of unique number or reference-id to identify this test plan and the software that it is related to.

4. Schedule:

Details start dates and deadlines for testers to deliver results.

5. Reference Documents:

List all documents that support this test plan. Refer to the actual version/release number of the document as stored in the document or configuration management system.

Documents that could be referenced include:

⇒ Project Plan
⇒ Requirements specifications
⇒ Test Strategy
⇒ High Level design document
⇒ Low Level design documents
⇒ Quality system process documents
⇒ Corporate standards and guidelines etc…

6. Features/Functions to be Tested:

This is a high-level view of what is to be tested from the user's viewpoint of what the system does.

7. Features/Functions not to be Tested:

What is not to be tested can be sometimes just as important as stating what is to be tested. It removes any ambiguity in order that other project stakeholders are clear on what to expect from the test phases.

8. Risks:

development is full of risk, and the testing phases are no exception. It is always wise to take an active lead in managing the risks facing you. Details what risks may occur during software testing, and what risks the software itself may suffer if released without sufficient testing.

9. Test Strategy (Approach):

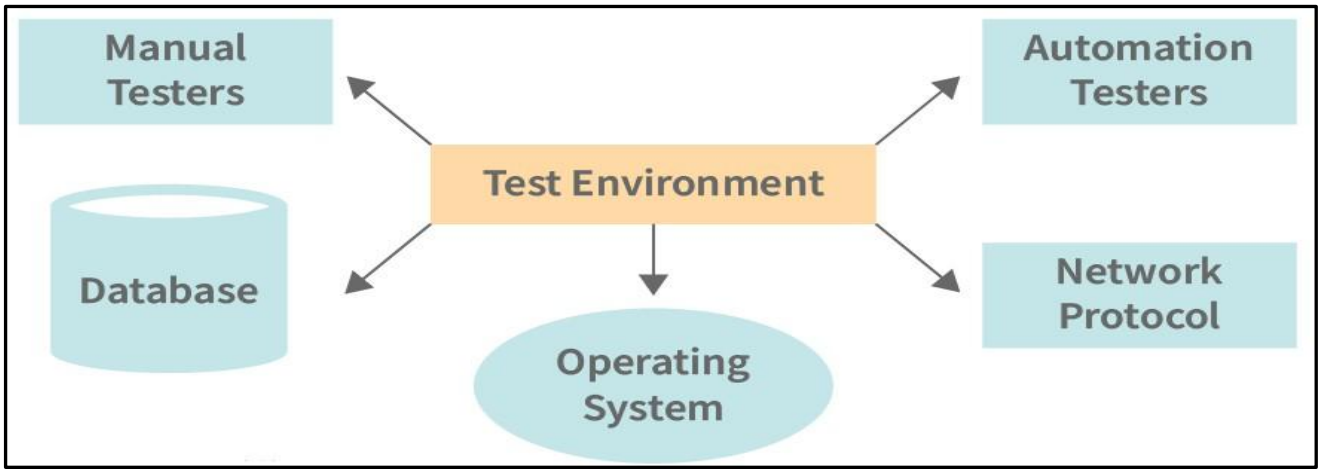
The test approach is the overall test strategy that underpins the whole test plan. A test approach asks, "how are you going to test the software?"

10. Test Environment:

The test environment encompasses the software being tested, related platform software, third-party software, communications software, etc. Ensure that you have the resources required to install, set up and configure your test environment.

Testbed.

Testbed is generally referred to as a digital platform that is used for testing an application. It includes an operating system, database, hardware, network configuration, software application under test, and all other software-related issues.



11. Tools:

Details what tools are to be used for testing, bug reporting, and other relevant activities.

12. Human Resources:

List the members of your test team here. Think about the specialisms each has when assigning them work tasks.

Define Roles and Responsibilities

13. Resource Allocation:

Details which tester will work on which test.

14. Training:

Since you have reviewed the roles and responsibilities of test-related personnel it may have become apparent that there are skills gaps. If this is the case then you will need to decide the best way to fill these gaps.

15. Defect Management:

Details how bugs will be reported, to whom and what each bug report needs to be accompanied by. For example, should bugs be reported with screenshots, text logs, or videos of their occurrence in the code?

16. Management and Metrics:

Who will be managing the testing? Will different people be managing different phases, for example, integration-test phase, component test phases? Does the Project Manager require metrics to be collected from you? If so, then list these here and state when and how you will construct the metrics and report them.

17. Test Estimation and Schedule:

The duration of the testing schedule should have been estimated as a result of a team consensus. It can be useful to break test estimates down to a level that matches that in the related functional specifications or requirements.

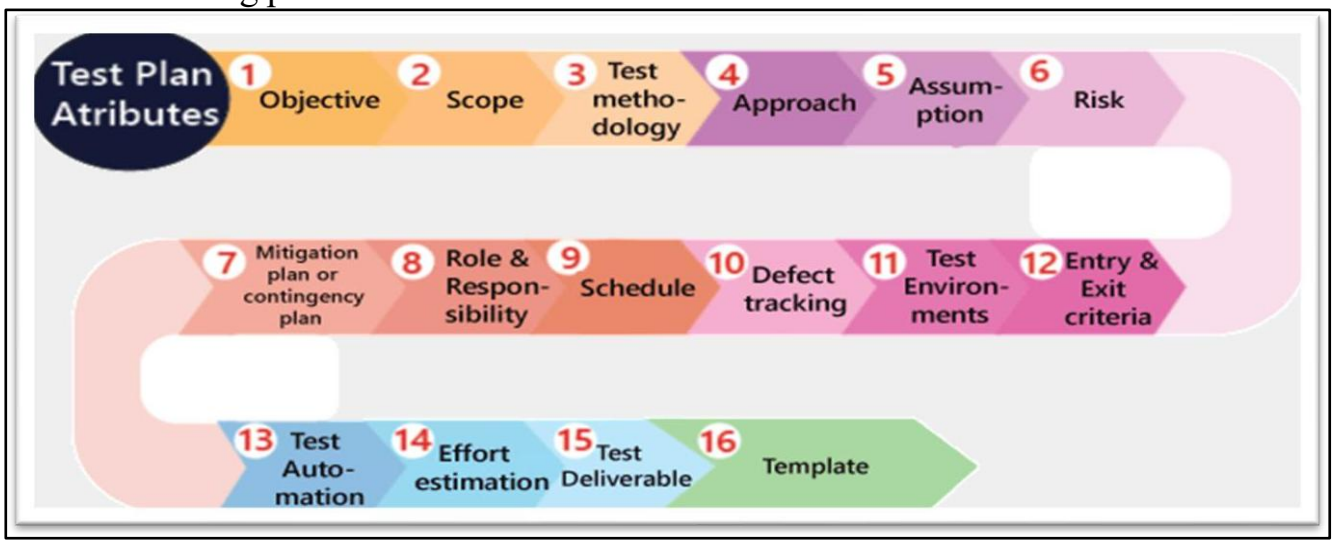
18. Test Phase Entry, Exit and Suspension Criteria:

In order that you can manage software stability and quality grade through successive test phases it is useful to plan for test phase Entry Exit and Suspension criteria. Details when testing activities must stop. This part describes the results that are expected from the QA operations, giving testers a benchmark to compare actual results to.

19. Test Deliverables:

Test deliverables are essentially the work products of the entire test regime. This may include:

⇒ Test estimates
⇒ Test schedules
⇒ Test Plan
⇒ Test Specification
⇒ Test Scripts
⇒ Test Data
⇒ Test tools
⇒ Test execution results
⇒ Test Reports
⇒ Issue Logs
⇒ Lessons to be learned
⇒ Test Risk Register
⇒ Test Training plans



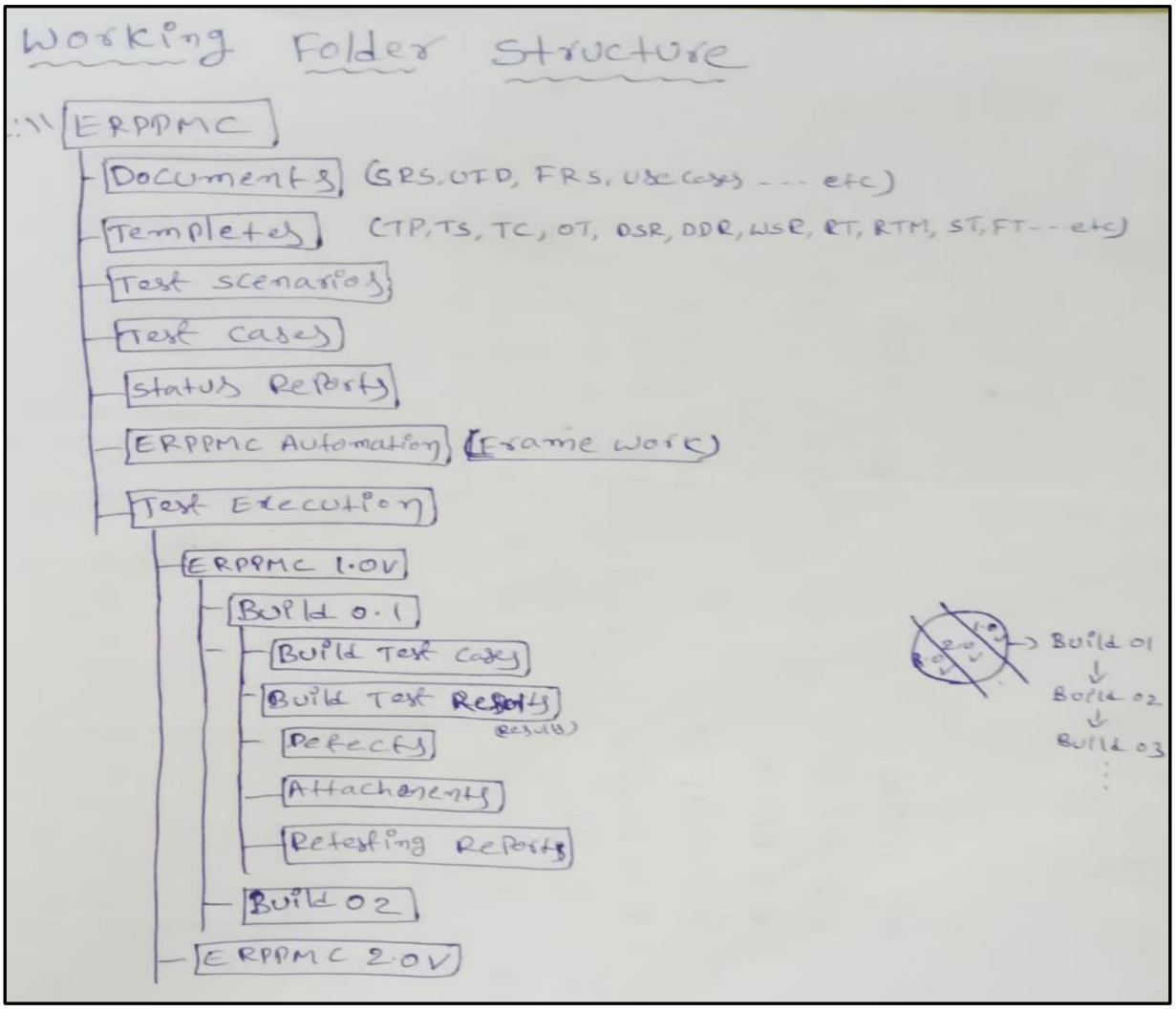
Working Folder Structure:

A working folder structure in software testing refers to the organization of the files and directories that are used to store and manage testing artifacts such as test plans, test cases, test data, test results, and other related documents. Typically, a working folder structure will

contain subfolders for various types of files, such as documents, images, templates and among others. Working folder structures are essential for maintaining organization, ensuring that files are stored in a consistent and logical manner. This not only makes it easier for individuals or teams to find what they need but also helps to ensure that important files are not lost or accidentally deleted. By using this folder structure, testers can keep their testing documents and results organized and easily accessible, which can help improve their efficiency and effectiveness.

Organizing the working folder structure in this way helps to ensure that testing artifacts are easily accessible and well-organized, which can save time and improve the efficiency of the testing process.

The following is an example of a typical Folder Structure for Software Testing:

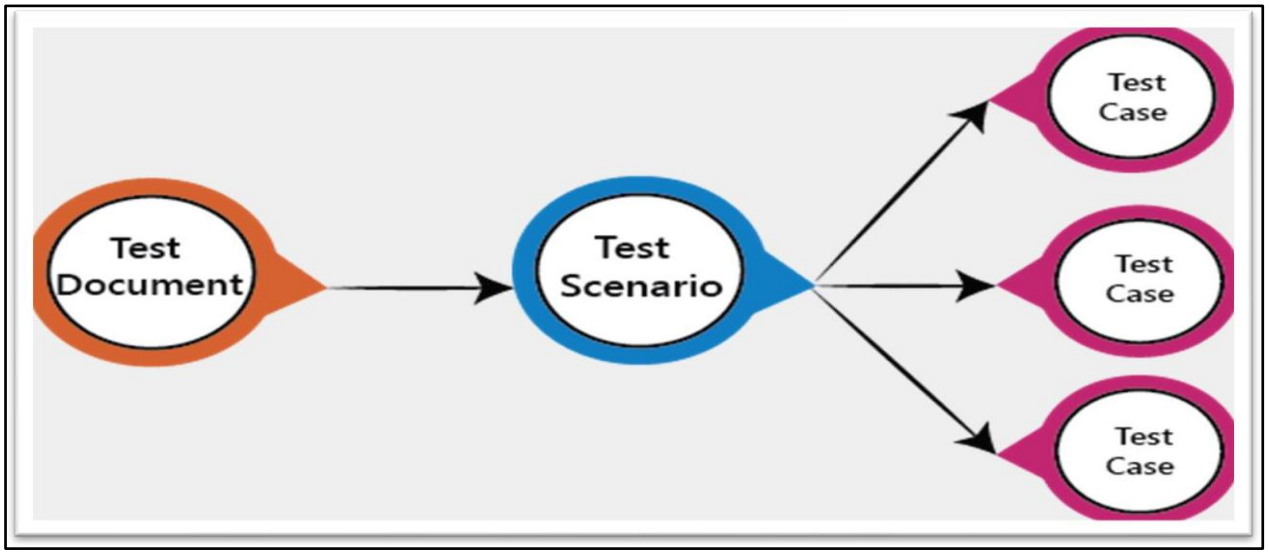


Test Scenarios:

Test scenario is 'What functionality is to be tested'. Identify all the possible areas to be tested. "**What is to be tested**". It is also called Test Condition or Test Possibility. The test scenarios are those derived from the use case and give the one-line information about what to test. Test scenarios are created by software testers and are used to design test cases that will verify the software application's behavior under various conditions. Test scenarios can be created based

on user requirements, functional specifications, use cases, and other design documents. They should cover all the critical functionality of the software application and be designed to test the software thoroughly. It is a single line statement and test cases comprise step-wise description to complete the purpose of the test scenario statement. One Test Scenario can have multiple 'Test Cases'. It is a single line statement and test cases comprise step-wise description to complete the purpose of the test scenario statement.

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.



Important Points of Test Scenarios:

- The test scenarios can never be used for the text execution process because it does not consist of navigation steps and input.
- These are the high-level documents that talks about all the possible combination or multiple ways or combinations of using the application and the primary purpose of the test scenarios are to understand the overall flow of the application.
- Test scenario becomes more important when the tester does not have enough time to write test cases, and team members agree with a detailed liner scenario.
- The test scenario is a time saver activity.
- It provides easy maintenance because the addition and modification of test scenarios are easy and independent.

Entry Criteria to identify Test Scenarios:

- Approved Test Plan.
- Approved SRS or FRS.
- Design document like UID, Use cases etc…
- Test Scenario guidelines.
- Approved Test Scenario Template.

Exit Criteria for identifying Test Scenarios:

- Test Scenarios should be reviewed and approved [Mapping Test Scenarios (TS#) with Requirements (Req#)].
- Once Test Scenarios are approved, Test Lead will create a base line for test scenarios (TS 1.0v) and he will update same into Repository (TFS/VSS/GIT).

As an example, consider a test scenario:

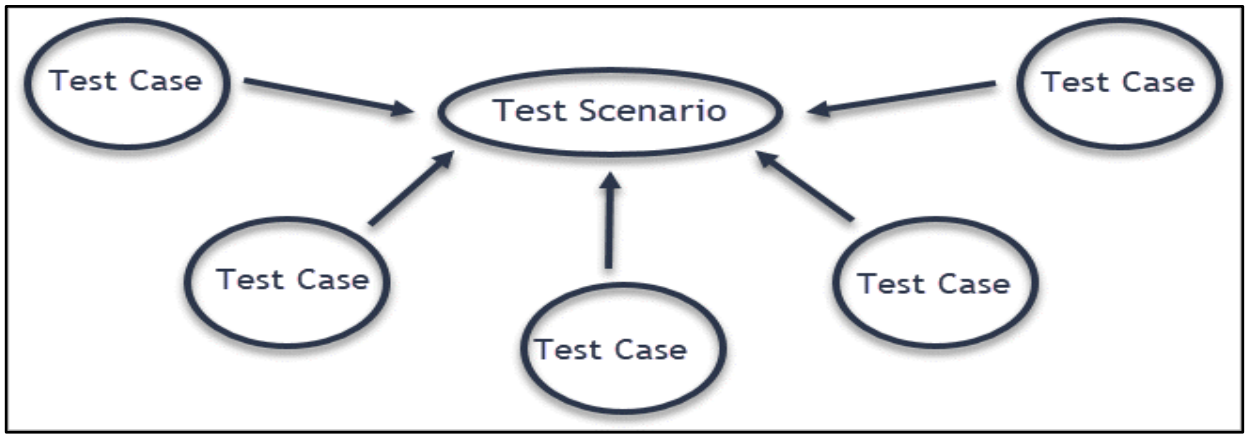
"Verify that the user is not able to login with incorrect credentials".

Now, this test scenario can be further broken down into multiple test cases like…

⇒ Checking that a user with the correct username and incorrect password should not be allowed to log in.

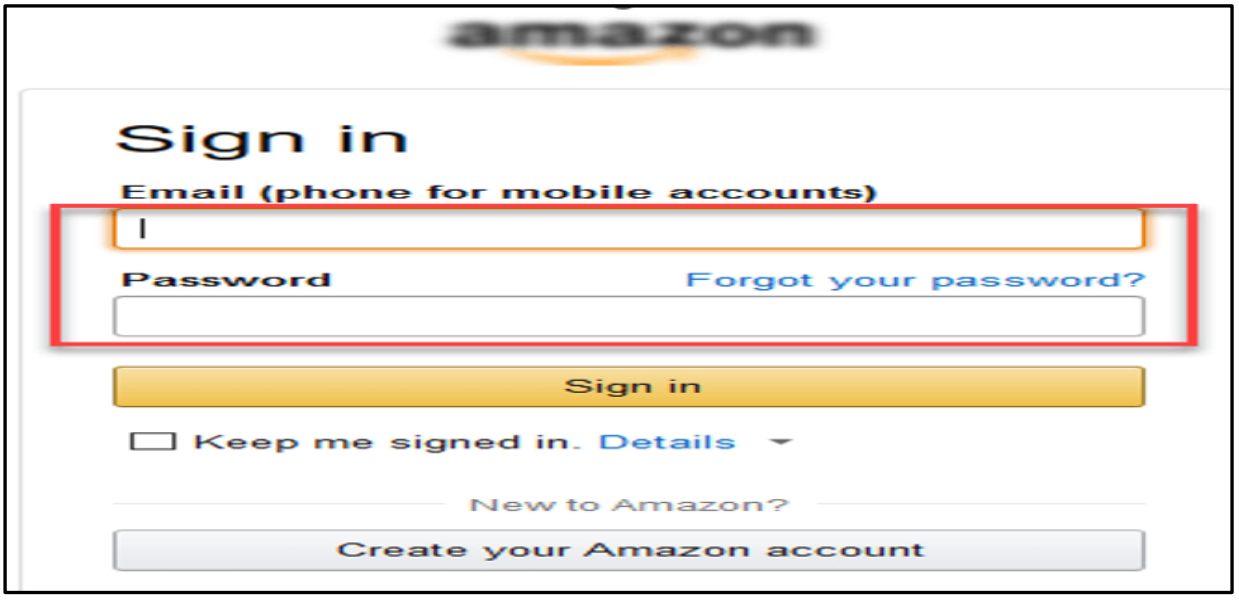
⇒ Checking that a user with an incorrect username and correct password should not be allowed to log in.

⇒ Verifying that users with incorrect usernames and incorrect passwords should not be allowed to log in.



Example 1: Test Scenario for E-Commerce Application.

Test Scenario 1: Check the Sign in Functionality.



In order to help you understand the difference Test Scenario and Test Cases, specific test cases for this Test Scenario would be

1. The Sign in screen contains the required elements like Email (phone for mobile accounts), Password, Sign in button. Forgot your Password link, keep me signed in check box, etc.
2. Check system behavior when valid email id and valid password is entered.
3. Check system behavior when invalid email id and valid password is entered.
4. Check system behavior when valid email id and invalid password is entered.
5. Check system behavior when invalid email id and invalid password is entered.
6. Check system behavior when email id and password are left blank and Sign in entered.
7. Check Forgot your password is working as expected.
8. Check system behavior when valid/invalid phone number and password is entered.
9. Check system behavior when "Keep me signed" is checked.

Test Scenario Template

* Test Cases field is Optional

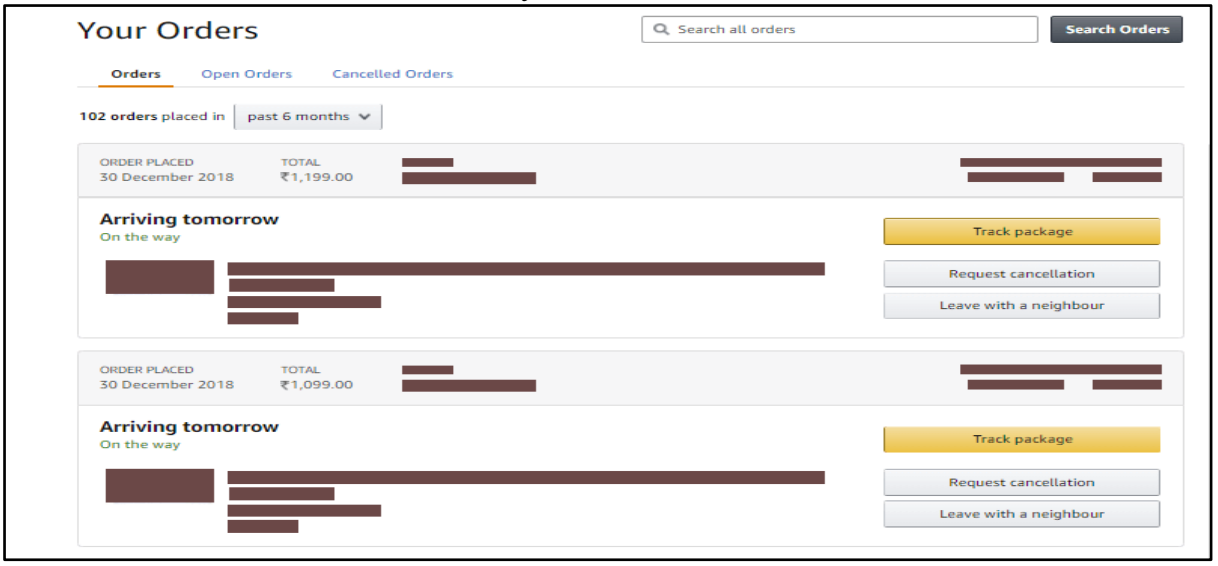
Test Scenario #	Requirement ID	Test Scenario	Test Cases
1	S1.1	Check the Login Functionality	1.Check system behavior when valid email id and password is entered. 2.Check system behavior when invalid email id and valid password is entered. 3.Check system behavior when valid email id and invalid password is entered. 4.Check system behavior when invalid email id and invalid password is entered. 5.Check system behavior when email id and password are left blank and Sign in entered. 6.Check Forgot your password is working as expected 7.Check system behavior when valid/invalid phone number and password is entered. 8.Check system behavior when "Keep me signed" is checked
2	S1.2	Check the Search Functionality	
3	S1.3	Check the Product Description Page	
4	S1.4	Check the Payments Functionality	
5	S1.5	Check the Order History	
6	S1.6	Check Home Page behavior for returning customers	
7	S1.7	Check Category/Product Pages	
8	S1.8	Check Customer Service/Contact Pages	
9	S1.9	Check Daily Deals pages	

Test Scenario 2: Check the Search Functionality.

Test Scenario 3: Check the Product Description Page.

Test Scenario 4: Check the Payments Functionality.

Test Scenario 5: Check the Order History.



Example 2: Test Scenarios for a Banking Site.

- Test Scenario 1: Check the Login and Authentication Functionality.
- Test Scenario 2: Check Money Transfer can be done.
- Test Scenario 3: Check Account Statement can be viewed.

- Test Scenario 4: Check Fixed Deposit/Recurring Deposit can be created.

How to write Test Scenarios:

As a tester, follow the following steps to create Test Scenarios:

1. Read the requirement document such as BRS (Business Requirement Specification), SRS (System Requirement Specification) and FRS (Functional Requirement Specification) of the software which is under the test.
2. Find all the possible ways by which the user can operate the software.
3. After reading the requirement document and completion of the scheduled analysis make a list of various test scenarios to verify each function of the software.
4. Once you listed all the possible test scenarios, create a traceability matrix to find out whether each and every requirement has a corresponding test scenario or not.
5. Supervisor of the project reviews all scenarios. Later, they are evaluated by other stakeholders of the project.

Test Scenario Template for Manual Testing:

TEST SCENARIOS TEMPLATE						
		Project Code :			Reviewed By:	
		Project Name:			Date Reviwed:	
		Identified By:			Approved By:	
		Date of Identified:			Date Approved :	
TS #	Req #	MAIN FUNCTIONALITY	SUB FUNCTIONALITY	TEST SCENARIO NAME	TEST SCENARIO OBJECTIVE	REFERENCE ID

A Test Scenario document can include the following fields:

1. Test Scenario ID(TS#): In this field we will add the Test Scenario ID.
2. User Story ID/Requirement ID(Req#): In this field we will add the User Story ID or Requirement ID which is related to the scenario we are going to write.
3. Main Functionality: In this field we will add the Main Functionality of the Test Scenario.
4. Sub Functionality: In this field we will add the Sub Functionality of the Test Scenario.
5. Test Scenario Name: In this field we will add the name of the Test Scenario.
6. Test Scenario Objective: In this field we will add the main objective of the Test Scenario.
7. Reference ID: In this field we will add the any document or reference id of the Test Scenario.

Test Scenario Example for Manual Testing:

TEST SCENARIOS For ERPPMC						
		Project Code :	LT_333		Reviewed By:	CHARAN
		Project Name:	ERPPMC		Date Reviwed:	23-Jul-22
		Identified By:	N.KRISHNA		Approved By:	VENKAT
		Date of Identified:	22-Jan-23		Date Approved :	24-Jan-23
TS #	Req #	MAIN FUNCTIONALITY	SUB FUNCTIONALITY	TEST SCENARIO NAME	TEST SCENARIO OBJECTIVE	REFERENCE ID
TS_001		ERPPMC	Home Page	ERPPMC_Home Page	Verify usability and user interface of Home page	UID_Page002
TS_002		"	"	"	Verify functionality of 'Home'	"
TS_003		"	"	"	Verify functionality of 'Employee Login'	"
TS_004		"	"	"	Verify functionality of 'Preview of the Computers'	"
TS_005		"	"	"	Verify functionality of 'Contact Us'	"
TS_006		"	"	"	Verify functionality of [Submit]	"
TS_007		"	"	"		"

Test Cases:

A test case is a document which contains a set of input values, execution preconditions, expected result which is developed to achieve a particular objective, such as to verify compliance with specific requirements.

(or)

How "What is to be tested".

What is a good test case?

"A test case that has high probability of catching defects is called a good test case";

Entry Criteria to Prepare Test Case:

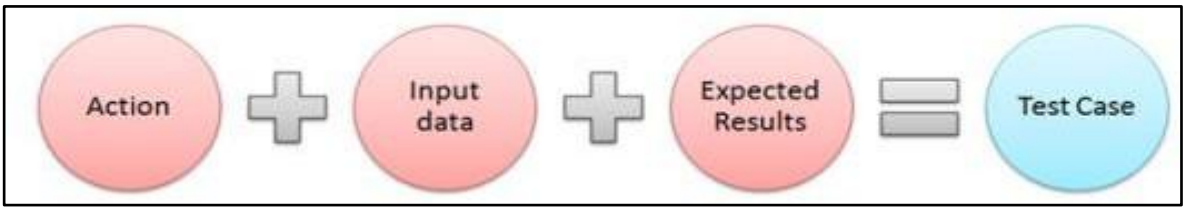
⇒ Approved Test Plan
⇒ Approved SRS
⇒ Functional requirement specification
⇒ Approved Test Scenario's
⇒ Test Case Template
⇒ Test Case Guidelines

Exit Criteria for Preparing Test Case:

⇒ Test Case should be reviewed and approved.
⇒ Mapping Test Cases (TC#) against Test Scenario's (TS#) and Requirement (Req#).
⇒ Once the Test Case approved the TL can create the baseline.
⇒ And also, will update the same in the configuration repository.

Test Case Contains (or) Test Case Template Contains:

A test case should contain particulars such as test case name, objective, test conditions, input data requirements, expected results and the name of the person who prepared it.



Test Case Design:

A good test case design technique is crucial to improving the quality of the software testing process. This helps to improve the overall quality and effectiveness of the released software. Test case design refers to how you set-up your test cases. It is important that your tests are designed well, or you could fail to identify bugs and defects in your software during testing.

There are many different test case design techniques used to test the functionality and various features of your software. Designing good test cases ensure that every aspect of your software gets tested so that you can find and fix any issues.

A Basic Example of Test Case Design:

Title: Login to the website or app

Description: User should be able to successfully log in to their account on the website/app

Preconditions: User must already be registered and use their correct login details

Assumptions: They are using a supported device or browser to log in

Test Steps:

⇒ Open website or app
⇒ Enter the username and password in the appropriate fields
⇒ Click “login”

Expected Result: The user should log in successfully.

A	B	C	D	E	F	G	H	I	J	K	L	M	N
TEST CASES TEMPLATE													
Componay Logo	Project Code							Total No of Tc'S					
	Project Name							Passed					
	Prepared By		Date Prepared					Failed					
	Reviewed By		Date Reviewed					Not Executable					
	Approved By		Date Approved					Defects Reported					
	TestCase Name		TestCase Objective					Tc'S Automated					
	Executed Name		Date Executed										
	Version		Build										

TC#	TS#	Test Design/Test Steps	Test Data	Expected Result	Actual Result	Pass	Fail	Not Executable	Severity				Defect ID
									Critical	Major	Modarate	Minar	

A Test Scenario document can include the following fields:

Test Case Id (TC#): Unique identifier of the test case.

Test Scenario ID(TS#): Unique identifier of the test scenario.

Test Steps/Design: Detailed steps for performing test cases.

Test Data: Test data value used in the test case.

Expected Result: Estimated result to pass the test.

Actual Result: Actual result after executing the test steps.

Test Result: Status of the test execution (Pass or Fail).

Date: Test execution/prepared/reviewed/approved date.

Executed By: Person name executing the test case.

Good test Case Design:

⇒ Effective [Find fault (Verify objective)]
⇒ Evolvable [Easy to maintain]
⇒ Economic cheap to use [Minimum number of test cases to cover maximum functionalities].
⇒ Test case design techniques are such thought process.
⇒ Represents to others easy to understand.
⇒ Doesn't do unnecessary things.

Project Name:

Test Case Template

Test Case ID: Fun_10
Test Priority (Low/Medium/High): Med
Module Name: Google login screen
Test Title: Verify login with valid username and password
Description: Test the Google login page

Test Designed by: <Name>
Test Designed date: <Date>
Test Executed by: <Name>
Test Execution date: <Date>

Pre-conditions: User has valid username and password
Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to login page	User= example@gmail.com	User should be able to login	User is navigated to	Pass	
2	Provide valid username	Password: 1234		dashboard with successful		
3	Provide valid password			login		
4	Click on Login button					

Post-conditions:
User is validated with database and successfully login to account. The account session details are logged in database.

Test Case ID	118	Test Case Description	Staging Regression Login Page		
Created By	Flannery	Reviewed By	Cody	Version	2
QA Tester's Log	Review comments from Regression Version 2				
Tester's Name	Evan	Date Tested	July 20, 2020	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:	Test Data
1	Access to Chrome Browser	Username = info@workwithloop.coms
2	Access to testing environment	Password = loopisthebest

Test Scenario: Verify on entering valid userid and password, the customer can login

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Clicking "Forgot Password" link will lead user to workwithloop.com/forgotpassword	Site should open to Forgot Password modal	As Expected	Pass
2	Enter blank email submission in email box	Shows email error	As Expected	Pass
3	Enter email with no account	Shows user does not exist	As Expected	Pass
4	Valid email submission	Shows reset password sent	As Expected	Pass
5	Go to valid email inbox	Able to go to valid email inbox	As Expected	Pass
6	Validate email is received	A validation email is received	As Expected	Pass

When writing test cases, you should follow the following guidelines:

- Assess the project's risks and deadlines before planning and prioritizing test cases and write accordingly.
- Keep the 80/20 rule in mind. For the best coverage of your application, 20% of your tests should cover 80% of its functionality.
- Make sure you don't try to test all cases at once, but rather improvise them as you go.

- Create a list of your test cases and categorize them according to business scenarios and functionality.
- Provide test cases in a way that can easily be understood by others and modified if necessary.
- Remember that the software designed is ultimately for customers, so keep their requirements in mind.
- Manage a stable release cycle by using a test management tool.
- Keep track of your test cases on a regular basis. Test cases must be unique and irrelevant or duplicated must be removed.

Why Test Techniques:

⇒ Exhaustive testing (Use of all possible inputs and conditions) is impractical.
⇒ Need thought process that help us select test cases more intelligently.

What is a Testing Technique?

⇒ A procedure for selecting or designing tests based on a structure or functional model of the software.
⇒ Successful at finding faults.
⇒ A best practice.
⇒ A way of delivering good test cases.
⇒ A way of objectively measuring a test effort.

Test Case Design Techniques:

Software Testing Techniques help you design better test cases. The main purpose of test case design techniques is to test the functionalities and features of the software with the help of effective test cases.

There are two main categories of Test Design Techniques. They are:

1. Static Techniques
2. Dynamic Techniques

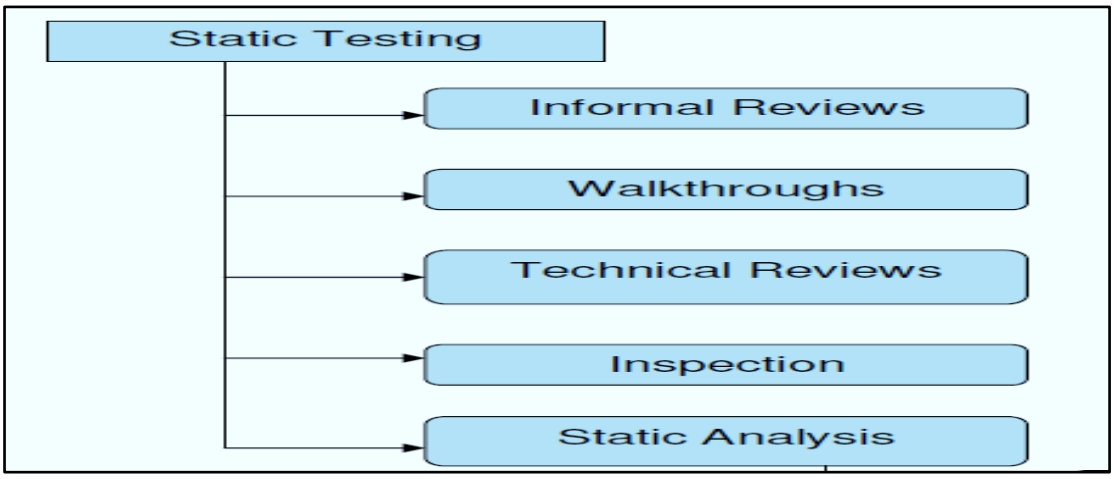
1.Static Techniques:

Static testing is a type among different software testing techniques that are performed to identify the errors in software without executing the code. This type of testing is generally performed at the initial stage of software development to avoid defects as it is easier to figure out the sources of failures and can be fixed quickly.

Benefits of Static Testing Technique:

⇒ Helps in identifying the flaws in code
⇒ The testing is conducted by trained software developers with good knowledge of coding

⇒ It is a fast and easy way to find and fix errors
⇒ With automated tools, it becomes quite fast to scan and review the software
⇒ The use of Automated tools provides mitigation recommendations
⇒ With static testing it is possible to find errors at an early stage of the development life cycle, thus, in turn, reducing the cost of fixing.



Static Design Techniques:

A. Informal reviews:
As the name suggests, an informal review done by an individual without any process or documentation.

B. Walkthrough:
A Walk-through is a step-by-step presentation of different requirements and design documents by their authors. This is done with the intent of finding defects or any missing pieces in the documents.

C. Technical reviews:
A technical review involves reviewing the technical approach used during the development process. It is more of a peer review activity and less formal as compared to audit and inspection.

D. Inspection:
An inspection is a formal and documented process of reviewing the different documents by experts or trained professionals.

E. Static analysis:
The static analysis techniques for the source code evaluation using tools are-
⇒ Control flow analysis – The control flow analysis requires analysis of all possible control flows or paths in the code.
⇒ Data flow analysis – The data flow analysis requires the analysis of data in the application and its different states.

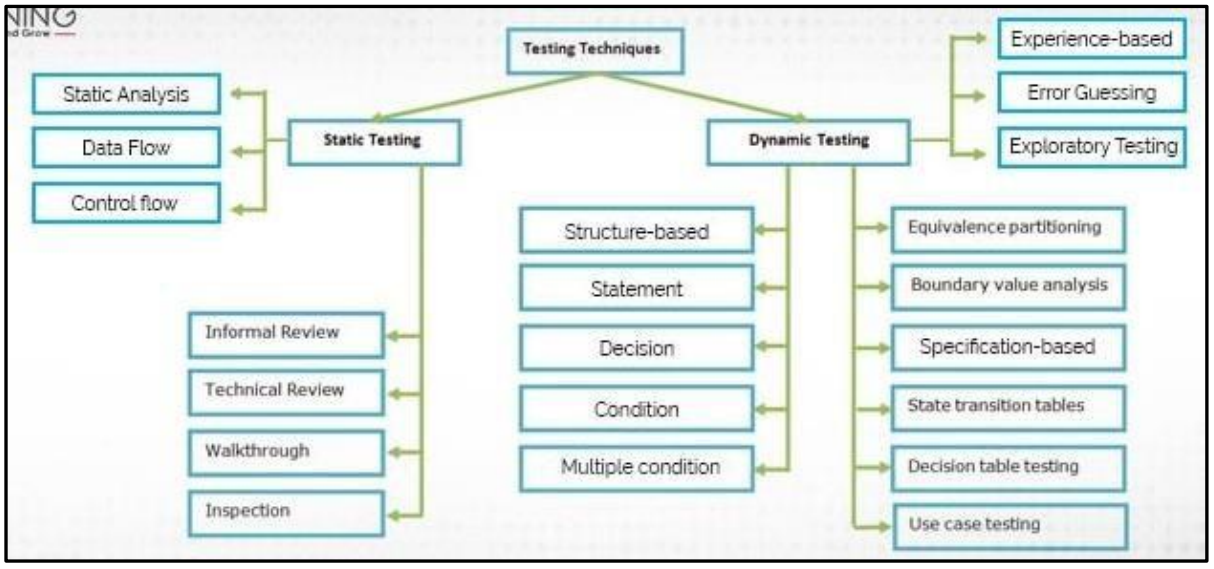
2. Dynamic Techniques:

Dynamic test design techniques involve testing by running the system under test. In this

technique, the tester provides input data to the application and executes it. This is done to verify its different functional and non-functional requirements.

They are 3 types of Dynamic Techniques

1. Black Box (Specification-Based) Test case design techniques.
2. White Box (Structure-Based) Test case design techniques.
3. Experience-Based techniques.



1. Black Box (Specification-Based) Test Case Design Techniques:

Specification-based – Specification-based test design techniques are also referred to as black-box testing. These involve testing based on the specification of the system under test without knowing its internal architecture.

The different types of specification-based test design or black box testing techniques are…

1. Equivalence Class Partitioning.
2. Boundary Value Analysis.
3. State transitions testing.
4. Decision Tables (Cause-Effect Graphing):

1. Equivalence Class Partitioning (ECP):

ECP is a method for deriving test cases. It identifies similar functionalities and divides equally.

⇒ In this method, classes of input conditions called equivalence classes are identified such that each member of the classes causes the same kind of processing and output to occur.

⇒ Equivalence partitioning drastically cuts down the number of testcases required to test a system reasonably.

⇒ A technique that divides the input domain of a program into classes of data from which test cases can be derived.

⇒ For each piece of the specification, generate one or more equivalence classes.
⇒ Labels the classes as "valid" and "invalid".
⇒ Generate test cases that cover as many possible valid and invalid equivalence classes.

Example 1:

If an item can be -9999 to +9999 identify the equivalence classes?

Valid: -9999 to +9999

Invalid: -10,000 to 10,000

Less than: -9999, Greater than +9999

Example 2:

Name: [ ] 5 to 30 Alphabets

Here Blank is invalid.

ECP:

5 to 30 Valid Partition

< 5 Invalid Partition > 30

Example 3:

Program accepts 1 to 100characters and divides equivalence classes.

Valid Class: 1 to 100

Invalid Class: < 1 and > 100

Example 4:

Percentage: [ ] *Accepts Percentage value between 50 to 90.

Equivalence Class Partitioning		
Invalid	Valid	Invalid
<=50	50-90	>=90

Example 5:

Consider an OTP number that contains only 6-digit number, greater and even less than six digits will not be accepted, and the application will redirect customer or user to error page. If password entered by user is less or more than six characters, that equivalence partitioning method will show an invalid OTP. If password entered is exactly six characters, then equivalence partitioning method will show valid OTP.

Enter OTP		*Must include six digits

Equivalence Partioning			
Invalid	Invalid	Valid	Valid
Digits>=7	Digits<=5	Digits=6	Digits=6
67545678	9754	654757	213309

2. Boundary Value Analysis:

A technique in which the test cases that explore the boundary conditions have a higher probability of detecting error.

⇒ Boundary Value Analysis is the test case design technique to test boundary value between partitions.
⇒ Boundary value is an input or value on the border of an equivalence partition.
⇒ It consists of start-end, lower-upper, maximum-minimum on inside and outside boundaries.
⇒ Boundary Value Analysis (BVA) checks the boundary values of Equivalence Class Partitioning (ECP), hence BVA comes after ECP.

Formula: Minimum, Maximum, Min-1, Max+1

Valid Boundaries: Minimum, Maximum

Invalid Boundaries: Min-1, Max+1

Example 1:

In Insurance application, user can apply for different types insurance policies. when user applies for type B insurance, system asks to enter the age of the customer. In functional specifications it is mentioned that user age should be greater than 18 years and less than 60.

Ans: Valid Boundaries: 19 and 59

Invalid Boundaries: 18 and 60

Example 2:

A text box accepts 1 to 100 characters and identifies Boundary

values. Ans: Valid Boundaries: 1 and 100

Invalid Boundaries: 0 and 101

Example3:

Assume you're testing an input box accepting numbers from 1 to 20. As a result of boundary value analysis, we can divide test cases into three categories:

The test data will be the same as the input boundaries of input: 1 and 20.

Input values above the extreme edges: 2 and 21.

Input values below the extreme edges: 0 and 19.

Therefore, the boundary values are 0, 1, 2, and 19, 20, 21.

Example: Loan Application

Customer Name		[2 to 64 characters (A to Z, a to z, space)]
Account Number		[6 digits, 1st non zero]
Lone Amount Requested		[$500 to $9000]
Term of Loan		[1 to 30 years]
Monthly Repayment		[Minimum $10]

1.Customer Name: Valid 2 64 1 65 Invalid

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Customer Name	2 64	<2 >64	2 6	1 65 Blank
Invalid Characters:	Any other characters			

2. Account Number:

5 6 7

Invalid Valid Invalid

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Account Number	6 1st non zero	<6 >6 1st digit=0 1st digit= abcd	100000 999999	99999 1000000 Blank

3. Lone Amount:

499 500 9000 9001

Invalid Valid Invalid

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Lone Amount	500 9000	<500 >9000 Non-Numeric	500 9000	499 9001 Blank

3.State Transition Testing:

State Transition Testing is basically a black box testing technique that is carried out to observe the behaviour of the system or application for different input conditions passed in a sequence. In this type of testing, both positive and negative input values are provided and the behaviour of the system is observed.

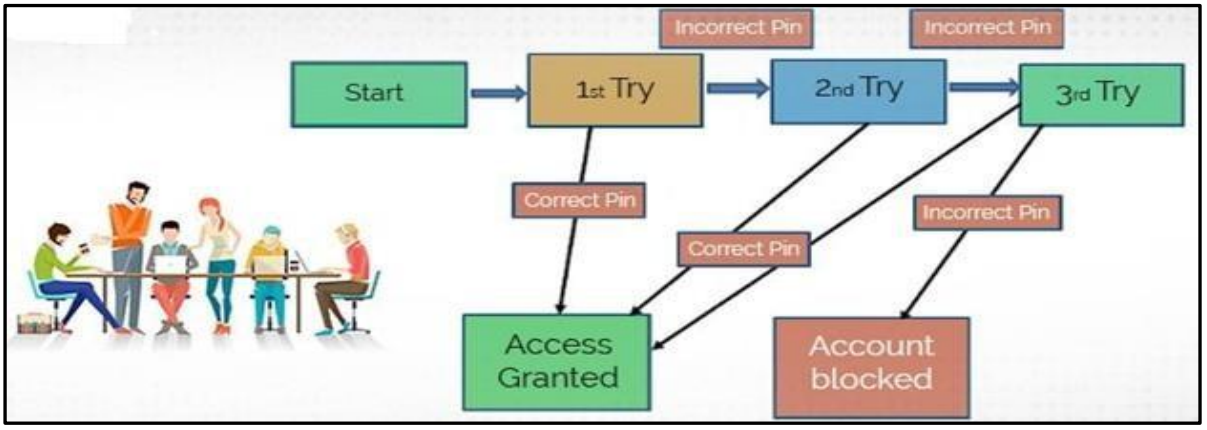
⇒ The system can be in a limited number of various states and the transitions from one state to another. It is determined by the rules of the "machine". This is the model based on the system and the tests.

⇒ The model is more elaborated as we require the advantage of the state transition technique.

Following are the basic parts of the state transition model:

⇒ The state of the software could be start, open, close, exit etc.
⇒ The transition from one state to another is allowed but not compulsory.
⇒ An event causes a transition like closing a file or withdrawing money etc.
⇒ The action results from a transition i.e an error message or access the cash.

For example, State transition diagram for **ATM machine**:



4. Decision Tables (Cause-Effect Graphing**):**

- Decision table testing is a combination of inputs that generates various outputs. It is also known as Cause-Effect table.
- It gives a standard way to test complicated business rules that are useful for testers and developers.
- The first task is to recognize a suitable function that behaves according to the combination of inputs.
- If the system contains many inputs, then combination becomes unmanageable.

Example: The condition states that if the user provides the correct username and password the user will be redirected to the homepage. If any of the input is wrong, an error message will be displayed.

Conditions	**Rule 1**	**Rule 2**	**Rule 3**	**Rule 4**
Username	F	T	F	T
Password	F	F	T	T
Output	E	E	E	H

In the above example,

T – Correct username/password

F – Wrong username/password

E – Error message is displayed

H – Home screen is displayed

LOAN APPLICATION:

Customer Name		[2 to 64 characters (A to Z, a to z, space)]
Account Number		[6 digits, 1st non zero]
Lone Amount Requested		[$500 to $9000]
Term of Loan		[1 to 30 years]
Monthly Repayment		[Minimum $10]

SUBMIT

Condition Template:

Conditions	Valid Partitions	Tag	Invalid Partitions	Tag	Valid Boundaries	Tag	Invalid Boundaries	Tag
Customer Name	2 to 64 Characters	V 1 V 2	<2 >64 Invalid Characters	X1 X2 X3	2 chars 64 chars	B1 B2	1 char 65 chars Blank	D1 D2 D3
Account Number	6 digits 1st non-Zero	V 3 V 4	<6 >6 digits 1st digit=0 Non-digit(abcd)	X4 X5 X6	100000 999999	B3 B4	99999 100000 Blank	D4 D5 D6
Loan Amount	500 – 9000	V5	<500 >9000 Blank Non-Integer	X7 X8 X9 X10	500 9000	B5 B6	499 9001	D7 D8
Term	1 – 30 years	V6	<1 >30 Blank Non-digit(abcd)	X11 X12 X13 X14	1 30	B7 B8	0 31	D9 D1 0

TEST CASES FOR LONEAPPLICATION:

	Project Code	LAP_101			
Company Logo	Project Name	Loan App			
	Prepared By	Krishna	Date Prepared		
	Reviewed By		Date Reviewed		
	Approved By		Date Approved		
	TestCase Name	Loan App	TestCase Objective	Verify objective of Loan app	
	Executed Name		Date Executed		

	Version		Build		
TC#	**TS#**	**Test Design/Test Steps**	**Test Data**	**Expected Result**	**Tags Covered**
TC_001	TS_002	Enter valid customer name, valid account number with first non-zero, valid loan amount	Name: Krishna AccNo:123456 Loan Amount:2500 Term:30 years	Term:30 years Re-Payment:79.86 Intrest Rate: 10% Total Paid: 2874.96	V1, V2, V3, V4, V5, V6
TC_002	"	Enter valid customer name, valid account number with first non-zero, valid loan amount	Name: AB AccNo:10000 Loan Amount:500 Term:1 year	Term:1 year Re-Payment:44.80 Interest Rate: 7.5% Total Paid: 537.60	B1, B3, B5, B7
	"	Enter valid customer name, valid account number	Name: Rakesh(64chars) AccNo:999999 Loan	Term:30 years Re-Payment:93.23 Intrest Rate:	

TC_003		with first non-zero,valid loan amount	Amount:9000 Term:30 years	9.7% Total Paid: 97256.36	B2, B4, B6, B8
TC_004	"	Enter valid customer name, valid account number with	Name: Rakesh(64chars) AccNo:999999 Loan Amount: abcd	Should display a popup message	X11

		first non-zero and Invalid loan amount			
TC_005	"	Enter valid customer name, valid account number with first non-zero and Invalid loan amount	Name: Althaf AccNo:10000 Loan Amount:499	Should display a popup message	X8, D7
TC_006	"	Enter valid customer name	Name: A	Should display a popup message	X1, D1
TC_007	"	Enter valid customer name	Name: ABCD….65	Should display a popup message	X2, D2
TC_008	"	Enter valid customer name	Name:123@56&#	Should display a popup message	X3
TC_009	"	Customer name is blank	Name:	Should display a popup message	D3

TC_010	"	Enter valid customer name, valid account number with first non-zero and Invalid loan amount	Name: Althaf AccNo:99999	Should display a popup message	X4, D4
TC_011	"	Enter valid customer name, Invalid account number	Name: Kiran AccNo:100000	Should display a popup message	X5, D5
TC_012	"	Enter valid customer name, Invalid	Name: Rakesh AccNo:0123456	Should display a popup message	X6

		account number			
TC_013	"	Enter valid customer name, Invalid account number	Name: Althaf AccNo: ABCDEF	Should display a popup message	X7
TC_014	"	Enter valid customer name, Account number is blank	Name: Althaf AccNo:	Should display a popup message	D6
TC_015	"	Enter valid customer name, valid account number with first non-zero and Invalid loan amount	Name: Krishna AccNo:123456 Lone Amount:90001	Should display a popup message	X9, D8
TC_016	"	Enter valid customer name, valid account number with first non-zero and loan amount is blank	Name: Kiran AccNo:123456 Lone Amount:	Should display a popup message	X10

3. White Box (Structural) Test case design techniques:

White box testing test design techniques are also referred to as Structure-based testing. White box testing is a code-based testing technique in which the internal structure is being known to the tester who is going to test the software. Here, the test cases are calculated after analysing the internal structure of the system based on code, branch, path, and condition coverage. Structure-based testing techniques use the internal structure of a software to derive test cases. They are commonly called 'white-box' or 'glass-box' techniques.

White Box Testing Techniques:

- Statement Coverage
- Decision Coverage

- branch testing,
- path testing,
- Condition Coverage
- Data flow testing
- Control flow testing
- Verifying security loops in the code
- Verifying the broken paths in the code
- Verifying the specified flow structure
- Verifying the desired output
- Verifying the condition loop to check the functionality
- Verifying each line and section

All these techniques used by white box testing as a guideline to create an error-free software.

3.Experience-Based Testing Technique:

⇒ Experience-Based testing technique requires testers to possess knowledge, skills and background. These are important to test conditions and test cases.

⇒ The experience people in both technical and business are needed, because they bring various aspects to test, analysis and design process as well as they may have an idea about which is the best way for testing and what is going wrong.

⇒ This technique is used with specification-based and structured based techniques and also used when there is no specialization or the specification is out of date.

⇒ This technique is used for low-risk system.

Types of Experience Based Testing Technique:

1. Error guessing
2. Exploratory testing

1.Error guessing:

⇒ In error guessing, experienced and good testers are required to recognize the defects in the component.

⇒ The experienced testers are able to find weaknesses of a system. So, the error guessing approach is used after more formal techniques and it is effective.

⇒ The assumption and guesses are constructed by the experienced testers. Hence it saves time in error guessing.

⇒ The success of Error Guessing technique is absolutely dependent on the skills and experience of the tester.

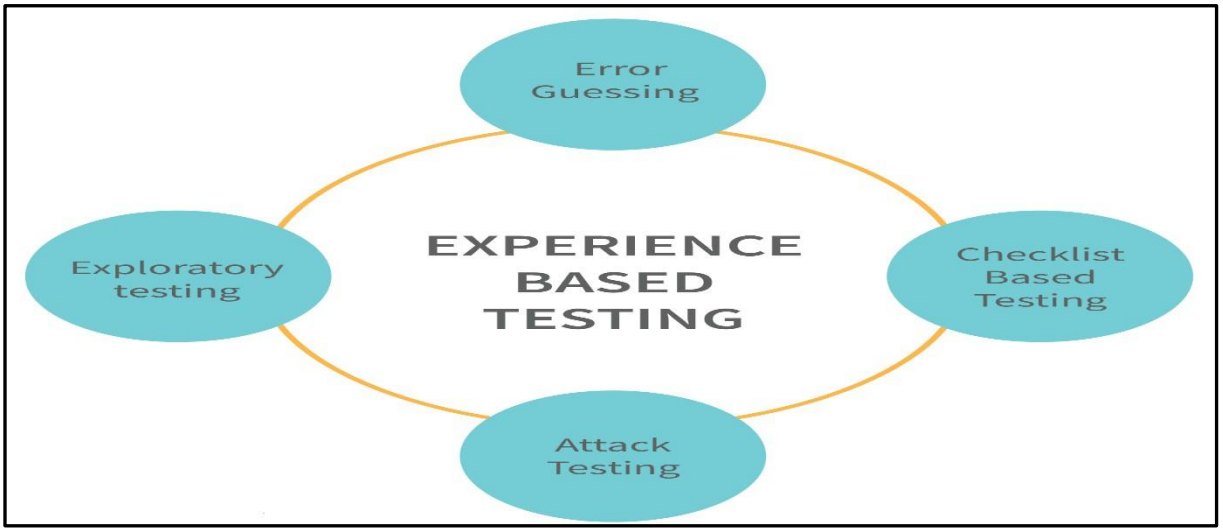
2.Exploratory testing:

⇒ Exploratory testing examines or explores the software. It finds out if software works or not.

⇒ The testers continuously make decisions about what to test next and where to spend time. This approach is useful when there is limited time and poor specifications are available.

⇒ In exploratory testing, testers are involved in minimum planning and maximum test execution.

⇒ This testing is used to check formal test process i.e., planning, test design, test execution etc. which ensures that most serious defects that are found.



Requirement Traceability Matrix or RTM:

Requirement Traceability Matrix or RTM, it's a high-level document to map and trace user requirements with test cases to ensure that for each and every requirement adequate level of testing is being achieved. The process to review all the test cases that are defined for any requirement is called Traceability. The aim of any testing project should be 100% test coverage.

In testing, the responsibility for creating and maintaining the requirements traceability matrix (RTM) typically falls on the test team or the testing lead. The testing team is responsible for identifying the test cases that are required to validate each requirement and tracking the test results in the RTM. They should also ensure that the RTM is updated regularly and that any changes to requirements are reflected in the corresponding test cases.

A Requirement Traceability Matrix (RTM) is a document that helps to ensure that all requirements defined for a system or project are met through the various stages of development, testing, and implementation. It is used to track each requirement and its progress throughout the project lifecycle.

Requirement Traceability Matrix is the means to map and trace all the client's requirements with the test cases and discovered defects. It is a single document that serves the main purpose that no test cases are missed and thus every functionality of the application is covered and tested.

Advantage of RTM:

- With the help of the RTM document, we can display the complete test execution and bugs status based on requirements.
- It is used to show the missing requirements or conflicts in documents.
- In this, we can ensure the complete test coverage, which means all the modules are tested.
- It will also consider the efforts of the testing teamwork towards reworking or reconsidering on the test cases.

Examples Of RTM:

#1) Business Requirement:

BR1: Writing emails option should be available.

Test Scenario (technical specification) for BR1

TS1: Compose mail option is provided.

Test Cases:

Test Case 1 (TS1.TC1): Compose mail option is enabled and works successfully.

Test Case 2 (TS1.TC2): Compose mail option is disabled.

#2) Defects

After executing the test cases if any defects are found that too can be listed and mapped with the business requirements, test scenarios and test cases.

For Example, If TS1.TC1 fails i.e. Compose mail option though enabled does not work properly then a defect can be logged. Suppose the defect ID auto-generated or manually assigned number is D01, then this can be mapped with BR1, TS1, and TS1.TC1 numbers.

Thus, all Requirements can be represented in a table format.

Business Requirement #	Test Scenario #	Test Case #	Defects #
BR1	TS1	TS1.TC1 TS1.TC2	BR1
BR2	TS2	TS2.TC1 TS2, TC2 TS2.TC3	D02 D03
BR3	TS3	TS1.TC1 TS2.TC1 TS3.TC1 TS3.TC2	NIL

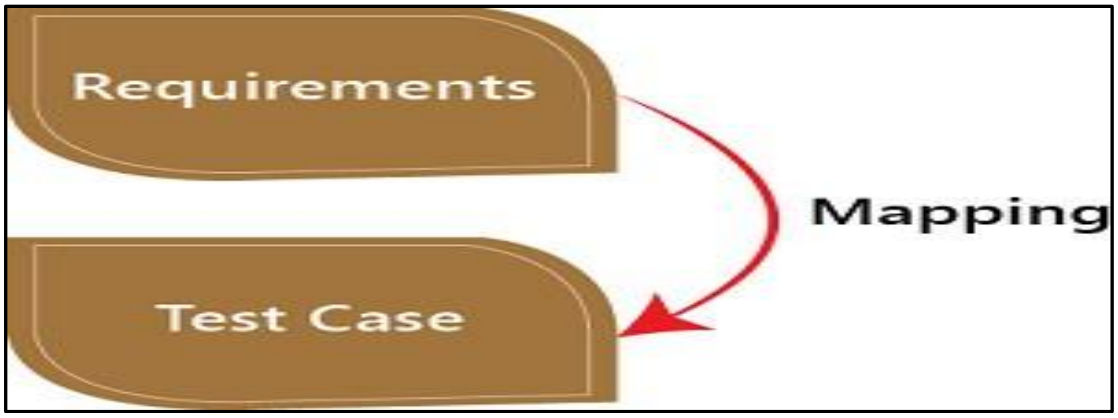
Types of Traceability Test Matrix:

The traceability matrix can be classified into three different types which are as follows:

1. Forward traceability.
2. Backward or reverse traceability.
3. Bi-directional traceability.

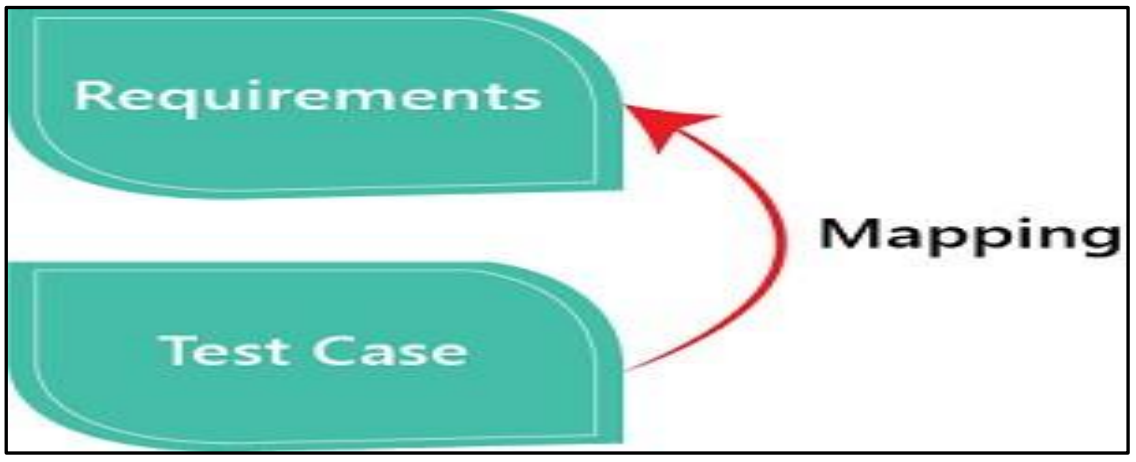
1) Forward Traceability:

The forward traceability test matrix is used to ensure that every business's needs or requirements are executed correctly in the application and also tested rigorously. The main objective of this is to verify whether the product developments are going in the right direction. In this, the requirements are mapped into the forward direction to the test cases.



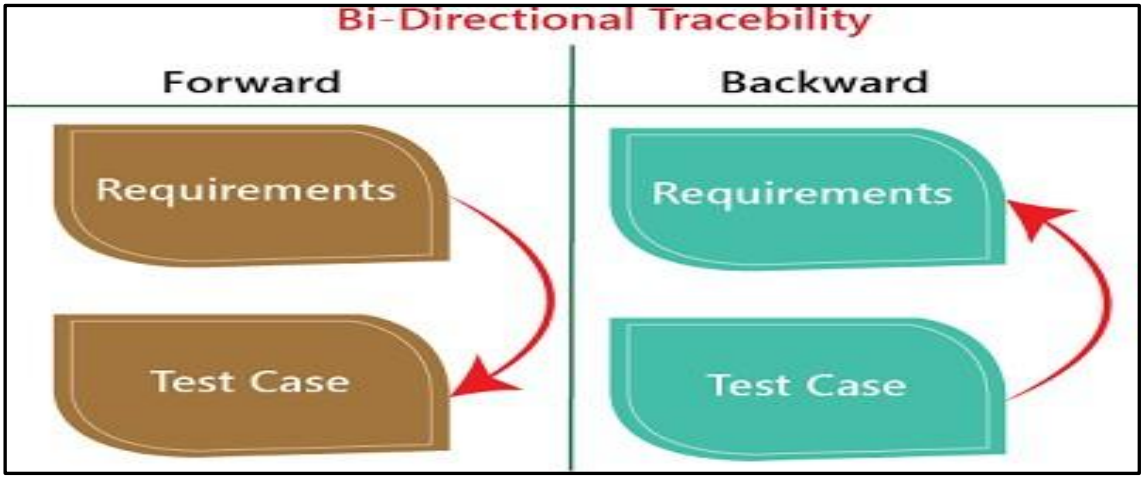
2) Backward or Reverse Traceability:

The reverse or backward traceability is used to check that we are not increasing the space of the product by enhancing the design elements, code, test other things which are not mentioned in the business needs. And the main objective of this that the existing project remains in the correct direction. In this, the requirements are mapped into the backward direction to the test cases.



3) Bi-directional Traceability:

It is a combination of forwarding and backward traceability matrix, which is used to make sure that all the business needs are executed in the test cases. It also evaluates the modification in the requirement which is occurring due to the bugs in the application.



Testing Environment:

Project: Changning the manual system to Automation system using some software's is called Project.

- Project is developed depends upon specified customer requirements.

Product:

⇒ Depends upon OVERALL requirement of the market the software's is developed is called product.

⇒ Initially there is no customer (Developing company itself is the customer) for the Product, later we will get unlimited number of customers.

Ex: Java, .Net, QTP

There are mainly three types of applications:

1. Web Applications
2. Desktop or Standalone Applications
3. System Applications

1. Web Applications:

Client and server will be different machine: Here single database server is sharable to unlimited number clients(users) via web.

Example:

- Banking Applications
- Social Networking Applications
- Health Care Applications
- Insurance Applications
- E-Commerce Applications

2. Desktop or Standalone Applications:

Client and server processing should be on single machine.

Example:

- Ms-Office
- Media player
- Antivirus

3. System Applications:

Software which makes move to run the hardware components is called system applications/products.

Example:

- Device Drivers
- Operating Systems

Guidelines and checklist for Web application testing:

Worldwide web is browsed by customers with different knowledge levels, while testing web applications(static/dynamic) testing department should concentrate on various aspects to make effective presentation of website in www.

Static=Usability and user interface.

Dynamic=Functionality.

Aspects to cover:

1.Functionality	3.User Interface	5.Security
2.Usability	4.Compatibility	6.Performance

7.Server log files

1. Functionality:

1.1. Links:

Objective is to check for all the links in the website.

- All Hyperlinks
- All Internal links.
- All External links.
- All Mail links
- Image links

1.2 Forms/screens:

- Check for the integrity of Submission of all forms.
- All field level checks.
- All field level validations
- Functionality of [Submit] [Update] [Delete] and [View]
- Handling of wrong inputs
- Default values if any (standard)
- Optional v/s Mandatory fields

1.3 Data base:

Two types of errors that may occur in web application

- Data Integrity: Missing or wrong data in table.
- Output Errors: Errors in writing, editing and reading operation in the table.

2. Usability:

How simple customer can browse the website (or) user Friendliness.

2.1 Navigation:

Navigation describes the way user navigates with in a webpage, between different user interface controls (buttons, text boxes, combo boxes, dropdown lists --- etc).

→ Application navigation is proper through tab (key board).
→ Application navigation through mouse.
→ Main features accessible from the main home page (Mother window).
→ Any hotkeys, Control keys to access menus.

2.2 Content:

Correctness is whether the information is truthful. The accuracy of the information is whether it is without grammatical or spelling errors. Remove irrelevant Information from your site this may otherwise cause mis-understanding (or) confusion.

→ Spelling and grammar.
→ Updated information (Contact details, mail IDs help reports).
→ Background / Foreground.
→ Tool tip message.

3. User Interface:

All about how exactly website looks like (View).

Basic guidelines for web user interface

→ Every dropdown box should have the first choice of NONE (it could be any other meaningful sentence such as 'Choose one or 'select'.
→ Horizontal scrolling is not preferable in general. Avoid using horizontal scroll bar ensure that the use of vertical scroll bar is judicious.
→ Illegal operations should give Popup message (message should be simple and clear).
→ Positive POPUP messages should be displayed (submitted, deleted, updated, done of cleared).
→ Avoid long scrolling drop down list make of short.

4. Client-Side Compatibility:

Check for the website compatibility with

4.1 Platforms:

- Windows (2000, XP, 2003, NT, 8.0, 8.1 10.0 and 11)
- UNIX/Linux
- MacOS (if applicable)

4.2 Browsers:

- Microsoft Edge 12.0
- Internet Explorer 7.0,8.0, 9.0, 10.0 and 11.0
- Fire Fox 14.0, 16.0, 36.0 and 48.0
- Safari 10.1 (compatible with MAC OS)
- Google Chrome 58.0

4.3 Mobile:

- Android and iOS

5. Security:

- Valid and invalid login
- Limits defined for the number of tries.
- Can it be bypassed by typing URL to a page inside directly in the browser?
- Verify log files are maintained to store the information for traceability.
- Verify encryption is done correctly is SSL is used (if applicable)
- NO access to edit scripts on the server without authorization.
- If webpage is idle for 10 to 15 seconds, session will expire
- Password Expire.

6. Performance:

6.1. Connection Speed:

→ Try with different connection speeds.

6.2 Load:

→ Will there be peak loads and how systems react.
→ Can your site handle a large number of users requesting a contain Page?

6.3. Stress:

→ Stress testing is done in order to actually break a site or Certain features to determine how the systems reacts.
→ Stress tests are designed to push and test systems limitations and determine whether the system recovers gracefully from Crashes.

7. Server log Files Testing:

For every client request to the server, server side. client information will be stored automatically is the form of logs.

Log Contains:

- Client Request.
- Server Response.
- Time of Request.

- Client IP Address.

Build:

It is an executable code of the s/w application to test and it is in URL format in my current project.

Builds (Testing URL) from Development to Testing (UAT).

Once UNIT level testing is approved, developers will integrate the limits and they will release build for testing in the form of URL's along with "Build Release Note".

The Development Lead (r) Tech Lead will update the Build and Release Note into Testing Server, he will send an email to Test Lead saying that

1. Build Ready for Testing

2. Build URL'S

3. He will mention Build Period

Build Release Note Contains:

1. Requirements implemented in this Build (Rea#)

2. URL of Build (Uniform Resource Locator)

3. Admin Username and Password.

4. DSN (Data source Name)

5. Database username and Password

6. Defect fixed in this Build (second Build onwards)

Test Execution:

Test execution is the process of executing the code and comparing the expected and actual results. The objective of this phase is real time validation of AUT before moving on to production/release. To sign off from this phase, the QA team performs different types of testing to ensure the quality of product. Along with this defect reporting and retesting is also crucial activity in this phase.

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:

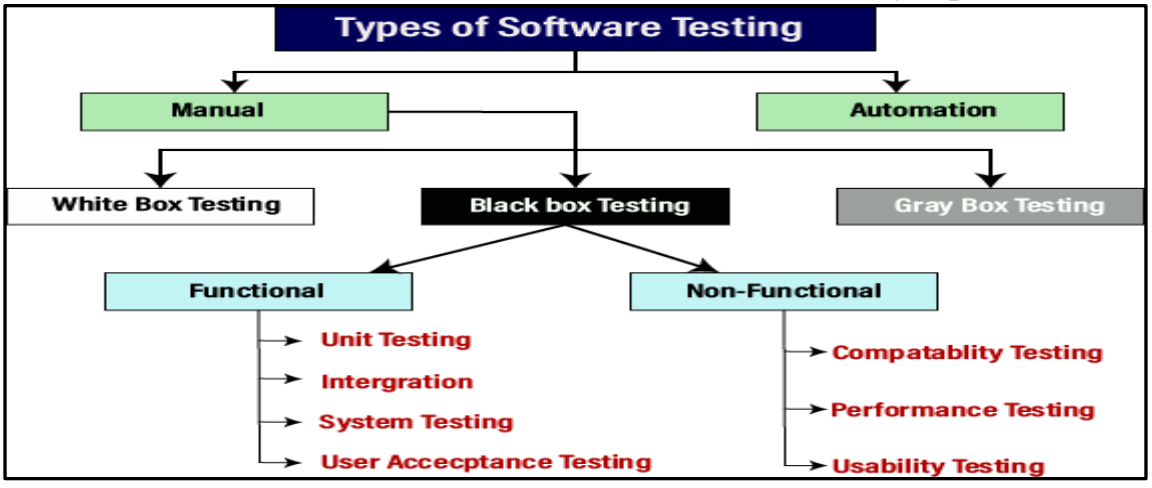
⇒ The Build being deployed to the QA environment is the most important part of
⇒ the test execution cycle.
⇒ Test execution is done in Quality Assurance (QA) environment.
⇒ Test execution happens in multiple cycles.
⇒ Test execution phase consists Executing the test cases + test scripts (if automation).

Following factors are to be considered for a test execution process:

→ Based on a risk, select a subset of test suite to be executed for this cycle.
→ Assign the test cases in each test suite to testers for execution.
→ Execute tests, report bugs, and capture test status continuously.
→ Report status, adjust assignments, and reconsider plans and priorities daily.
→ Report test cycle findings and status.

The following points need to be considered for Test Execution:

⇒ In this phase, the QA team performs actual validation of AUT (Application under test) based on prepared test cases and compares the stepwise result with the expected result.
⇒ The entry criteria of this phase are completion of the Test Plan and the Test Cases Development phase, the test data should also be ready.
⇒ The validation of Test Environment setup is always recommended through smoke testing before officially entering the test execution.
⇒ The exit criteria require the successful validation of all Test Cases; Defects should be closed or deferred; test case execution and defect summary report should be ready.



Manual Testing:

Manual testing is a software testing process in which test cases are executed manually without using any automated tool. All test cases executed by the tester manually according to the end user's perspective. It ensures whether the application is working, as mentioned in the requirement document or not. Manual Testing is one of the most fundamental testing processes as it can find both visible and hidden defects of the software. Test cases are planned and implemented to complete almost 100% of the software application. Test case reports are also generated manually. Manual testing is mandatory for every newly developed software before automated testing. This testing requires great efforts and time, but it gives the surety of bug-free software. Manual Testing requires knowledge of manual testing techniques but not of any automated testing tool.

Automation Testing:

The most significant part of Software testing is Automation testing. It uses specific tools to automate manual design test cases without any human interference. It is used to re-run the test scenarios, which were executed manually, quickly, and repeatedly. It is the most acceptable way to enhance the efficiency, productivity, and test coverage of Software testing. We cannot write the test script or perform the automation testing without understanding the programming language. In automation testing, the test automation engineer will write the test script or use the automation testing tools to execute the application. Some organizations still perform only manual testing to test the application as those companies are not fully aware of the automation testing process.

White box Testing:

White box testing is a test technique in which the internal structure or code of an application is visible and accessible to the tester. In this technique, it is easy to find loopholes in the designof an application or fault in business logic. Statement coverage and decision coverage/branch coverage are examples of white box test techniques.

Testing conducted on the source code by developers to check does the source code is working as expected or not is called white box testing.

- **What is the need of white box testing?**

As the source code is visible, finding and rectifying the problems is easy for developers.

The defects that are identified in white box testing are very economical to resolve. To reduce the defects as early as possible white box testing is helpful. To ensure 100% code coverage.

Note: White box testing is also called as glass box, structural, clear box testing.

Unit Testing:

A smallest separatable portion in the source code of the application is called unit. Unit testing is a type of software testing which is done on an individual unit or component to test its corrections. Typically, Unit testing is done by the developer at the application development

phase. Each unit in unit testing can be viewed as a method, function, procedure, or object. Developers often use test automation tools such as NUnit, Xunit, JUnit for the test execution. Unit testing is important because we can find more defects at the unit test level.

For example, there is a simple calculator application. The developer can write the unit test to check if the user can enter two numbers and get the correct sum for addition functionality.

Note: It is also called module testing or component testing.

Integration Testing:

Integration testing is a type of software testing where two or more modules of an application are logically grouped together and tested as a whole. The focus of this type of testing is to find the defect on interface, communication, and data flow among modules. Top-down or Bottom-up approach is used while integrating modules into the whole system.

This type of testing is done on integrating modules of a system or between systems. For example, a user is buying a flight ticket from any airline website. Users can see flight details and payment information while buying a ticket, but flight details and payment processing are two different systems. Integration testing should be done while integrating of airline website and payment processing system.

Black Box Testing:

It is a Software Testing method that analyses the functionality of a software/application without knowing much about the internal structure/design of the item that is being tested and compares the input value with the output value. The main focus of Black Box Testing is on the functionality of the system as a whole. The term 'Behavioural Testing' is also used for Black Box Testing. Testing is conducted on the application by test engineers or by domain experts to check whether the application is working according to the customer requirements or not. Inthis method, tester selects a function and gives input value to examine its functionality, and checks whether the function is giving expected output or not. If the function produces correct output, then it is passed in testing, otherwise failed. The primary source of black box testing is a specification of requirements that is stated by the customer.

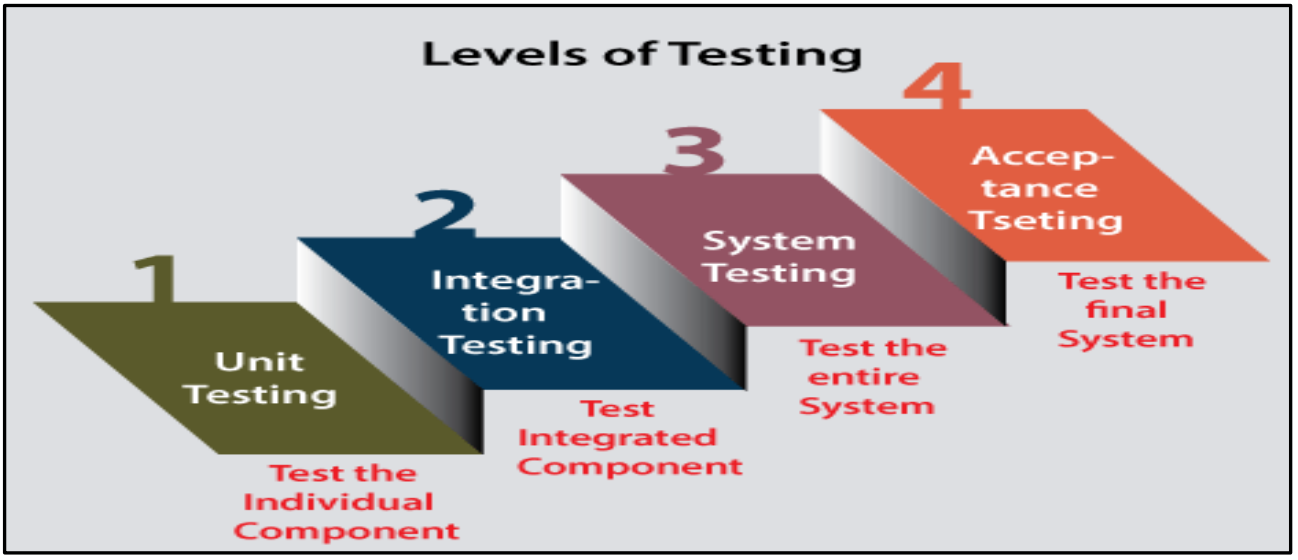
Need of Black Box Testing:

- ⇒ White box testing conducted by developer in a technical perception where as black box testing is conducted by test engineers with an end-user perception.
- ⇒ Programmers will conduct white box testing in a positive perception whereas tester will conduct black box testing with a negative perception. This will provide a chance of finding more defects
- ⇒ The more defects you identify, you will achieve more results in a quality system.
- ⇒ White box testing will not cover non-functional areas as non-functional requirements. It is very important for live production and is covered under black box testing.
- ⇒ The objective of white box testing is 100% coverage whereas the objective of black box testing is 100% customer business requirement coverage.

⇒ Black Box Testing = System Testing + User Accepting Testing. It is called a Requirement Based Testing (or) Specification Based Testing.

Software Testing Levels:

Levels of testing include different methodologies that can be used while conducting software testing. Testing levels are the procedure for finding the missing areas and avoiding overlapping and repetition between the development life cycle stages. Software testing has various levels. Each of these testing levels has a specific purpose. These testing level provide value to the software development lifecycle.



There are mainly four Levels of Testing in software testing:

1. Unit Testing
2. Integration Testing
3. System Testing
4. Acceptance Testing

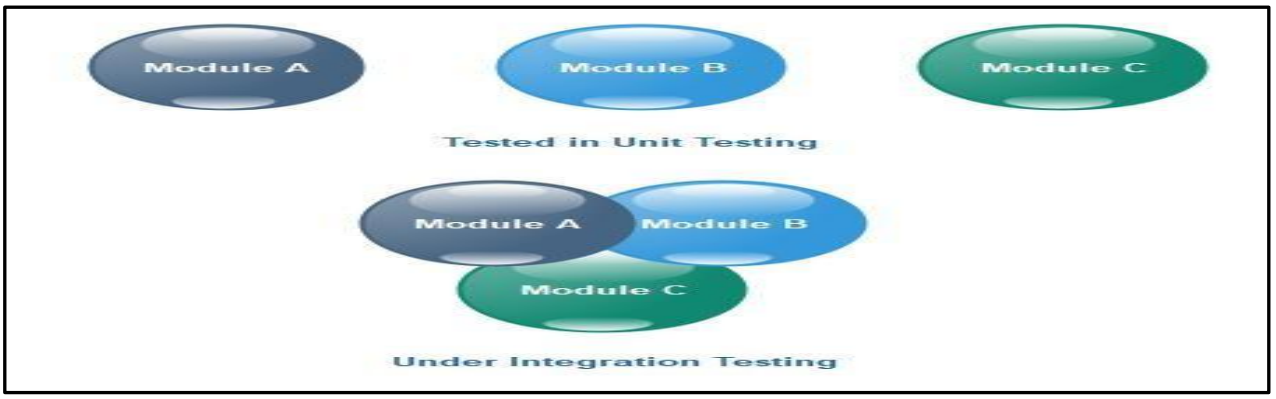
1. Unit Testing:

The smallest independent and testable part of the source code is referred to as a unit. Unit testing is the first level of software testing, which is used to test if software modules are satisfying the given requirement or not. The first level of testing involves analysing each unit or an individual component of the software application. In this type of testing, errors are detected individually from every component or unit by individually testing the components or units of software to ensure that if they are fit for use by the developers. It is the smallest testable part of the software. There are several benefits of unit testing. First of all, you get the confidence for going ahead with the integration testing only when you are sure that all units are working correctly.

Deliverable: Software unit ready for testing with other system components.

2. Integration Testing:

Once the unit testing phase is over, it is time to move on to integration testing. During integration testing the tester checks how one or more units interact with each other and produce output for various scenarios. This form of testing is carried out a software testing engineer. Unit testing confirms that various units are able to perform as per the requirement individually but integration testing confirms whether these independent units are able to perform as per expectations when integrated together.



The second level of software testing is the integration testing. The integration testing process comes after unit testing. It is mainly used to test the data flow from one module or component to other modules. In integration testing, the test engineer tests the units or separate components or modules of the software in a group. The primary purpose of executing the integration testing is to identify the defects at the interaction between integrated components or units. When each component or module works separately, we need to check the data flow between the dependent modules, and this process is known as integration testing. We only go for the integration testing when the functional testing has been completed successfully on each application module. In simple words, we can say that integration testing aims to evaluate the accuracy of communication among all the modules.

Deliverable: Portions of the system ready for testing with other portions of the system.

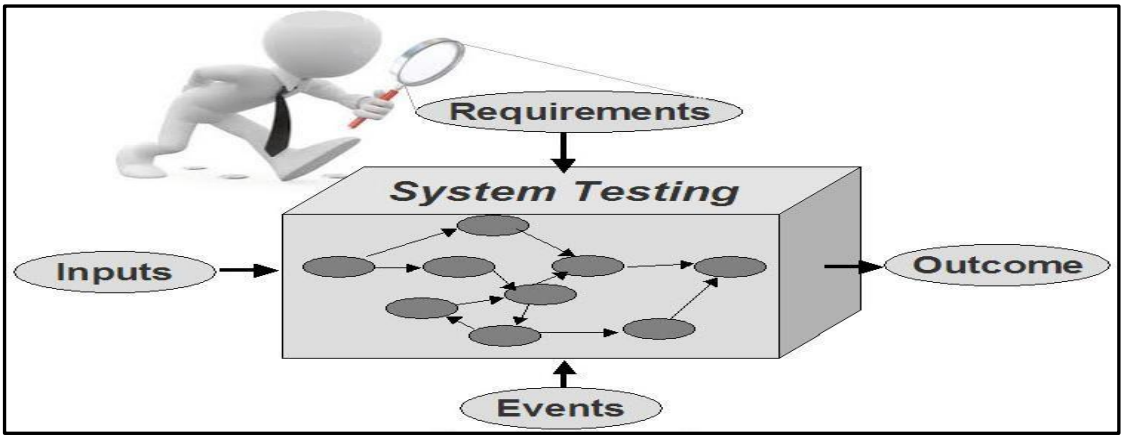
3.System Testing:

The third level of software testing is system testing, which is used to test the software's functional and non-functional requirements. It is end-to-end testing where the testing environment is parallel to the production environment. In the third level of software testing, we will test the application as a whole system. To check the end-to-end flow of an application

or the software as a user is known as System testing.

Once the integration testing phase gets successfully completed it is time to move on to system testing where the system as a whole with all components well integrated is ready for further testing. System testing is purely black box testing. The system is checked as per the requirement specifications. The testing is carried out from the user's point of view. This type of testing is carried out to check the behaviour of the application, software design and expectation of the end user. System testing validates and verifies both Application architecture and business requirements of the client.

Deliverable: Tested computer system, based on what was specified to be developed.



4. Acceptance Testing:

The last and fourth level of software testing is acceptance testing, which is used to evaluate whether a specification or the requirements are met as per its delivery. The software has passed through three testing levels (Unit Testing, Integration Testing, System Testing). The quality assurance team to come and have a look at the system and test it for quality with the help of predefined test scenarios and test cases. Some minor errors can still be identified when the end-user uses the system in the actual scenario. The acceptance testing is also known as User acceptance testing (UAT) and is done by the user or customer before accepting the final product. In sometimes stockholders also can be involved in this process.

Acceptance testing can be of following types:

a. User Acceptance testing:

This form of testing is carried out by the actual user before the software is accepted. It can be performed at the user's site or in the software organization where the software was developed.

Operation Acceptance testing: this form of testing is done to ensure that all the processes and procedures are in place so that the system can be used easily and also be maintained easily.

Contract and regulation acceptance testing: is carried out to ensure that the software is in line with all the necessary government, legal and safety standards.

Deliverable: Tested and accepted system based on the needs.

b. Alpha testing:

It is carried to ensure that the product is of good quality and also to prepare the system for beta testing. This form of testing is performed towards the end of software development where the system can be tested as a whole. This can be a long process as the testers are looking into quality as well as engineering aspects of the developed software. This form of testing is carried out by test engineers and other team members and the product is tested from the point of view of a customer.

Deliverable: Stable Application.

c. Beta testing:

Once the alpha testing is over, beta testing follows in order to improve the quality of the product and see that the product is as per the requirement of the customer. This form of testing is done a couple of days or weeks before the launch of the product. Beta testing can take a few days but may not take as much time as alpha testing as chances of defect detection are prettylow during this time. It is carried in the real-world scenario with the people who are actually going to use the product.

Deliverable: Successfully installed and running application.

Differences between alpha and beta testing:

Alpha Testing	Beta Testing
Alpha Testing performed at developer's site	Beta testing is performed at a client location or end user of the product
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.
Reliability and security testing are not checked in alpha testing.	Reliability, security and robustness are checked during beta testing.
Alpha testing requires a testing environment or a lab.	Beta testing doesn't require a testing environment or lab.

It follows the category of both white box testing and Black Box Testing.	It is only the kind of Black Box Testing.
Multiple test cycles are organized in alpha testing.	Only one or two test cycles are there in 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.

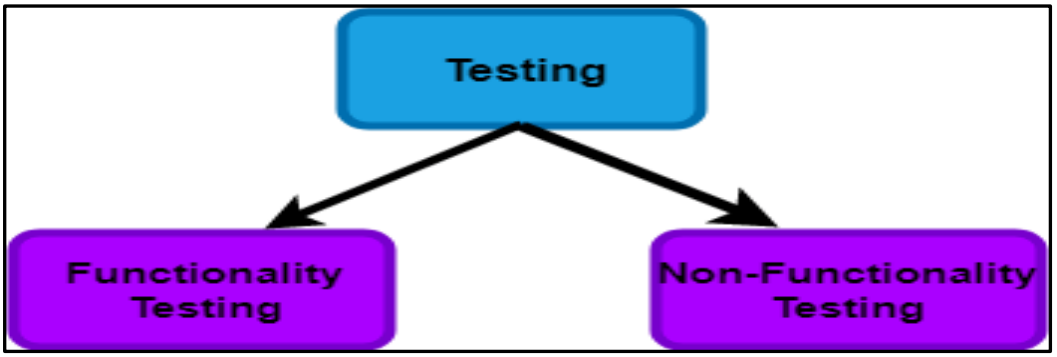
System Testing:

System Testing: Validating the functional and non-functional requirements of the system is called system testing. System testing is types of testing where tester evaluates the whole system against the specified requirements.

System testing broadly classified into two types i.e.

1) Functional system testing.

2) Non-functional system testing.



Functional Testing:

Functional system testing will be conducted both in a positive perception and also in a negative perception.

Functional Testing is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements. Manual Testing or automation tools can be used for functional testing. The testing is done by providing sample inputs, capturing resulting outputs, and verifying that actual outputs are the same as expected outputs. Functional testing mainly involves black box testing and it is not concerned about the source code of the application. This testing checks User Interface, APIs, Database, Security, Client/Server communication and other functionality of the Application Under Test.

Some functional testing examples are:

⇒ Can users successfully log in to the application once they provide legitimate credentials?
⇒ Does the payment gateway reject the input and display an error message when a user keys in an invalid credit card number?
⇒ Do inputs to the "Add New Record" screen successfully add and save a new record to the database?

Steps of Functional Testing:

A functional testing plan usually follows the below sequence:

1. Identify the Testing Goals:

Functional testing goals are the features the software is expected to have based on the project requirements. Testing goals include validating that the application works as it was intended to, and that it handles errors and unexpected scenarios gracefully.

2. Identifying Test Scenarios:

Develop a list of all possible (or at least all the most important) test scenarios for a given feature. Test scenarios describe the different ways the feature will be used. For instance, for a payment module, the test scenarios may include multiple currencies, handling invalid or expired card numbers, and generating a notification on successful transaction completion.

3. Design Test Cases:

Create test cases based on the different desired outcomes for the test inputs. For example, if you enter an invalid credit card number, the application should display a meaningful error message.

4. Preparing Test Data:

Create test data that simulates normal use conditions based on the test scenarios you identified. You could enter test data manually (e.g., from an MS-Excel spreadsheet or a printout) orautomatically via a script or test tool that reads and inputs the data from a database, flat file, XML, or spreadsheet. Each set of input data should also have associated data that describes the expected result that the input data should generate.

5. Execute the Test Cases:

Run the test cases through the application and compare actual outcomes against expected results. If actual and expected outputs are different, the feature has failed the test and a defect should be recorded.

6. Defect Reporting and Tracking:

Once a defect is identified, it should be recorded on a formal tracking system that's accessible to the entire project team. The requisite changes should be made to the application and the test case executed again to confirm resolution before a defect is marked as closed.

7. Re-Testing:

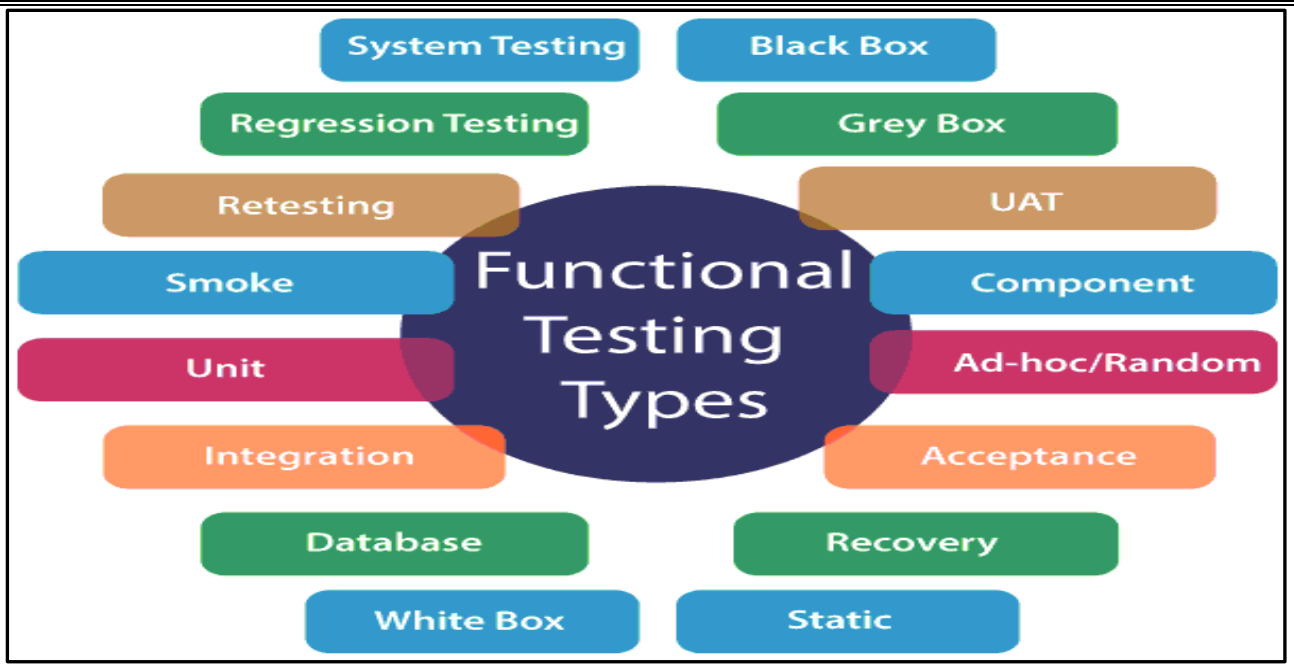
The objective of retesting is to check if the identified defects are fixed. Re-testing is executing a previously failed test against new software to check if the problem is resolved. After a defect has been fixed, re-testing is performed to check the scenario under the same environmental conditions.

8.Regression Testing:

Regression testing is testing existing software applications to make sure that a change or addition hasn't broken any existing functionality. Its purpose is to catch bugs that may have been accidentally introduced into a new build or release candidate, and to ensure that previously eradicated bugs continue to stay dead. By re-running testing scenarios that were originally scripted when known problems were first fixed, you can make sure that any new changes to an application haven't resulted in a regression, or caused components that formerly worked to fail.

Examples of Functional testing are:

1. Unit Testing
2. Smoke Testing
3. Sanity Testing
4. Integration Testing
5. White box testing
6. Black Box testing
7. User Acceptance testing
8. Regression Testing



Non-functional Testing:

Non-Functional Testing is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per non-functional parameters which are never addressed by functional testing. The primary purpose of non-functional testing is to test the reading speed of the software system as per non-functional parameters. The parameters of non-functional testing are never tested before the functional testing. Non-functional testing is also very important as functional testing because it plays a crucial role in customer satisfaction.

⇒ An excellent example of non-functional test would be to check how many people can simultaneously login into a software.

⇒ Functional and Non-functional testing both are mandatory for newly developed software. Functional testing checks the correctness of internal functions while Non-Functional testing checks the ability to work in an external environment.

⇒ Non-functional testing gives detailed knowledge of product behavior and used technologies. It helps in reducing the risk of production and associated costs of the software.

Advantages of Non-functional testing:

→ It provides a higher level of security. Security is a fundamental feature due to which system is protected from cyber-attacks.

→ It ensures the loading capability of the system so that any number of users can use it simultaneously.

→ It improves the performance of the system.

→ Test cases are never changed so do not need to write them more than once.

→ Overall time consumption is less as compared to other testing processes.

→ Non-functional testing should increase usability, efficiency, maintainability, and

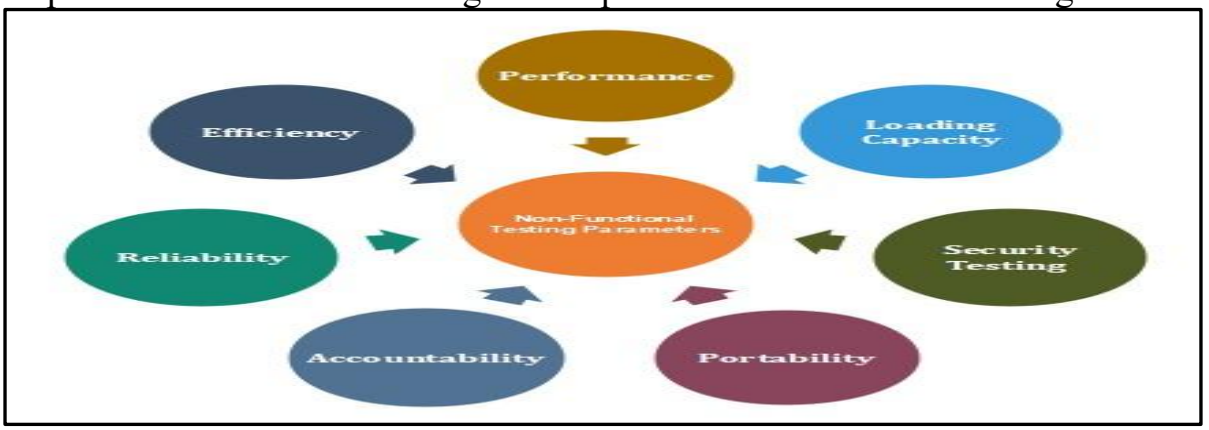
portability of the product.

→ Helps to reduce production risk and cost associated with non-functional aspects of the product.

→ Optimize the way product is installed, setup, executes, managed and monitored.

→ Collect and produce measurements, and metrics for internal research and development.

→ Improve and enhance knowledge of the product behavior and technologies in use.



Non-functional testing Parameters:

Security:

The parameter defines how a system is safeguarded against deliberate and sudden attacks from internal and external sources. This is tested via Security Testing.

Reliability:

The extent to which any software system continuously performs the specified functions without failure. This is tested by Reliability Testing

Survivability:

The parameter checks that the software system continues to function and recovers itself in case of system failure. This is checked by Recovery Testing

Availability:

The parameter determines the degree to which user can depend on the system during its operation. This is checked by Stability Testing.

Usability:

The ease with which the user can learn, operate, prepare inputs and outputs through interaction with a system. This is checked by Usability Testing

Scalability:

The term refers to the degree in which any software application can expand its processing capacity to meet an increase in demand. This is tested by Scalability Testing

Interoperability:

This non-functional parameter checks a software system interfaces with other software systems. This is checked by Interoperability Testing

Efficiency:

The extent to which any software system can handles capacity, quantity and response time.

Flexibility:

The term refers to the ease with which the application can work in different hardware and software configurations. Like minimum RAM, CPU requirements.

Portability:

The flexibility of software to transfer from its current hardware or software environment.

Reusability:

It refers to a portion of the software system that can be converted for use in another application.

Non-Functional Testing Types:

1. Performance Testing
2. Load Testing
3. Compatibility Testing
4. Usability Testing
5. Stress Testing
6. Security Testing

7. Disaster Recovery Testing
8. Portability Testing
9. Efficiency Testing
10. Reliability Testing
11. Recovery Testing
12. Localization Testing

Difference between Functional and Non-Functional Testing:

Functional Testing	**Non-Functional Testing**
Functional testing is performed using the functional specification provided by the client and verifies the system against the functional requirements.	Non-Functional testing checks the Performance, reliability, scalability and other non-functional aspects of the software system.
Functional testing is executed first	Non-functional testing should be performed after functional testing
Manual Testing or automation tools can be used for functional testing	Using tools will be effective for this testing
Business requirements are the inputs to functional testing	Performance parameters like speed, scalability are inputs to non-functional testing.
Functional testing describes what the product does	Non-functional testing describes how good the product works
Easy to do Manual Testing	Tough to do Manual Testing
Examples of Functional testing are: Unit Testing Smoke Testing Sanity Testing Integration Testing White box testing Black Box testing User Acceptance testing Regression Testing	Examples of Non-functional testing are: Performance Testing Load Testing Volume Testing Stress Testing Security Testing Installation Testing Penetration Testing Compatibility Testing Migration Testing

Functional Testing Types:

Smoke Testing:

Smoke Testing is a software testing process that determines whether the deployed software build is stable or not. Smoke testing is a confirmation for QA team to proceed with further software testing. It consists of a minimal set of tests run on each build to test software functionalities. In simple terms, smoke tests mean verifying the important features are working

and there are no showstoppers in the build that is under testing. This helps determine if the build is flawed as to make any further testing a waste of time and resources.

⇒ In the smoke testing, we only focus on the positive flow of the application and enter only valid data, not the invalid data.
⇒ In smoke testing, we verify every build is testable or not; hence it is also known as "Build Verification Testing" or "Confidence Testing."
⇒ Generally, whenever the new build is installed, we will perform one round of smoke testing because in the latest build, we may encounter the blocker bug.
⇒ After all, there might be some change that might have broken a major feature (fixing the bug of or adding a new feature could have affected a major portion of the original software), or we do smoke testing where the installation is happening.
⇒ After releasing the build to QA environment, Smoke Testing is performed by QA engineers/QA lead. Whenever there is a new build, QA team determines the major functionality in the application to perform smoke testing.

When do we do smoke testing:

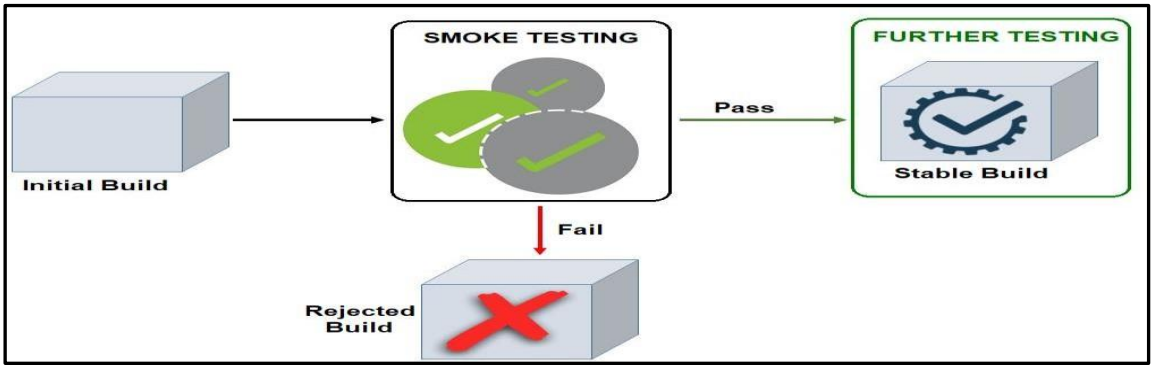
Smoke Testing is done whenever the new functionalities of software are developed and integrated with existing build that is deployed in QA/staging environment. It ensures that all critical functionalities are working correctly or not.

When the stable build is installed anywhere (Test Server, Production Server, and User Acceptance testing), we do smoke testing to find the blocker bug.

Why we do smoke testing?

⇒ We will do the smoke testing to ensure that the product is testable.
⇒ We will perform smoke testing in the beginning and detect the bugs in the basic features and send it to the development team so that the development team will have enough time to fix the bugs.
⇒ We do smoke testing to make sure that the application is installed correctly.

For example, tester is testing pet insurance website. Buying an insurance policy, adding another pet, providing quotes are all basic and critical functionality of the application. Smoke testing for this website verifies that all these functionalities are working fine before doing any in-depth testing.



Sanity Testing:

Sanity testing is performed to make sure that all the defects have been solved and no added issues come into the presence because of these modifications. It is also known as a subset of Regression testing, which ensures that the previously working software is still functional after a new change. Sanity testing also ensures that the modification in the code or functions does not affect the associated modules. Consequently, it can be applied only on connected modules that can be impacted. Sanity testing was performed when we are receiving software build (with minor code changes) from the development team. It is a checkpoint to assess if testing for the build can proceed or not. Sanity testing is performed on a system to verify that newly added functionality or bug fixes are working fine. Sanity testing is done on stable build. It is a subset of the regression test.

⇒ The execution of sanity testing makes sure that new modifications don't change the software's current functionalities.
⇒ It also validates the accuracy of the newly added features and components.

Example:

⇒ Suppose we have an e-commerce application, which contains several modules, but here, we mainly concentrate only a few modules such as the login page, the home page, the new user creation page, the user profile page, etc.
⇒ While a new user tries to login into the application, he/she is not able to log in, as there is a bug in the login page.
⇒ Because the password field in the login module accepts less than four alpha-numeric characters and based on the specification, the password field should not be accepted below 7-8 characters.
⇒ Thus, it is considered as bug, which is reported by the testing team to the development team to fix it.
⇒ Once the development team fixes the specified bug and reports back to the testing team, the testing team tests the same feature to verify that the modification that happens in the code is working fine or not.
⇒ And the testing team also verifies that the particular modification does not impact other related functionalities.
⇒ To modify the password on the user profile page there is a process.
⇒ As part of the sanity testing process, we must authenticate the login page and the profile page to confirm that the changes are working fine at both the places.

Regression Testing:

Regression testing is a black box testing technique. It is used to authenticate a code change in the software does not impact the existing functionality of the product. Regression testing is making sure that the product works fine with new functionality, bug fixes, or any change in the existing feature.

Once a defect is detected in the system it is immediately sent for fixing. However, once the defect is fixed it is important to carry out intense testing in order to check that changes made in the code has not affected any other area of the system. Regression testing is carried out to ensure that bug fixing has not caused any functionality or business logic violation. Regression testing helps in minimizing gaps in testing process. It ensures that the application has no defects before it is sent for next testing phase.

Regression testing is a type of software testing. Test cases are re-executed to check the previous functionality of the application is working fine, and the new changes have not produced any bugs.

Generally, we go for the automation whenever there are multiple releases or multiple regression cycle or there is the repetitive task.

When can we perform Regression Testing:

1. When new functionality added to the application.

2. When there is a Change Requirement.

3. When the defect fixed

4. When there is a performance issue fix

5. When there is an environment change

Regression Testing Types:

1. Unit Regression Testing [URT]: In this, we are going to test only the changed unit, not the impact area, because it may affect the components of the same module.

2. Regional Regression testing [RRT]: In this, we are going to test the modification along with the impact area or regions, are called the Regional Regression testing. Here, we are testing the impact area because if there are dependable modules, it will affect the other modules also.

3. Partial Regression testing [PRT]: Partial regression is used to ensure that the code continues to function properly even after modifications have been made to it and the unit is merged with other code that is unmodified or already exists.

4. Complete Regression Testing [CRT]: When a change in the code affects a number of modules and the effect of a change in another module is unknown, complete regression is used. The whole product is regressed to see whether there are any modifications as a result of the altered code.

Re-Testing:

Re-testing is executing a previously failed test against new software to check if the problem is resolved. After a defect has been fixed, re-testing is performed to check the scenario under the same environmental conditions. Retesting is running the previously failed test cases again on the new software to verify whether the defects posted earlier are fixed or not.

When a software flaw is discovered and patched, it's critical to ensure that the solution works. Retesting is used in this situation. It's a method that involves executing the same test cases that were used to find the issue in the first place to validate that the defect has been corrected. In terms of implementation, if the tests pass, the fix was successful.

⇒ **Example:** Say, Build 1.0 was released. While testing the Build 1.0, Test team found some defects (example, Defect Id 1.0.1 and Defect Id 1.0.2) and posted. The test team tests the defects 1.0.1 and 1.0.2 in the Build 1.1 (only if these two defects are mentioned in the Release Note of the Build 1.1) to make sure whether the defects are fixed or not.

When Do We Do Re-testing:

1. When there is a particular bug fix specified in the Release Note:

2. When a Bug is rejected:

3. When a client calls for a retesting:

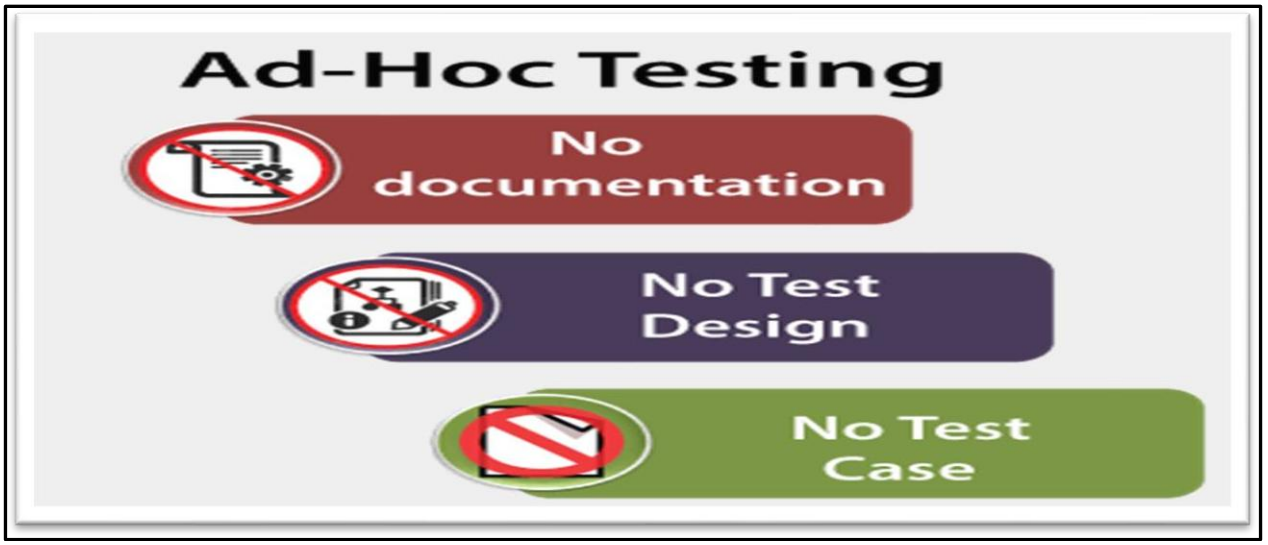
Difference between Regression Testing and Retesting:

Regression Testing	**Retesting**
Regression testing verifies that a recent change in program code has not affected the existing feature	Retesting verifies the failed test cases in the final execution especially when the defects are fixed.
Defect verification is not the part of Regression Testing	Defect verification is the part of re-testing
The purpose of Regression Testing is that new code changes should not have any side effects to existing functionalities.	Re-testing is done on the basis of the Defect fixes.
Regression testing is done for passed test cases	Retesting is done only for failed test cases
Regression testing checks for unexpected side-effects	Re-testing makes sure that the original fault has been corrected
Regression testing is known as a generic testing	Re-testing is a planned testing

Ad-hoc Testing/Monkey Testing/Gorilla Testing:

Ad hoc Testing is an informal or unstructured software testing type that aims to break the testing process in order to find possible defects or errors at an early possible stage. Ad hoc testing is done randomly and it is usually an unplanned activity which does not follow any documentation and test design techniques to create test cases. The objective of this testing is to find the defects and break the application by executing any flow of the application or any random functionality.

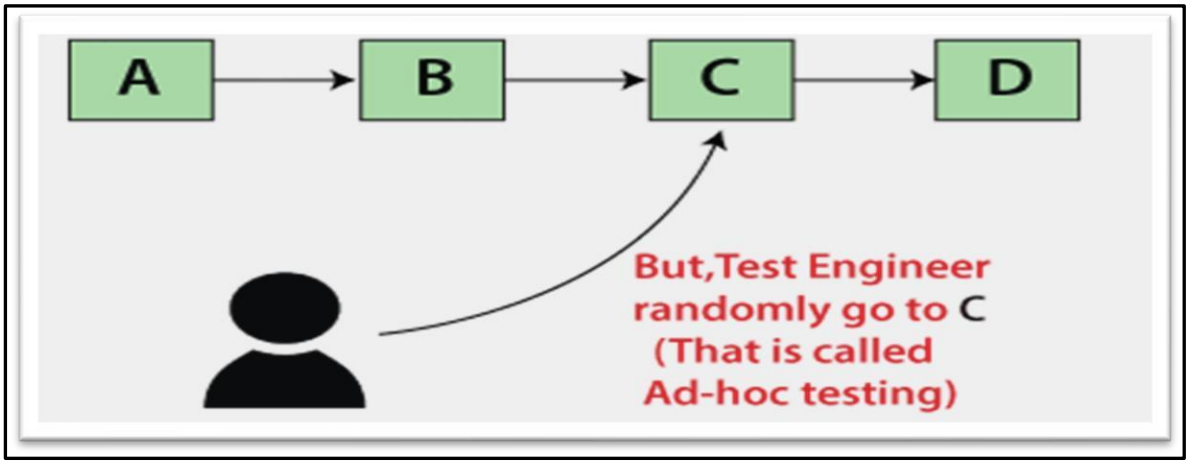
Ad-hoc testing is also known as Monkey testing and Gorilla testing.



Ad-hoc testing is an informal way of finding defects and can be performed by anyone in the project. It is difficult to identify defects without a test case, but sometimes it is possible that defects found during ad-hoc testing might not have been identified using the existing test cases.

Example:

In Ad-hoc testing, we don't follow the requirement because we randomly check the software. Our need is A ? B ? C ? D , but while performing Ad-hoc testing, the test engineer directly goes to the C and test the application as we can see in the below image:



Formal Testing:

Testing performed with a plan, documented set of test cases, etc. Test documentation can be developed from requirements, design, equivalence partitioning, domain coverage, etc. The level of formality and thoroughness of test cases will depend upon the needs of the project. Formal testing follows a systematic process called Software Testing Life Cycle (STLC). It is a mode of testing in which proper planning and documentation of the test cases are done before testing the software.

Formal Testing follows a systematic process called Software Test Life Cycle, in which we have various phases like Requirement Analysis, Test Planning, Test Case development, Test Environment Setup, Test Execution, and Test Cycle Closure.

Note: IT Software Companies prefer Formal Testing if they have proper documentation, time, and budget.

There are multiple steps taken in formal testing:

⇒ Requirement Analysis.
⇒ Planning about tests.
⇒ Documentation of Test Cases.
⇒ Setting up the Test Environment.
⇒ Execution of Tests.
⇒ Closure of Tests.

Informal Testing:

It is a mode of testing in which there is no prior planning nor there are any documented test cases of the software before testing the software. Experienced Test Engineers conduct Informal Testing using their previous experience and Experienced based Test Techniques like Exploratory testing and Error guessing etc. Informal testing relies on the intuition and skills of the individual performing the testing. Experienced engineers can be productive in this mode by mentally performing test cases for the scenarios being exercised.

Note: If no documentation, time, and budget for any project then IT Software Companies prefer Formal Testing.

Static Testing:

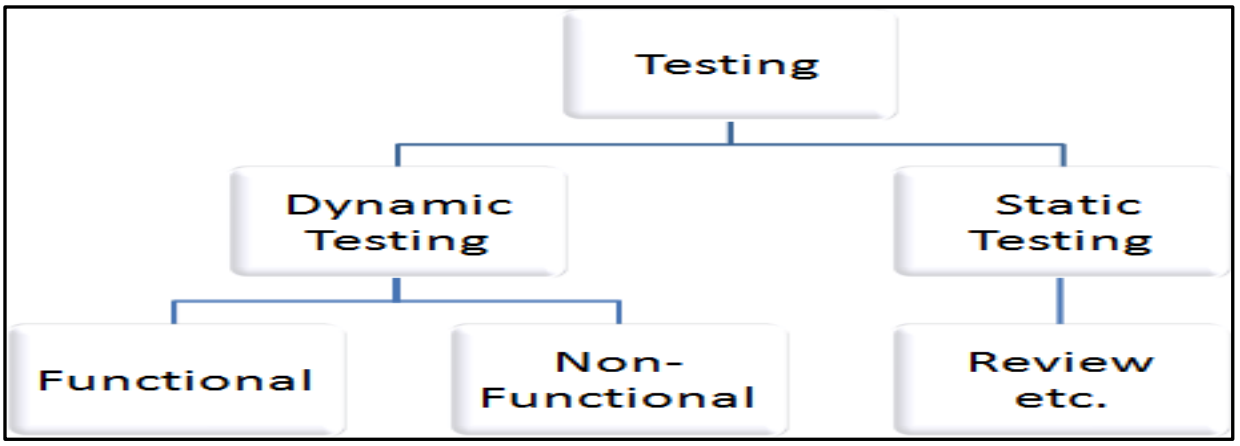
Static Testing is a type of software testing in which software application is tested without code execution. It is a verification process. Some of the essential activities are done under static testing such as business requirement review, design review, code walkthroughs, and the test documentation review. Manual or automated reviews of code, requirement documents and document design are done in order to find the errors. The main objective of static testing is to improve the quality of software applications by finding errors in early stages of software development process. The documents review, high and low-level design review, code walkthrough take place in the verification process.

Static testing involves manual or automated reviews of the documents. This review is done during an initial phase of testing to catch Defect early in STLC

Dynamic Testing:

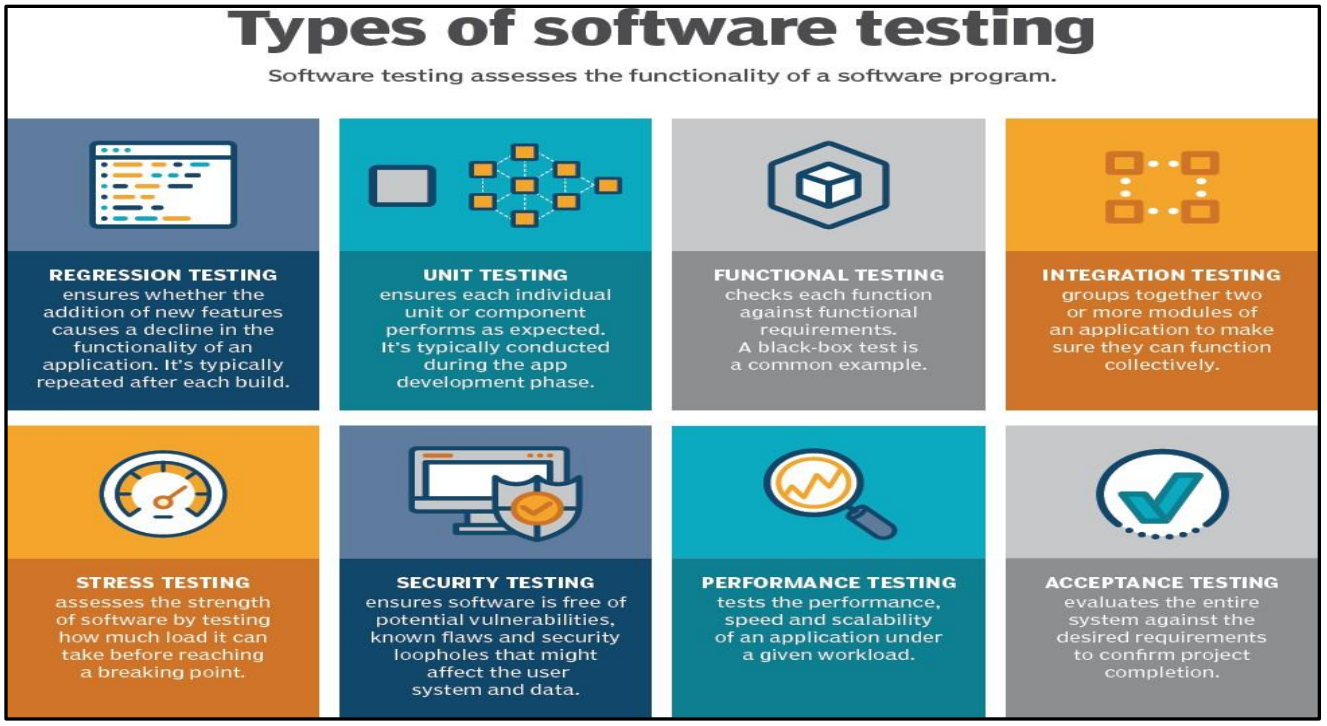
Dynamic testing is testing, which is done when the code is executed at the run time environment. Dynamic testing is performed at all levels of testing and it can be either black or white box testing. It is a validation process where functional testing [unit, integration, and system testing] and non-functional testing [user acceptance testing] are performed. Dynamic testing executes the software and validates the output with the expected outcome.

We will perform the dynamic testing to check whether the application or software is working fine during and after the installation of the application without any error.



Difference between Static testing and Dynamic Testing:

Static Testing	**Dynamic Testing**
In static testing, we will check the code or the application without executing the code.	In dynamic testing, we will check the code/application by executing the code.
Static testing does the verification process	Dynamic testing does the validation process
Static testing is about prevention of defects	Dynamic testing is about finding and fixing the defects
Static testing includes activities like code Review, Walkthrough, etc.	Dynamic testing includes activities like functional and non-functional testing such as UT (usability testing), IT (integration testing), ST (System testing) & UAT (user acceptance testing).
More reviews comments are highly recommended for good quality	More defects are highly recommended for good quality.



End-to-End Testing:

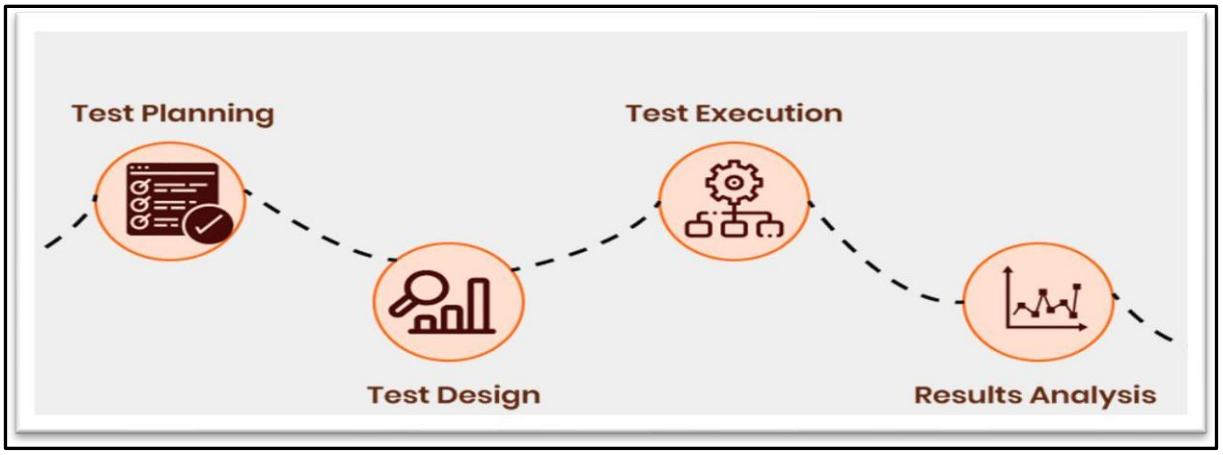
End To End Testing is a software testing method that validates entire software from starting to the end along with its integration with external interfaces. The purpose of end-to-end testing is interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate to exercise complete production like scenario.

End to End testing complements other forms of testing like unit testing, system testing, functional testing by providing additional coverage and verifying the system as a whole.

Example:

Let's consider an example of a customer using an E-Commerce Application and see how we can apply End to End Testing to this scenario

- Step #1: Log in to the application
- Step #2: View different products
- Step #3: Select an item and add it to the cart
- Step #4: Confirm and place the order
- Step #5: Perform the payment
- Step #6: Check the order details.
- Step #7: Log out of the application



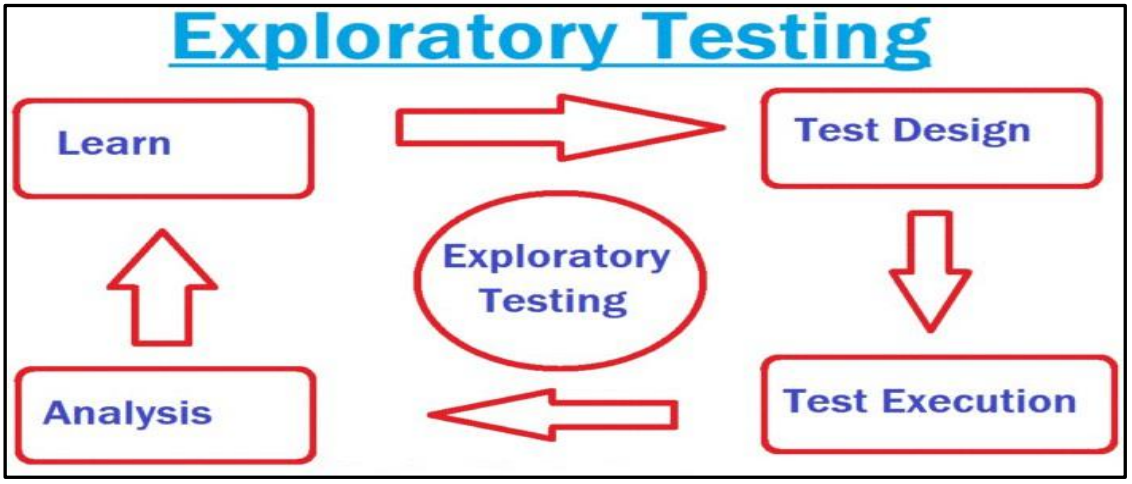
Exploratory Testing:

Exploratory Testing is informal testing performed by the testing team. The objective of this testing is to explore the application and look for defects that exist in the application. Testers use the knowledge of the business domain to test the application. Test charters are used to guide the exploratory testing. It is an unscripted approach for software testing. Exploratory testing checks the functionality and operations of the software as well as it identifies the functional and technical faults in it. It is often performed as a black box testing technique. It is also called as Un-Scripted testing

Example:

Suppose one company has a 15-year-old project, and they hired a new test engineer now. The new test engineer faces many difficulties in understanding every scenario or requirement from scratch or starting because he/she is new with the application.

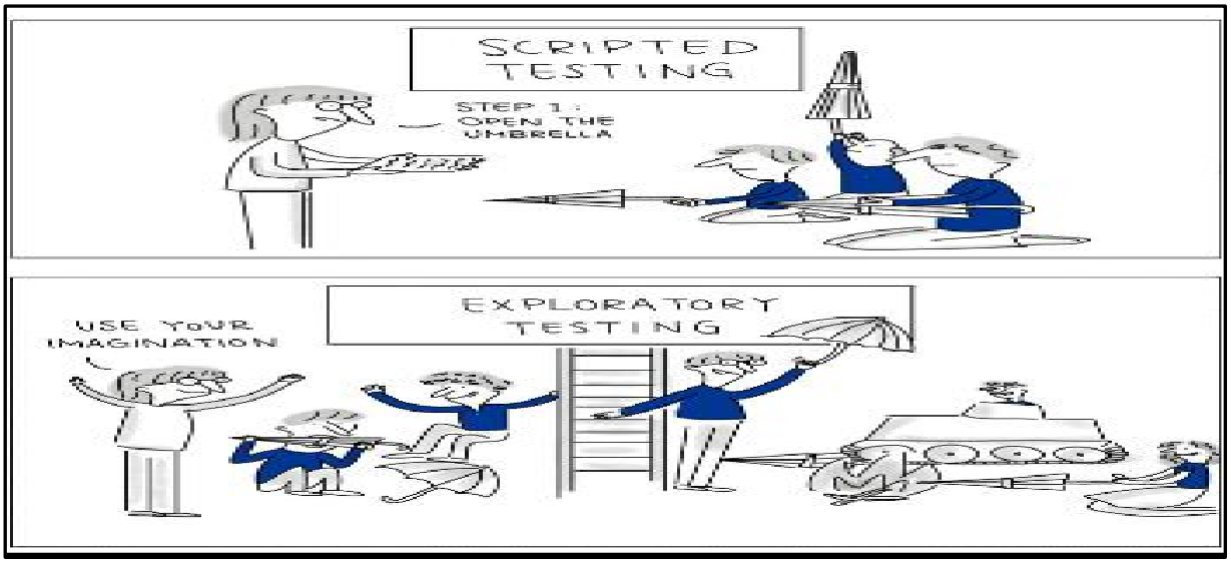
So firstly, he/she will take the application and start exploring the application. Once the test engineer starts using the application, he/she will get to know how the application is working. And, this process is nothing but exploratory testing



Scripted Testing:

Scripted testing is an approach where the testing team sticks to the script that consists of test cases and detailed steps. The testing team follows these written test cases and executes them and tries to find any bugs in the system.

In scripted testing, the testing team prepares various documents such as test plan, test case specification, and many others based on the requirements specification. The main goal of scripted testing is to keep the necessary documents ready before the testing process. The testing team should have instructions ready to be followed during the process.



Difference between Scripted Testing and Exploratory Testing:

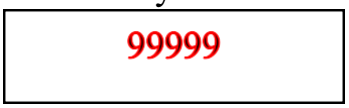
Scripted Testing	Exploratory Testing
Before starting the testing, the testing team prepares the documents required for the testing process such as test plan, test cases.	The testing team does not need any test documents to start the testing.
For scripted testing, clear and complete requirements specification is a must.	While conducting exploratory testing, testers can proceed further even without incomplete or no documentation.
Reproducing the bugs is easy as testers follow steps to test any scenario.	Reproducing the bugs may not be possible as testers do not have any written steps to follow.
A new tester can easily start testing the system as s/he has a set of steps that needs to be followed.	For a new tester, getting familiar with the system may take time. Especially when any complex applications are being developed.
Is about Controlling tests	Is about Improvement of test design
The script is in control	The tester's mind is in control
Determination of test cases well in advance	Determination of test cases during 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



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.

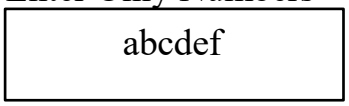
To do positive testing, set the valid input values from 0 to 99999 and check whether the system is accepting the values.

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



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.

Positive V/S Negative Test Cases:

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 upto 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.

Globalization and Localization 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.

Non-Functional Testing Types:

Non-Functional Testing:

Non-Functional Testing is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It focuses on the software's performance, usability, and scalability. In Non-Functional Testing to be tested to improve the quality and performance of the application. It is designed to test the readiness of a system as per non-functional parameters which are never addressed by functional testing. The primary purpose of non-functional testing is to test the reading speed of the software system as per non-functional parameters. The parameters of non-functional testing are never tested before the functional testing. Non-functional testing is also very important as functional testing because it plays a crucial role in customer satisfaction.

1. UI Testing:

UI testing is a type of software testing that specifically focuses on the graphical user interface (GUI) of a software application.

The primary goal of UI testing is to ensure that the interface is user-friendly, functions correctly, and meets the requirements and expectations of end-users.

During UI testing, testers evaluate the various components of the GUI, such as menus, buttons, forms, input fields, icons, and graphics, to ensure that they work as expected. The testing may also involve verifying the application's layout, navigation, and responsiveness across different devices, platforms, and screen sizes.

UI testing can be performed manually or automated, depending on the complexity and scale of the application. Manual UI testing involves human testers manually performing the testing process, while automated UI testing uses specialized tools and scripts to perform the tests automatically.

Ultimately, the goal of UI testing is to improve the quality of the software application and provide end-users with an enjoyable and efficient experience while using the application.

❖ **UI Testing, also known as GUI (Graphical User Interface) Testing.**

In the computer application, we have mainly two types of interfaces such as:

1. Command Line Interface (CLI)
2. Graphical User Interface (GUI)

The **Command-line interface** is used when we need to type text, and at the same time computer responds to that command.

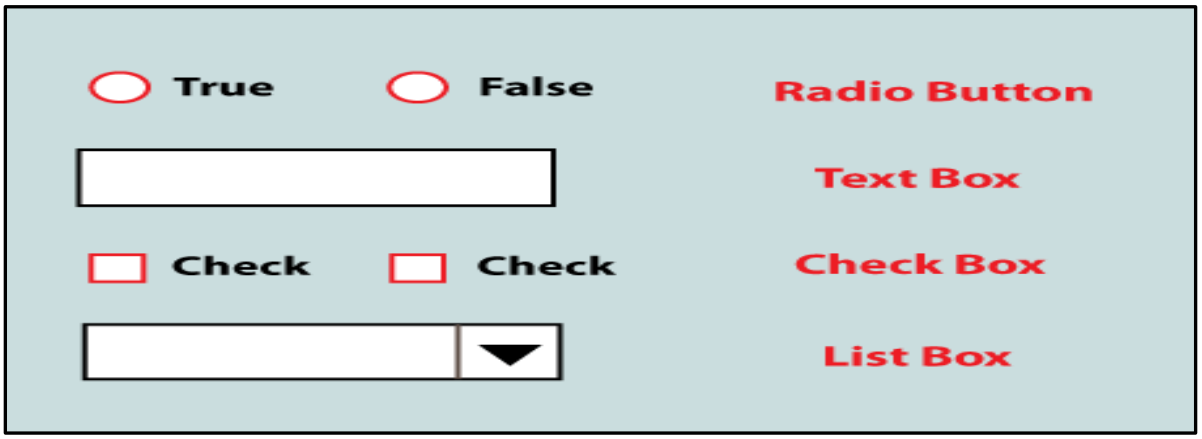
On the other hand, the **Graphical user interface** is used to interrelate along with the computer by using the images rather than text.

Usually, the GUI testing is used to assesses a design of elements or features like:

- Text boxes
- Font size
- Font color
- Buttons
- Menus
- Links
- Layout
- Labels
- Text Formatting
- Lists
- Captions
- Icons
- Content

We have some of the critical GUI elements that can be used for communication between the user and application.

- Check Box
- List Box
- Radio button
- Text Box



2. Usability Testing:

Usability Testing also known as User Experience (UX) Testing, is a testing method for measuring how easy and user-friendly a software application is. It is a process that measures the usability, efficiency, and effectiveness of the software application, with the goal of identifying and fixing any usability issues that may affect the user experience.

Usability testing is instead focused on the end-user, on how easily he was able to use the interface and if the design of the interface was friendly enough.

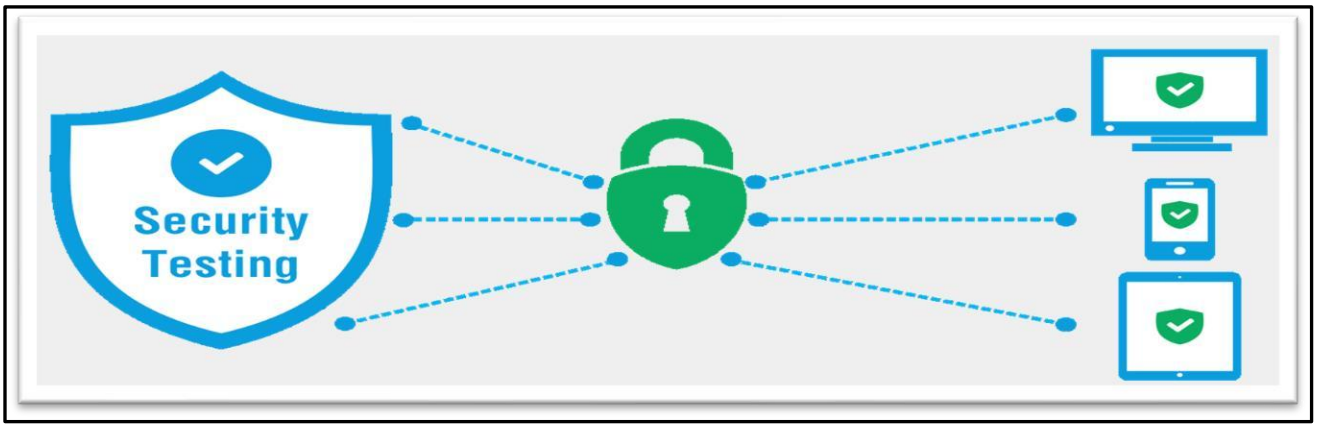
When software is made-ready, it is important to make sure that the user experience with the product should be seamless. It should be easy to navigate and all the functions would be working properly, else the competitor's website will win the race. Therefore, usability testing is performed. The objective of usability testing is to understand customers' needs and requirements and also how users interact with the product (software). With the test, all the features, functions, and purposes of the software are checked.

The primary goals of usability testing are – discovering problems (hidden issues) and opportunities, comparing benchmarks, and comparison against other websites. The parameters tested during usability testing are efficiency, effectiveness, and satisfaction. It should be performed before any new design is made.

3. Security Testing:

Security Testing is a type of Software Testing that uncovers vulnerabilities, threats, risks in a software application and prevents malicious attacks from intruders. The purpose of Security Tests is to identify all possible loopholes and weaknesses of the software system which might result in a loss of information, revenue, repute at the hands of the employees or outsiders of the Organization.

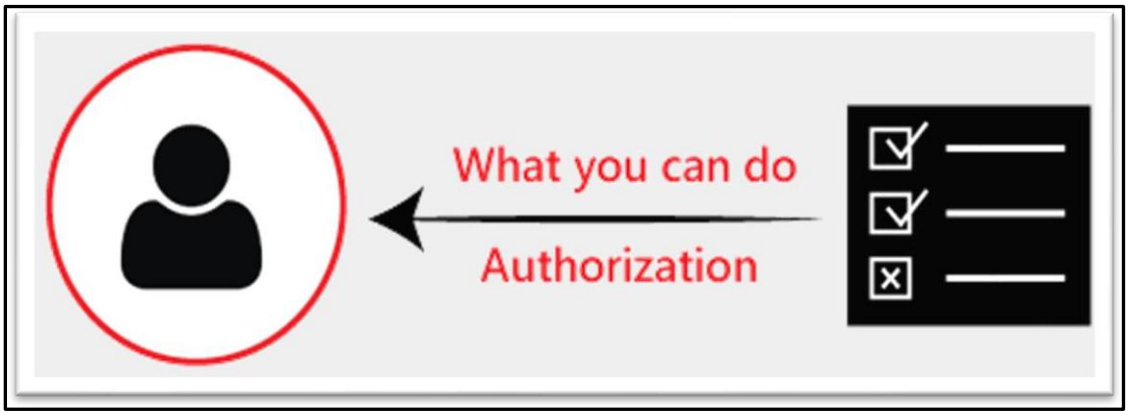
The primary objective of security testing is to find all the potential ambiguities and vulnerabilities of the application so that the software does not stop working. If we perform security testing, then it helps us to identify all the possible security threats and also help the programmer to fix those errors.



Principle of Security testing:

→ Availability → Authorization → Authentication
→ Integrity → Confidentiality → Non-repudiation

- ✓ **Availability:** In this, the data must be retained by an official person, and they also guarantee that the data and statement services will be ready to use whenever we need it.
- ✓ **Integrity:** In this, we will secure those data which have been changed by the unofficial person. The primary objective of integrity is to permit the receiver to control the data that is given by the system. The integrity systems regularly use some of the similar fundamental approaches as confidentiality structures. Still, they generally include the data for the communication to create the source of an algorithmic check rather than encrypting all of the communication. And also verify that correct data is conveyed from one application to another.
- ✓ **Authorization:** It is the process of defining that a client is permitted to perform an action and also receive the services. The example of authorization is Access control.



- ✓ **Confidentiality:** It is a security process that protracts the leak of the data from the outsider's because it is the only way where we can make sure the security of our data.
- ✓ **Authentication:** The authentication process comprises confirming the individuality of a person, tracing the source of a product that is necessary to allow access to the private information or the system.

- ✓ **Non- repudiation:** It is used as a reference to the digital security, and it a way of assurance that the sender of a message cannot disagree with having sent the message and that the recipient cannot repudiate having received the message.

4. Compatibility Testing:

Compatibility testing is a type of software testing that is performed to ensure that a software application or system is compatible with different hardware, software, operating systems, and network configurations. The goal of compatibility testing is to ensure that the application or system can work as intended on various platforms without any issues.This testing is done only when the application becomes stable. Means simply this compatibility test aims to check the developed software application functionality on various software, hardware platforms, network and browser etc.

Types of Compatibility Testing:

Software:

- Testing the compatibility of an application with an Operating System like Linux, Mac, Windows
- Testing compatibility on Database like Oracle SQL server, MongoDB server.
- Testing compatibility on different devices like in mobile phones, computers.

Hardware: Checking compatibility with a particular size of

- RAM
- ROM
- Hard Disk
- Memory Cards
- Processor
- Graphics Card

Smartphones: Checking compatibility with different mobile platforms like android, iOS etc.

Network : Checking compatibility with different :

- Bandwidth
- Operating speed
- Capacity

5. Load Testing:

Load testing is a type of software testing that is performed to evaluate the behavior of a system or application under normal and peak load conditions.Here, load means that when N-number of users using the application simultaneously or sending the request to the server at a time. The goal of load testing is to determine the system's response time, throughput, and resource utilization when it is subjected to a high load, such as a large number of users or a high volume of transactions.Load testing determines the behavior of the application when multiple users use it at the same time. It is the response of the system measured under varying load conditions. The load testing is carried out for normal and extreme load conditions.The load testing is mainly used to test the Client/Server's performance and applications that are web-based.

Some common load testing techniques include:

a) Stress testing: Testing the system's ability to handle a high load above normal usage levels
b) Spike testing: Testing the system's ability to handle sudden spikes in traffic
c) Soak testing: Testing the system's ability to handle a sustained load over a prolonged period of time

6. Performance Testing:

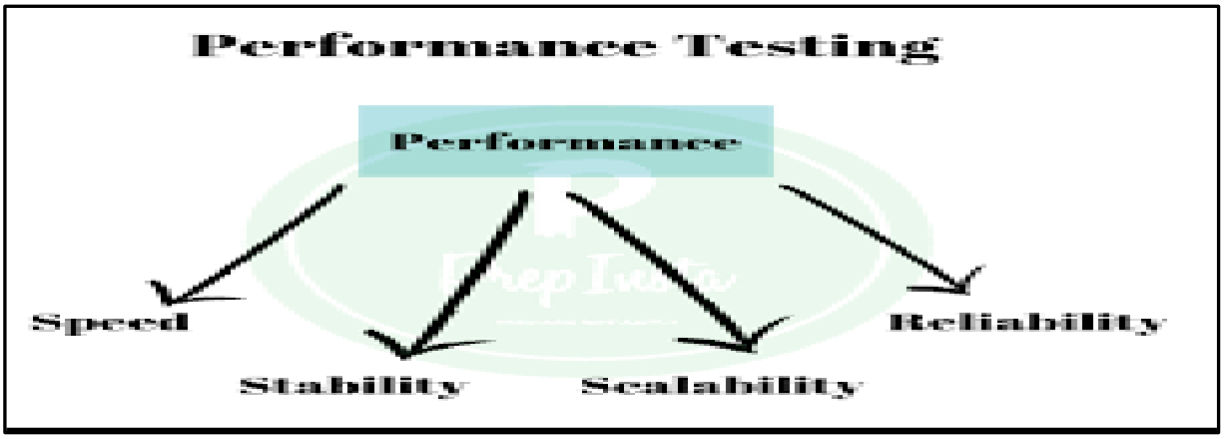
"Checking the behavior of an application by applying some load is known as performance testing".

Performance testing is a type of software testing that is performed to evaluate the speed, stability, scalability, and responsiveness of a system or application under different workload conditions. The goal of performance testing is to identify any performance-related issues and ensure that the system or application meets its performance requirements. The main objective of performance testing is to identify performance-related issues such as response time, throughput, resource utilization, and stability of the system or application. Performance testing is used to ensure that the system or application meets its performance requirements and can handle the expected workload

without any performance-related issues or failures.

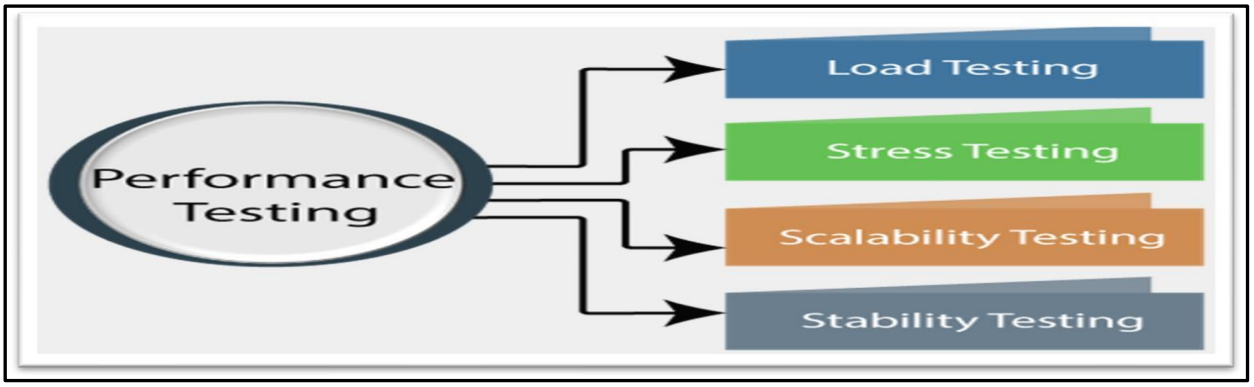
Performance Testing Attributes:

- Speed: It determines whether the software product responds rapidly.
- Scalability: It determines amount of load the software product can handle at a time.
- Stability: It determines whether the software product is stable in case of varying workloads.
- Reliability: It determines whether the software product is secure or not.



Types of Performance Testing:

1. Load testing
2. Stress testing
3. Scalability testing
4. Stability testing



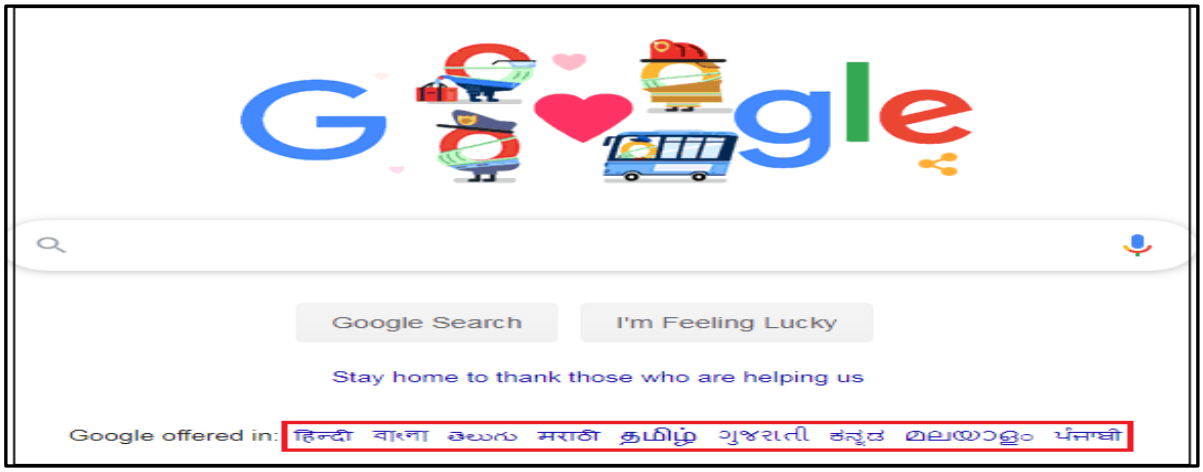
7. Globalization Testing:

Globalization testing is the process of evaluating a software application or product to ensure that it can function in different countries and regions with diverse languages, cultural preferences, and regulatory requirements. This type of testing ensures that the application is compatible with various locales, character sets, and time zones.

The goal of globalization testing is to verify that the software product can adapt to the cultural and language differences of various regions, and that it meets the legal and regulatory requirements of the target markets. This type of testing includes functional

testing, linguistic testing, and usability testing.

For example, In India, the Google.com supports most of the languages, and it can also be retrieved by the large numbers of people of the various countries as it is a globalized application.



Objective / Purpose of Globalization Testing:

- It is used to make sure that the application is to support all the languages around the world.
- This testing will focus on the world-wide experiences of the application.
- It is used to make sure that code can control all international support without breaking the functionality of the application.
- To ensure the use of software application all over the world.

Types of Globalization Testing:

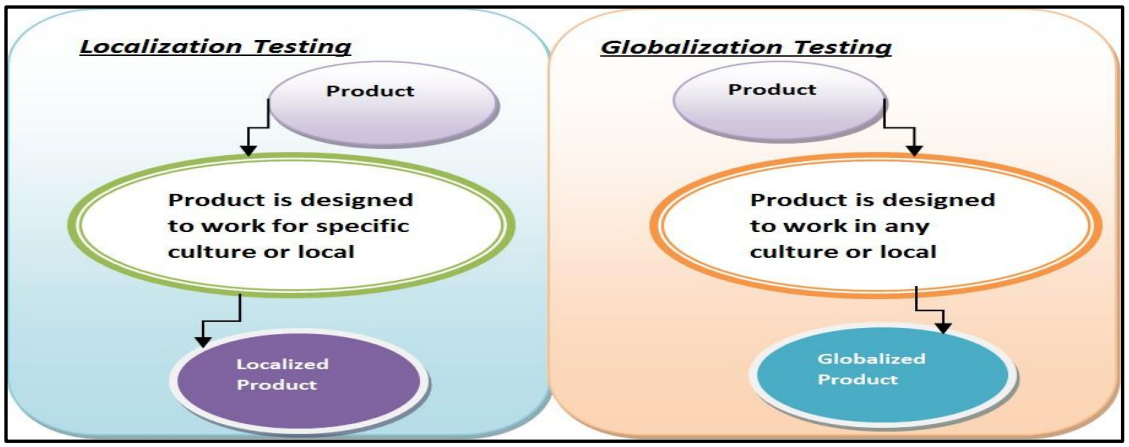
1. **Internationalization Testing:**

   Internationalization testing is an important aspect of software testing that aims to ensure that a software application or product is designed and developed in a way that makes it easy to adapt to different languages, cultures, and regions. This type of testing is performed to identify and remove any cultural or language-specific dependencies in the software product, so that it can be easily localized and customized for different markets.

   Internationalization testing involves various testing techniques such as functional testing, usability testing, and compatibility testing to verify that the application can handle different character sets, date and time formats, and other cultural differences that may impact the application's functionality. This testing also ensures that the application is compliant with local regulations, laws, and standards of different regions. Internationalization testing plays a crucial role in the success of a software product in the global market, and it is essential to ensure that the product is flexible, adaptable, and compatible with different languages, cultures, and regions.

2. **Localization Testing:**

Localization Testing is a software testing technique in which the behavior of a software is tested for a specific region, locale or culture. The purpose of doing localization testingfor a software is to test appropriate linguistic and cultural aspects for a particular locale. It is the process of customizing the software as per the targeted language and country. It is a process of testing a globalized application whose UI, default language,

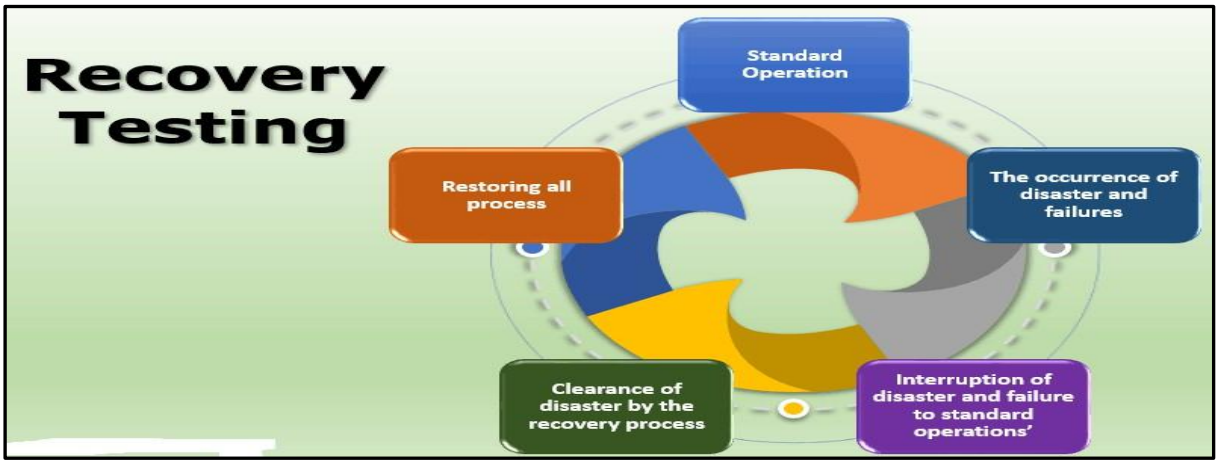


currency, date, time format, and documentation are designed as per the targeted country or region. It ensures that the application is capable enough for using in that particular country.

8. Recovery Testing:

Recovery testing is a type of software testing that aims to verify the application's ability to recover from various types of failures, errors, or disruptions. This testing is performed to ensure that the application can continue to function properly even after encountering unexpected problems, such as system crashes, power failures, network outages, or database corruption.

The goal of recovery testing is to identify any weaknesses or vulnerabilities in the application's recovery mechanisms, and to ensure that the application can restore its normal operation in a timely and efficient manner. Recovery testing involves simulating different types of failure scenarios and observing how the application responds to them. During recovery testing, the tester may deliberately induce failures such as disconnecting the network cable, killing a process, or forcing a system reboot to evaluate the application's ability to recover from such events. The testing may also include testing the backup and restore mechanisms to ensure that the application can recover data and configuration settings.



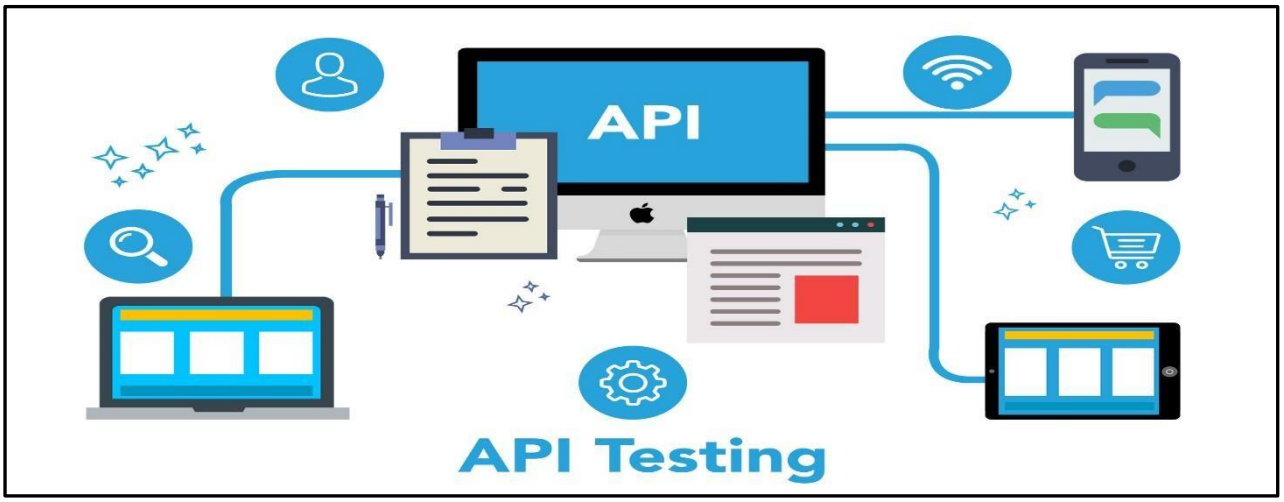
A system or software should be recovery tested for failures like:

- Power supply failure
- The external server is unreachable
- Wireless network signal loss
- Physical conditions
- The external device not responding
- The external device is not responding as expected, etc.

API Testing:

An API (Application Programming Interface) test involves testing API directly and as part of an integration test to ensure they meet reliable, functional, performance, and security requirements. Due to the lack of a GUI, APIs are tested at the message layer. The tester writes code that makes an API request to the server that provides the API, provides the required inputs, collects the output from the response, and matches the actual output with the expected output.

API testing primarily concerns the business logic of the software that's exposing the API. It does not involve the look and feel, accessibility, or usability of the software. API testing can be automated to make it repeatable and reproducible each time they run.



Defect:

The Problem which is identified by the Test Engineer Machine is called a DEFECT or a BUG or Issue. A defect is an anomaly which causes a deviation between the expected and actual results.

A defect is a condition in software which does not meet a software requirement or user expectations. Defects are used to identify differences between what is expected and what is actually produced. A Defect, in simple terms, is a flaw or an error in an application that is restricting the normal flow of an application by mismatching the expected behaviour of an application with the actual one. The defect occurs when any mistake is made by a developer during the designing or building of an application and when this flaw is found by a tester, it is termed as a defect. It is the responsibility of a tester to do thorough testing of an application to find as many defects as possible to ensure that a quality product will reach the customer. It isimportant to understand the defect life cycle before moving to the workflow and different states of the defect.

“Defect Reporting, Defect Tracking, and Status Tracking is called Defect Management”.

Defect Report Template

In most companies, a defect tracking tool is used and the elements of a defect report can vary from one tool to the other. However, in general, a defect report can consist of the following elements.

ID: Unique identifier given to the defect. (Usually, automated).

Name: Name of the Project or Product.

Tittle: Tittle of the Defect.

[Ex: SBI 1. 0V_B0l_ Krishna_ Registration_ [Reset] is not working

Category: Category of the defect.

[Project Bug or Product Bug or Question or Enhancement or Other's].

Status: The status of the defect.

[NEW, OPEN, FIXED, FIX VERIFIED, CLOSED, RE-OPEN, NOT A BUG, DUPLICATE, NEED MORE INFO, CAN’T REPRODUCE and WILL NOT FIX]

Defect Severity: Severity of the Defect.

[CRITICAL, MAJOR, MODERATE, MINIMAL]

Defect Priority: Priority of the Defect.

[P1, P2, P3, P4]

Module: Specific module of the product where the defect was detected.

[Customer, Admin, Inventory, Purchase, Manufacturing and Other’s]

Reported By or Originator Name: The name of the person who reported the defect.

[Test Engineer Name or Test Lead Name]

Assigned To: The name of the person that is assigned to analyze/ fix the defect.

[Developer Name or Development Lead Name]

Browser: Name of the Browser.

[All Browsers, ex: Edge 100.0, Chrome 103.0, Mozilla FireFox 90.0, Safari 10.03 and Other's]

Release Version: Release version of the product. (Ex: v1).

Found in Version: Version of the product where the defect was detected (ex: v1.2.3.5).

Detected in Build or Found in Build: Build of the product where the defect was detected (ex: 1.2.3.5).

Fixed in Version: Version of the product where the defect was fixed (ex: v1)

Fixed Build: Build of the product where the defect was fixed (ex:1.2.3.9),

Summary: Summary of the defect. Keep this clear and concise.

Description: Detailed description of the defect. Describe as much as possible but without repeating anything or using complex words. Keep it simple but comprehensive.

Steps to Reproduce: Step by step description of the way to reproduce the defect. Number the steps.

Actual Result: The actual result you received when you followed the steps.

Expected Results: The expected results.

Attachments: Attach any additional information like screenshots and logs.

Remarks: Any additional comments on the defect.

DEFECT TEMPLATE			
Category :		Last Changed :	
Title :			
Status :		Browser :	
Severity :		Found in Version :	
Priority :		Found in Build :	
Module :		Fixed in Version :	
Originator :		Fixed in Build :	
Assigned to :		Duplicate ID :	
Steps To Reproduce :			
Actual Result :			
Expected Result :			
Comment :			

Note: Defect Reporting Template vary from one company to another.

If we use Tool for Defect management, every tool provides their own template.

Defect Reporting Process:

The defect Reporting Process varies from one company to another.

a) Small scale Company:

⇒ Tester -> Developer.

b) Large scale Company:

⇒ Tester -> Test Lead -> Development Lead -> Developer.

Defect Age: The time gap between date of detection & date of closure I known as the defect age.

In other words, how long, a bug is present in the development life cycle is called defect age.

Defect Life Cycle:

The Defect Life Cycle, also known as the Bug Life Cycle, is a cycle of defects from which it goes through covering the different states in its entire life. This starts as soon as any new defect is found by a tester and comes to an end when a tester closes that defect assuring that it won't get reproduced again.

⇒ If the Test Engineer identifies a bug in the project, Test Engineer should report it using Bugzilla by selecting the status, severity, found in version/build and providing the steps with screen shots related to the bug.

⇒ Test Engineer will report the bug to the Tech Lead (Development Lead) in company at the reporting time i.e., NEW

⇒ Once the bug is OPEN, the developer will fix the bug. Test Engineer will be notified about the FIXED BUG which can be retested against a new build.

⇒ If the bug is fixed a RE-TEST is performed and the status is changed to FIXED VERIFIED.

⇒ Finally, the Test Lead will verify once again and will CLOSE the Bug.

Different Flows of the Defect

a) New -> Opened -> Fixed -> Fixed Verified-> Closed

b) New -> Opened -> Fixed -> Re-opened -> Fixed -> Closed

c) New ->Rejected or Not a Bug-> Closed

d) New -> Deferred or Duplicate-> Closed

e) New -> Opened -> Rejected -> Re-opened -> Fixed -> Closed

etc…

Defect 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.

2) **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.

3) **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.

4) **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".

5) **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".

6) **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.

7) **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'.

8) **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'.

9) **Closed:** When the defect does not exist any longer, then the tester changes the status of the defect to "Closed".

A Few More:

Not a Bug: If the defect does not have an impact on the functionality of the application, then the status of the defect gets changed to "Not a Bug".

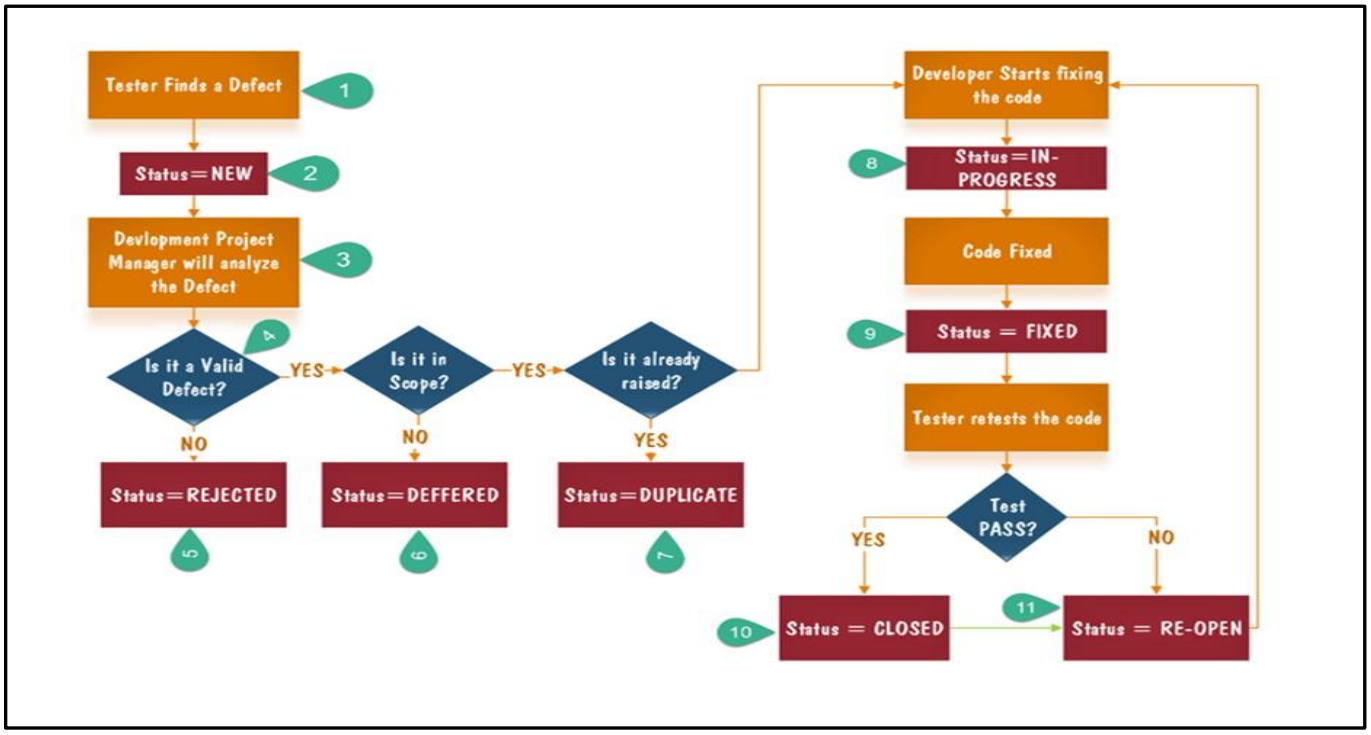
Duplicate: If the developer finds the defect as same as any other defect or if the concept of the defect matches any other defect, then the status of the defect is changed to 'Duplicate' by the developer.

Rejected: If the defect is not considered a genuine defect by the developer, then it is marked as "Rejected" by the developer.

Deferred: If the developer feels that the defect is not of very important priority and it can get fixed in the next releases or so in such a case, he can change the status of the defect as 'Deferred'.

The mandatory fields where a tester logs any new bug are Build version, Submit On, Product, Module, Severity, Synopsis and Description to Reproduce

In the above list, you can add some optional fields if you are using a manual Bug submission template. These Optional Fields include Customer name, Browser, Operating system, File Attachments, and screenshots.



Defect Management:

Any Flaw or, Imperfection in software work product (Deliverable). Expected Result is not matching with the application Actual Result.

Difference b/w Defect/ Error and failure:

⇒ Problem which is identified on Test Engineer machine i.e called DEFECT or BUG.
⇒ The same problem which is identified on Developer machine i.e called ERROR.
⇒ A deviation from the specified (or) expected behaviour that is visible to end-users is called FAILURE.

Major Activities in Defect Management:

1. Defect Identification
2. Defect Reporting.
3. Defect Tracking
4. Defect Re Testing
5. Defect closing

1. Defect Identification:

while executing the Test Cases, expected result is not matching with actual result.

2. Defect Reporting: Guidelines to Log defects: -

The moment Test Engineer identify the Defect in my current project, we are following Bug Reporting Guidelines to Log the Defect:

1. Reproduce the bug.

2. Reproduce the bug in peer (colleague) computer (Different Environments).

3. Check the Defect database for Duplicates.

4. Document the bug.

5. Send an Email to Test Lead for Review.

6. Once bug is approved, login into defect tracking tools and submit the bug.

7. At the time & Reporting the Defect status is NEW

Defect Reporting is done in Two ways:

1. Manually

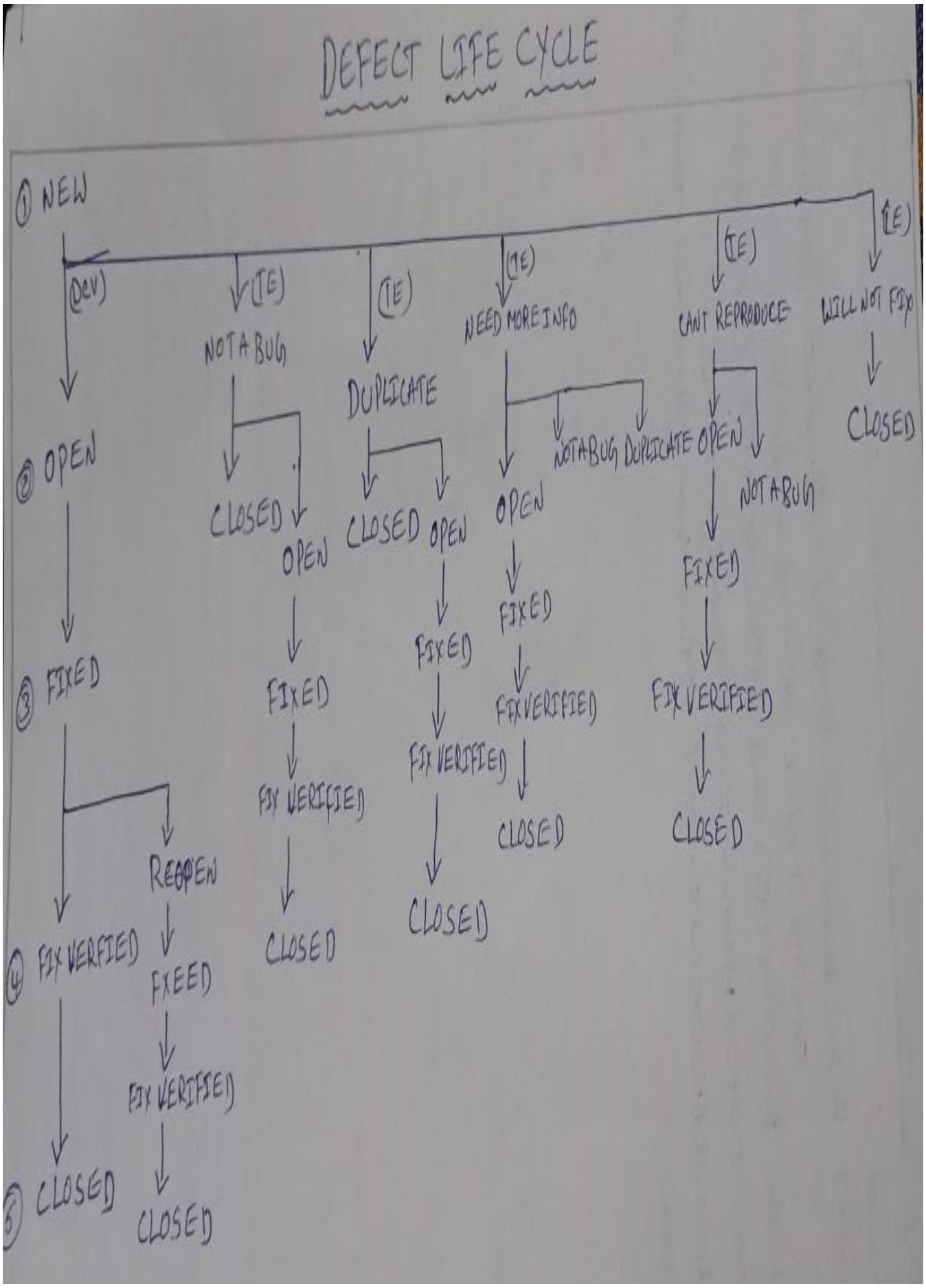
2. Automation

1. Manually: Copy the 'Defect Template from File Server vss and fill the template and send an Email to Assigned Person with Defect as an attachment.

2. Automation:

Industry specific defect Tracking Tools [company own Tools]

Third Party Defect Tracking Tools.


DEFECT LIFE CYCLE
1 NEW
(Dev)
(TE)
(TE)
(TE)
(TE)
(TE)
NOT A BUG
DUPLICATE
NEED MORE INFO
CANT REPRODUCE
WILL NOT FIX
2 OPEN
CLOSED
CLOSED
OPEN
CLOSED
OPEN
NOT A BUG
DUPLICATE
OPEN
CLOSED
NOT A BUG
3 FIXED
FIXED
FIXED
FIXED
FIXED
FIX VERIFIED
FIX VERIFIED
FIX VERIFIED
CLOSED
CLOSED
REOPEN
4 FIX VERIFIED
FIXED
FIX VERIFIED
CLOSED
CLOSED
5 CLOSED
CLOSED

⇒ Track+ [open source]

⇒ Bugzilla [open Source]

⇒ JTrac [open source]

⇒ Jira

⇒ Mantis [Open Source]

⇒ Quality centre/ALM (Commercial)

3. Defect Tracking:

Defect tracking is the process of tracking the logged defects in a product from beginning to closure (by inspection, testing, or recording feedback from customers), and making new versions of the product that fix the defects". Defect tracking is an important process in software engineering as Complex and business critical systems have hundreds of defects. One of the challenging factors is Managing, evaluating and prioritizing these defects.

⇒ A defect tracker keeps a track of the all defects. So that the management may not miss any defect from correction.

⇒ Due to these tracked defects, the most correct methods are adopted and preventive measures are taken to avoid further defects.

⇒ It saves the time of the engineers thus helping in the fast delivery of the work. Also, it enhances the efficient work delivered.

4. Defect Re Testing:

⇒ Retesting is testing of a particular bug after it has been fixed. Usually, tester raises the bug when they find it while testing the product or its component. This bug is assigned to a developer and he fixes it. Post fixing the bug is assigned to the tester for its verification. This testing is known as retesting.

⇒ Retesting also occurs when the product is already tested and due to some problems, it needs to be tested again. This test is named as retesting.

⇒ Retesting depends on the developer department that they are going to accept the bug testing or reject. Retesting is done when there is a specific bug when the bug is rejected by the developer and the tester department needs to tests the issues when the user reports a problem for retesting and fixing of an issue for better application and better workflow.

5. Defect Closing (Closure):

Once a defect has been resolved and verified, the defect is changed status as closed.

Sample bug report:

Application product Bug report sample

Application testing scenario:

Let’s assume in your application you want to create a new user with his/her information, for that you need to logon into the application and navigate to USERS menu > New User, then

enter all the details in the User form like, First Name, Last Name, Age, Address, Phone etc.Once you enter all these need to click on SAVE button in order to save the user and you can see a success message saying, "New User has been created successfully".

Now you entered into your application by logging in and navigate to USERS menu > New user, entered all the information and clicked on SAVE button and now the application crashed and you can see one error page on the screen, now you would like to report this BUG.

Now here is how we can report bug for above scenario:

Bug Name: Application crash on clicking the SAVE button while creating a new user.

Bug ID: The BUG Tracking tool will automatically create it once you save this.

Area Path: USERS menu > New Users

Build Number:/Version Number 5.0.1

Severity: HIGH (High/Medium/Low)

Priority: HIGH (High/Medium/Low)

Assigned to: Developer-X

Created By: Your Name

Created On: Date

Reason: Defect

Status: New/Open/Active – Depends on the Tool you are using

Environment: Windows 2003/SQL Server 2005

Description: Application crash on clicking the SAVE button while creating a new user, hence unable to create a new user in the application.

Steps To Reproduce:

1) Logon into the application

2) Navigate to the Users Menu > New User

3) Filled all the fields

4) Clicked on 'Save' button

5) Seen an error page "ORA1090 Exception: Insert values Error…"

6) See the attached logs for more information

7) And also see the attached screenshot of the error page.

Expected: On clicking SAVE button should be prompted to a success message "New User has been created successfully".

Save the defect/bug in the BUG TRACKING TOOL

Defects in WEB Application (Java):

List of Defects:

1 Blank Page is displayed.

2. Images are not loading.

3. Spelling mistakes and Alignments.

4. Invalid POPUP messages.

5. Links are not working.

6. Getting "ERROR ON PAGE on status Bar of the Browser.

7. Java. lang. Exception.

8.Java. SQL. Exception.

9. Sava.lang. NullPointer Exception.

10. java. lang. IndexOut of Bounds Exception.

11. HTML code is displayed on the Application.

12. CODE is displayed on Application.

13. Mandatory fields are not displayed in RED Colour.

14. Data Truncation.

15. Unable to view the Data from Database.

16. Able to add the Duplicate Names when fields are declared of primary key.

17. Rembering the previews added values when trying to add New Group.

18. Getting the JavaScript errors in application.

19. Java.lang. Number Format Exception.

20 Java.lang. reflect. Invocation Target Exception.

21. Displayed *NULL information is application.

22. While adding the Invalid/ Duplicate values, screen is not waiting to enter valid data, where as it is going to main page.

23. While working on application, getting "session Expired" error.

24. Executing HTML code.

25. While changing Browser Text size smallest to Largest getting alignment defects.

26. Error Rendering Page: java.lang. String Index Out of Bounds Exception: String index out of range: 4

27. Defect High Priority (P1) and High severity (Critical) Java.lang. Out of Memory Error: PermGen space

28. Java.lang. IllegalArgumentException: Path home does not start with

a "/" character.

Severity and Priority:

Severity:

Severity defines the importance of defect with response to functional point of view which means criticality of the defect with respective to the application. Defect Severity means how badly the defect has affected the application's functionality.

"Responsibility of Testing Team ".

Classification/Types of Severity:

1) Critical
2) Major
3) Moderate
4) Minimal

1.Critical (Blocker): This defect indicates complete shut-down of the process, nothing can proceed further. If the whole application's functionality is inaccessible or down because of a defect, such a defect is categorized as a critical defect. System Down/Critical Impact Complete system failure.

For Example: When the login screen of an application is not working and the user cannot log in, the whole application becomes inaccessible to the user.

2.Major: It is a highly severe defect and collapses the system. However, certain parts of the system remain functional. There won't be a complete system shutdown (which would be the case for a critical defect). Even so, it will prevent major, sometimes even basic, application functionalities from working.

For Example: In a banking application, a user cannot transfer money to any beneficiary, but that doesn't affect their ability to add new beneficiaries.

3. **Moderate:** It causes some undesirable behaviour, but the system is still functional. Usually, moderate severity defects have a workaround, so they may not block a functionality completely (unlike major severity defects where there is no workaround).

For Example: The download link in the Help section of an application needs to be fixed. However, the user is still able to read the document online.

4. **Minimal (Minor):** It won't cause any major break-down of the system. Defects of cosmetic nature that don't affect the application functionality or UX. But they are valid defects nonetheless. which means that all the U.I problems are not working fine, but testing can be processed without interruption. Such minor defects can wait until the next release because they do not restrict the application's functionality.

For Example: Spelling mistakes on the webpage. These are valid defects, but they can wait to be fixed since they're not affecting application functionality.

Priority:

Priority defines the importance of defect with respect to client point of view which means how soon it should be fixed.

"Responsibility of Development Team".

Classification of Priority:

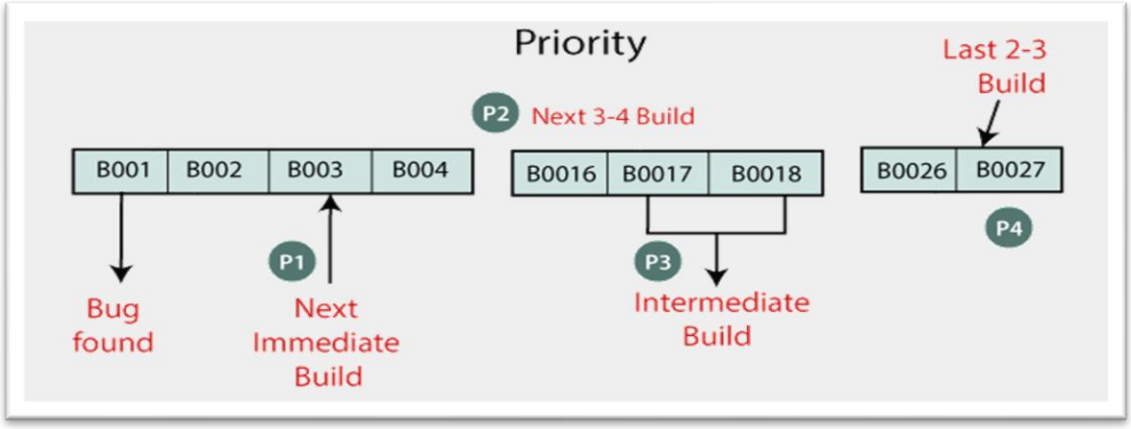
1) P1(Urgent)
2) P2(High)
3) P3(Medium)
4) P4(Low)

1. **P1-Urgent:** When the bug is just found, it will be fixed in the next immediate build, and give the Priority as P1 or urgent. It is a major impact on the customer application, and it has to be fixed first.

2. **P2-High:** If the Priority of the bug is P2 or high, it will be fixed in the next 3-4 builds.

3. **P3-Medium:** When the Priority of the bug is P3/medium, then it will be fixed in the intermediate build of the application. In this, the problem should be fixed before the release of the current version in development.

4. **P4-Low:** And at last, if the Priority is P4/low, it will be fixed in the last 2-3 build of the software. The flow should be fixed if there is time, but it can be deferred with the next release.



Different Levels of Priority and Severity:

Priority and Severity have some classifications amongst them that aid in determining how the defect must be handled. A lot of different organizations have different defect logging tools, so the levels might vary.

Let's take a look at the different levels for both Priority and Severity.

1) High Priority, High Severity
2) Low Severity, High Priority
3) High Severity, Low Priority
4) Low Severity, Low Priority

1) High Severity and High Priority:

Any Critical/major business case failure automatically gets promoted to this category. Any defects due to which the testing cannot continue at any cost or causes a severe system failure to fall into this category.

Examples:

1. The system crashes after you made the payment or when you are not able to add the items to the Cart, this defect is marked as High Severity and High Priority defect.
2. ATM vending currency feature wherein after entering the correct username and the password, the machine does not dispense money but deducts the transferred from your account.

3. An error which occurs on the basic functionality of the application and will not allow the user to use the system. (Ex: A site maintaining the student details, on saving record if it, doesn't allow to save the record then this is high priority and high severity bug.)
4. Clicking on a particular button doesn't load the feature itself. Or performing a particular function brings down the server consistently and causes data loss. The red lines in the above figure indicate these kinds of defects.

2) Low Severity, High Priority:

Any minor severity defects that could directly impact the user experience automatically gets promoted to this category. Defects that have to be fixed but do not affect the application come under this category. In terms of functionality, it is not affecting anything so we can mark as Low Severity, but it has an impact on user experience. This kind of defect needs to be fixed on high priority even though they have very less impact on the application side.

Examples:

1. The logo of the company in the front-page is wrong, it is considered to be High Priority and Low Severity defect. In the Online shopping website when the FrontPage logo is spelled wrong, for example instead of Flipkart it is spelled as Flikart. In the bank logo, instead of ICICI, it is written as ICCCI.
2. If the company name is misspelled in the home page of the website, then the priority is high and severity is low to fix it.
3. The action buttons are not visually appealing or the information on the page appears
4. The feature is expected to display a particular error to the user with respect to its return code. In this case, functionally the code will throw an error, but the message will need to be more relevant to the return code generated.

3) High Severity and Low Priority:

Any defect that is functionally not meeting the requirements or have any functional implications on the system but side lined to back seat by the stakeholders when it comes to business criticality automatically gets promoted to this category. Though this feature is having a functional defect, as it is not impacting the end customers directly, a business stakeholder can classify the defect under low priority though it has a severe functional impact on the application.

Defects that have to be fixed but not immediately. This can specifically occur during ad-hoc testing. It means that the functionality is affected to a large extent, but is observed only when certain uncommon input parameters are used.

Examples:

1. If the application is crashing on passing very large input for processing (which is very rarely done).
2. There are some buttons on the website which are overlapping. Although clickable, are

creating a fuss.

3. Web page not found when user clicks on a link (user' s does not visit that page generally).
4. In a social networking site, if a beta version of a new feature is released with not many active users using that facility as of today. Any defect found on this feature can be classified as a low priority as the feature takes back seat due to business classification as not important.

4) Low Severity and Low Priority:

These defects are occurred, when there is no functionality impact, but still not meeting the standards to a small degree. Generally cosmetic errors or say dimensions of a cell in a table on UI are classified here. Any spelling mistakes /font casing/ misalignment in the paragraph of the 3rd or 4th page of the application and not in the main or front page/ title.

Examples:

1. If the privacy policy of the website has a spelling mistake, this defect is set as Low Severity and Low Priority.
2. A spelling mistake on the page of the site which is not frequently visited.
3. The colour of any text does not match the theme of the website.
4. Any cosmetic or spelling issues which is within a paragraph or in the report
5. The 'Help' section of the banking website has a subsection whose theme does not match the whole website. Also, not many users will access this particular section. Hence, this defect falls under the category of Low Severity and Low Priority.

Difference between Severity and Priority:

Severity	Priority
Severity is a parameter to denote the impact of a particular defect on the software.	Priority is a parameter to decide the order in which defects should be fixed.
Severity means how severe defect is affecting the functionality.	Priority means how fast defect has to be fixed.
Testing engineer decides the severity level of the defect.	Product manager or developers or product owners. decides the priorities of defects.
Severity is associated with functionality or standards	Priority is associated with scheduling
Severity indicates the seriousness of the defect on the product functionality.	Priority indicates how soon the bug should be fixed.
Severity status is based on the technical aspect of the product.	Priority status is based on customer requirements.
Severity is categorized into four types: • Critical • Major • Moderate • Minor	Priority is categorized into three types: • Urgent(P1) • High(P2) • Medium(P3) • Low(P4)

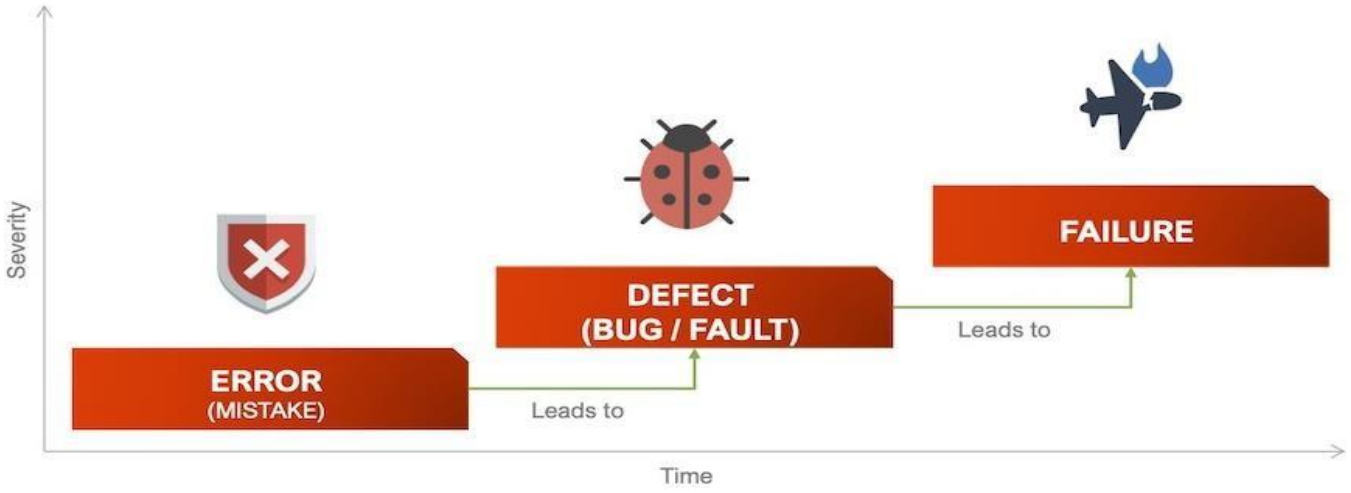
Error, Defect, Bug and Failure:

Defect: A defect is a variance between expected results and actual results, detected by the developer after the product goes live. The defect is an error found AFTER the application goes into production. In simple terms, it refers to several troubles with the software products, with its external behaviour, or with its internal features. In other words, "Deviation from the expected behaviour to the actual behaviour of the system is called defect".

Bug: It is an informal name specified to the defect. The Test Engineers submit the bug.

Error: An error is a mistake, misunderstanding, or misconception, on the part of a software developer. The category of developers includes software engineers, programmers, analysts, and testers. For example, a developer may misunderstand a design notation, or a programmer might type a variable name incorrectly – leads to an error. An error normally arises in software, it leads to a change the functionality of the program.

Failure: The deviation identified by end-user while using the system is called a failure. For example, in a bank application if the Amount Transfer module is not working for end-users when the end-user tries to transfer money, submit button is not working. Hence, this is a failure.



Difference between Error, Defect and Failure:

Defect	Error	Failure
A Defect is a variance between expected and actual results. An Error that the tester finds is known as Defect.	The Error is a human mistake. An Error appears not only due to the logical mistake in the code made by the developer.	Failure is a consequence of a Defect. It is the observable incorrect behaviour of the system. Failure occurs when the software fails to perform in the real environment.

The Testers identify the defect. And it was also solved by the developer in the development phase or stage.	The Developers and automation test engineers raise the error.	The End-Users or Realtime users are identify Failures.
The Defect is the difference between the actual outcomes and expected outputs.	An Error is a mistake made in the code; that's why we cannot execute or compile code.	If the software has lots of defects, it leads to failure or causes failure.
The variation between the actual results and expected results is known as defect.	We can’t compile or run a program due to coding mistake in a program. If a developer unable to successfully compile or run a program then they call it as an error.	Once the product is deployed and customers find any issues then they call the product as a failure product. After release, if an end user finds an issue, then that particular issue is called as failure.

I. DEFECT FROM TEST ENGINEER (TE) TO DEVELOPMENT LEAD (DL):

Category	Project Bug		
Tittle	ERPPMC-2.0V. Bol-Venkat - Admin - Unable to Add NEW		
Status	New	Browser	All Browsers
Severity	Major	Found in version	2.0V
Priority	* None *	Found in Build	01
Module	Admin	Fixed in version	
Originator	Rahul (TE)	Fixed in Build	

SUBMIT

Note:

As soon as Test Engineer Assign the defect to Development Lead (DL) (By clicking on [Save/Submit] button from Track+/Bugzilla/Jira.

1. Unique Defect ID will be created automatically.
2. Tool will send an Email notification to Development Lead.
3. Tool will send cc Email notification to Test Engineer (TE) and Test Lead (TL).

II. DEFECT FROM DEVELOPMENT LEAD (DL) TO DEVELOPER (DEV):

Defect ID: 1999

Last Changed Rahul (DL)

Status OPEN

Priority P2

Assigned to Geetham

Comments from Rahul:

Defect in "ERPPMC EMP• Java" file.

Please fix it.

SUBMIT

Note:

TO EMAIL: DEV

CC EMAIL: DL, TE & TL

III. DEFECT FROM DEVELOPER TO TEST ENGINEER:

Defect: ID 1999

Geetham (DEV)

Last changed

Status FIXED

Assigned To VENKAT(TE)

Fixed in version 3.0v

Fixed in Build 01

Comments from Geetham:

Defect is fixed.

Please Retest.

SUBMIT

Note:

To Email: TE

Cc Email: Dev, DL & TL

IV. DEFECT FROM TEST ENGINEER(TE) TO TEST LEAD (TL):

Defect ID: 1999

Last Changed Venkat (TE)

Status FIX VERIFIED

Assigned TO Ayush (TL)

Comments from Venkat (TE):

Verified on ERPPMC 3.0V Build 0l Defect is fixed.

Hence status is changed to "FIX VERIFIED"

SUBMIT

Note:

To Email: TL.

CC Email: TE, DEV, & DL.

V. DEFECT FROM TL TO TE:

Defect ID:	1999
Last changed	Ayush (TL)
Status	CLOSED
Assigned To	VENKAT(TE)

Comments from Ayush:
Defect is closed.

SUBMIT

Note:

To Email: TE

CC Email: TL, DEV & DL

IV.1 **DEFECT FROM TEST ENGINEER TO DEVELOPER:**

Defect FD:	1999	
Last Changed	VENKAT(TE)	III. Fixed:
Status	RE-OPEN	IV. Fix Verified
Assigned to	Geetham (Dev)	IV.1. Re-Open

Comments from Venkat (TE):
Verified on ERPPMC 3.0V Build 01, Defect is still existing.
Hence status is changed to "Reopen".

SUBMIT

Note:

To Email: DEV

CC Email: DE, DL &TL

II.2 **DEFECT FROM DL TO TE:**

Defect ID:	1999	II.0. Open
Last Changed	Rahul (DL)	II.1. Not a Bug
	DUPLICATE	II.2. Duplicate
Status		II.3 Need More Info
Assigned to	Geetham (Dev)	II.4. Can't Reproduce
Duplicate ID	1888	II.5 Will not Fix

Comments from Rahul:

Please find Duplicate ID.

SUBMIT

Note:

To Email: TE

CC Email: DL, DEV & TE

Defect Process:

Method 1:

Defect Life Cycle Diagram.

New->Open -> Fixed -> Fix Verified –>Closed.

Method 2:

1. Defect Identification
2. Defect Reporting (Track + | Bugzilla (JIRA)
3. Defect Tracking (Status = Fixed)
4. Defect Retesting:
5. Defect Closing.

Method 3:

As soon as the Test Engineer identifies the Defect in my current project, we are following Bug Reporting Guidelines to report the defect. Test Engineer has to report defect using Bugzilla/Jira/Track+ by selecting Status, Severity, found in version /build and Providing Steps to Reproduce and if it requires adding an Attachments, attach screen shots of necessary information regarding defect. Test Engineer will report the defect to Tech Lead (Development Lead) in my Company, at the time of reporting the defect Status is called NEW.

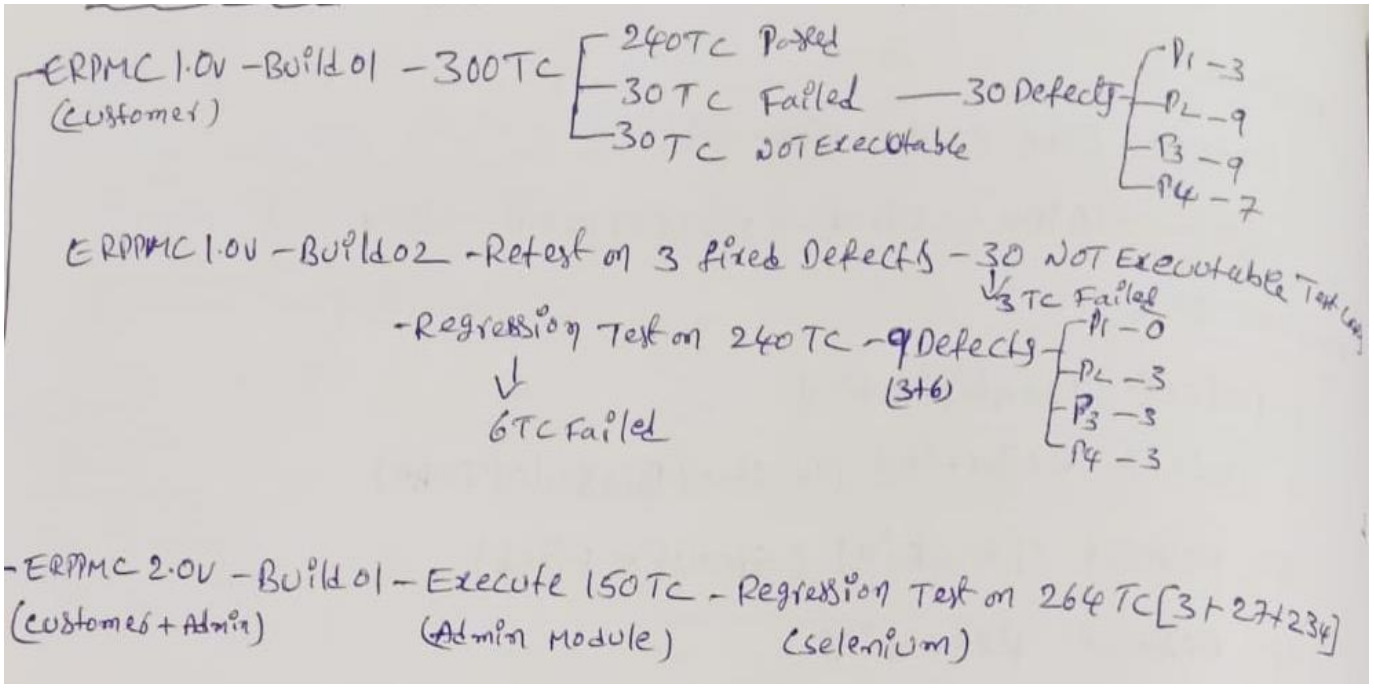
Once the defect OPEN the Developer will fix the defect. Test Engineer will be notified of FIXED defects which can be RETESTED against new build. If Defect is fixed Perform Re-Test and change the Status to FIXVERIFIED. Finally, the Test Lead will verify once again and he will CLOSE the Defect.

NEW ------(DL)------- →OPEN
OPEN------(Developer) ------- →FIXED
FIXED ------(Test Engineers) -------- → FIXVERIFIED

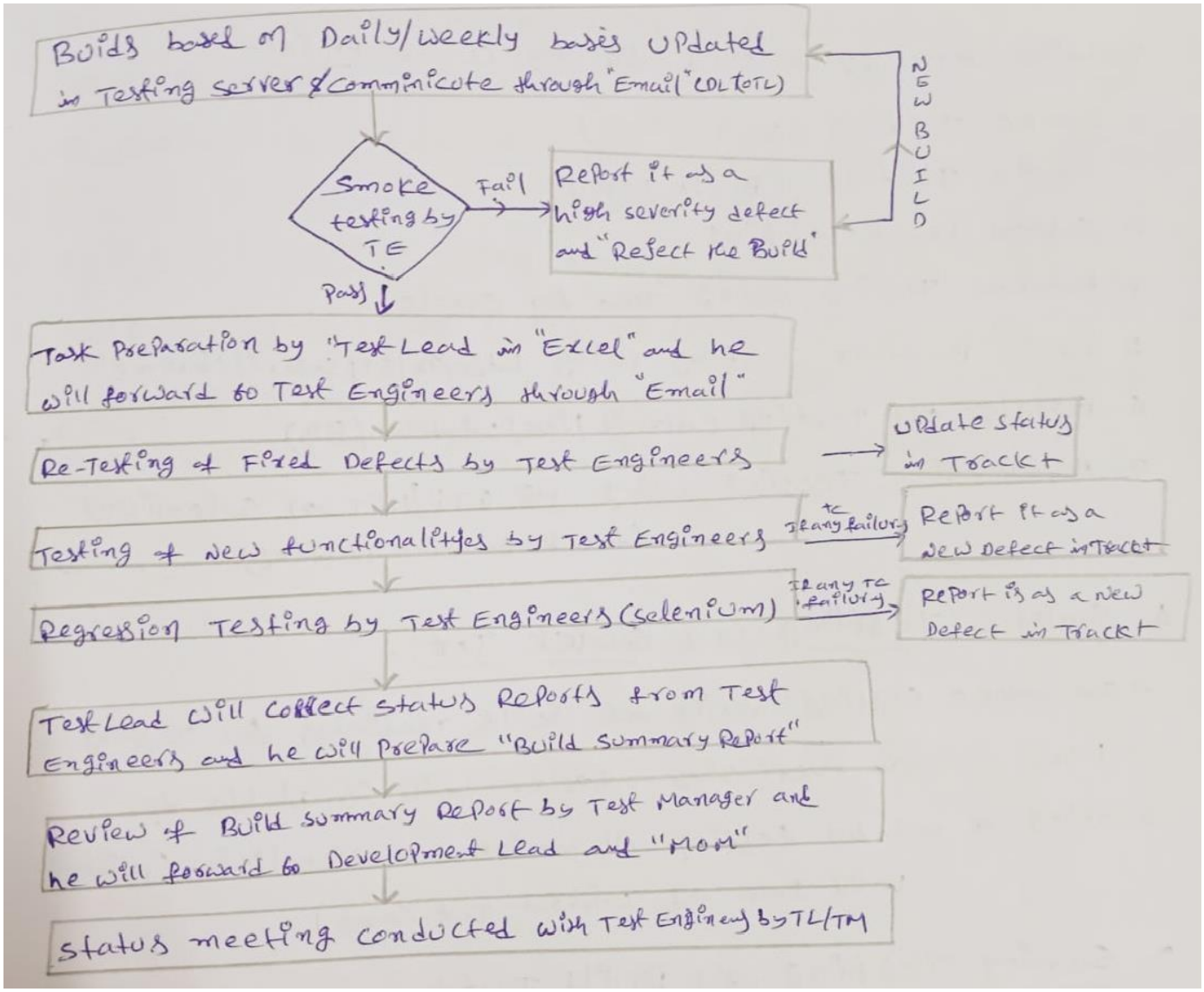
FIXVERIFIED------(Test Lead) -------- →CLOSED

Venkat: TE
Rahul DL
Geetham: DEV
Ayush: TL

Retesting and Regression Testing:



Project Build Workflow:



Testing Process Testing Life Cycle / STLC:

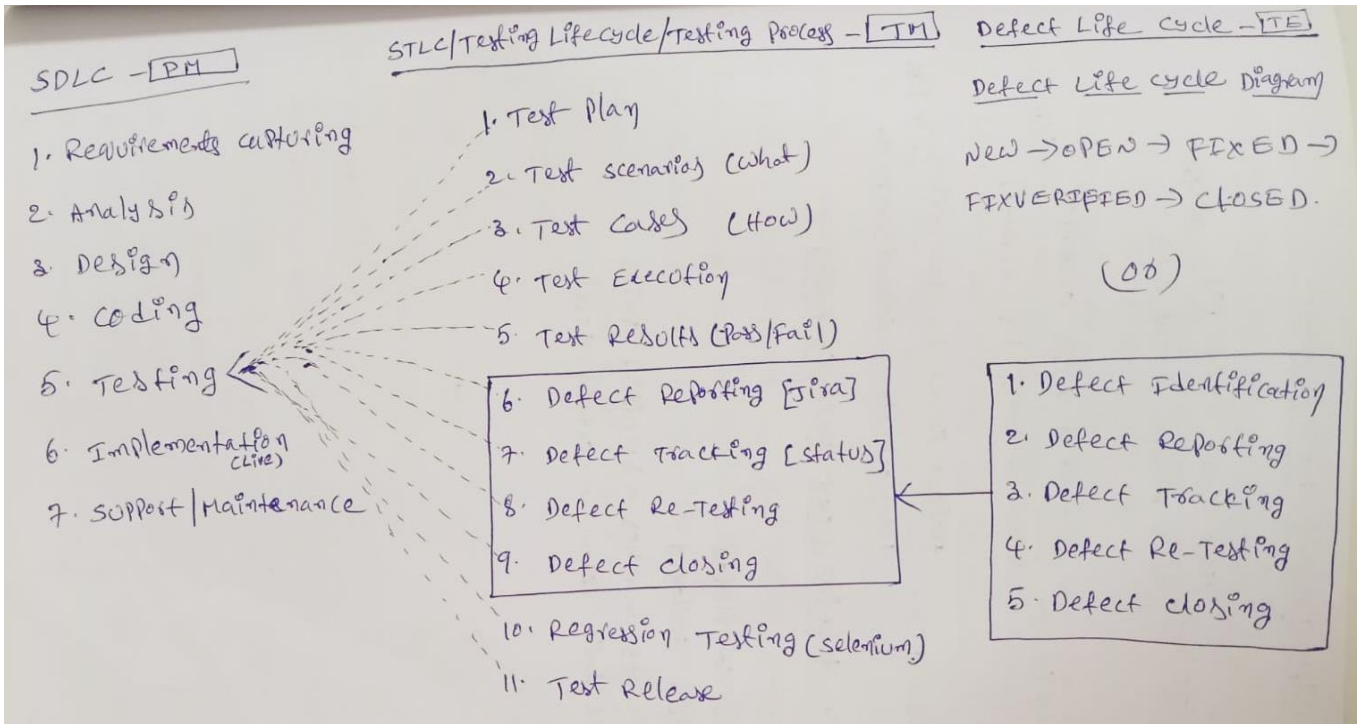
Current Project Testing Process:

My Current Project Testing Process started from Preparation of the Test Plan. My Test Lead is prepared the test plan for my current project. We are executing the Test plan, which includes, Identifying the Test Scenarios, Review and Approval of the Test scenarios (My self is involved in identified Test Scenarios for customer module). Once Test Scenarios are approved, we Prepared Test cases for all approved Test scenarios. Review and Approval of the Test cases. Executing the Test Cases, Performing all testing activities like Functionality Testing, Re-Testing, Regression Testing Compatibility, server Log files and Usability Testing + PROJECT BUILDS WORK flow.

We used to receive builds daily/weekly basis.

- Smoke Test
- Sanity Test
- Re-Test
- New Functionality

- Regression
- Build Summary Report by Lead
- Status Meeting
- Next Build……



Test Metrics /Quality Project Management:

Software testing metrics are quantifiable indicators of the software testing process progress, quality, productivity, and overall health. The purpose of software testing metrics is to increase the efficiency and effectiveness of the software testing process while also assisting in making better decisions for future testing by providing accurate data about the testing process.

Test Coverage (TC) = Test Cases executed / Total Number of Test Cases ×100

Test Efficiency (TE)=Number of bugs accepted / Total number of bugs reported ×100

Defect Leakage (DL) =Defect found in production / Total number of defects reported ×100

Test Coverage: Test coverage is defined as a statistic that indicates the quantity of testing completed by a collection of tests. Test coverage is defined as a metric in Software Testing that measures the amount of testing performed by a set of tests. It will include gathering information about which parts of a program are executed when running the test suite to determine which branches of conditional statements have been taken.

Test coverage (TC)=Test cases executed / Total number of Test cases *100.

Test Efficiency (TE): Test efficiency refers to the effectiveness and productivity of the software testing process. It measures the ability of the testing process to detect defects, ensure software quality, and achieve testing objectives with minimal time, effort, and resources. Test efficiency can be measured using various metrics, such as test coverage, defect detection rate, test automation coverage, and test cycle time. A high-test efficiency indicates that the testing process is optimized, and defects are detected and resolved promptly, resulting in a high-quality software product.

Test Efficiency (TE)=Number of bugs accepted / Total number of bugs reported ×100

Defect Leakage (DL): Defect leakage refers to the situation where a defect or a software bug is detected by end-users or customers after the software product has been released for deployment. In other words, it is the percentage of defects that are not detected during the testing phase of software development and are released into production. Defect leakage is a metric measuring the percentage of defects leaked from the current stage of testing to the next stage. The impact of defect leakage can be severe, leading to customer dissatisfaction, loss of revenue, brand reputation damage, and even legal consequences.

Defect Leakage (DL) =Defect found in production / Total number of defects reported ×100.

When to STOP Testing (Test Exit Criteria):

⇒ Deadlines.
⇒ Test cases completed.
⇒ Test budget depleted.
⇒ Requirements reaches a specified Point.
⇒ Bug rate falls below a certain level.

Status Reports:

Software testing status report track the software bugs or defects with their criticalness, project issues summary, pending work status with teams and project milestone. A testing status report is a report that contains a summary of all the testing-related activities that happened within a particular time. It highlights the status of the project using quantitative information.

Purpose:

- Part of Project Management.
- Monitoring and managing the project Deliverables.
- Part of SEI CMMI Level2.

Types of Status Reports:

1. Daily status Report [DSR]
2. Daily Defect Report [DDR]
3. weekly Status Report [WSR]
4. Retesting Report

1. **Daily status Report:**

Every day evening by 6:30 PM we will send the Daily Status Report to our Test Lead in the Email, about the status of the Assigned tasks.

Rajesh,

I have completed the 30% of assigned task. In that, I Prepared 100 Test cases.

Thanks and Regards

Krishna. N

2. **Daily Defect Report:**

Every day evening by 5:30 PM, If we found any defects, we will send the Daily Defect Report to Test Lead using Daily Defect Report Template.

Rajesh,

Today, I have executed To Test cases, In that 5 test cases are failed and reported as defects. Please find the Daily Defect Report as an attachment.

Regards.

Venkat.

Daily Defect Report Template:

S.NO	Defect ID	TITTLE	Severity	Found in Version	Found in Build	Logged By	Date Logged
1	352	ERPPMC-1.0V-B0l-Krishna - Employee Login is not available	Major	1.ov	01	Krishna	6-Nov-2022
2	359	ERPMC-1.0V-B0l-Krishna-Text box is not visible	Minimal	1.0V	01	Krishna	6-Nov-2022
3	370	ERPPMC-10V-B0I-Krishna –	Major	1.0V	01	Krishna	6-Nov-2022

		[Order Now] is not functioning					

3. **Weekly Status Report:**

Every week ending Friday by 5:00 PM we will send the weekly status Report to Test Lead using weekly Status Report Template.

Rajesh,

please find the weekly status Report as an Attachment-

Regards,

Krishna.

Weekly Status Report Template:

Project code		Week Ending:	
Project Name		Test Engineer:	

1. High Level Summary:

2. Major Accomplishments since Last Reporting Period:

3. Items Not Completed As planned:

4. Defect Summary: By severity

S.NO	Defect ID	Severity

5. Effort Summary:

S. No	Activity	Planned	Hours	Used to	Date	Balance

6. Risks and Issues:

7. Items planned for Next Reporting Period:

4. **Retesting Report:**

If Test Lead will assign the fined defects for Retesting Test Engineers will perform the Retesting on all fixed defects against new build and they will update status in Bugzilla and as well Re-testing Template and they will forward same to Test Lead.

S.NO	Defect ID	TITTLE	Severity	Fixed in Version	Fixed in Build	Test Engineer	Status Before Test	Status After Test	Comments
1	11588	ERPPMC-10V-B01-Krishna-Employee Login' is not available	Major	2.0v	1	Krishna	Fixed	Fix Verified	
2	14993	ERPPMC-1.0v-B0l-Krishna-Text box is not visible	Minimal	2.0v	1	Krishna	Fixed	Re-Open	Defect is Still Exists
3	14997	ERPPMC-1.0v-B01-Unable Shop Now' is	Major	2.0v	1	Krishna	Fixed	Unable to Test	'Shop Now' is not functioning and reported as a new

		not functioning							defect= 14998

Mobile Application Testing [Mobile APPS]:

As we know in the current time, the use of mobile phones is increasing day by day. People install new applications based on ratings and reviews daily. The ratings and reviews of the applications depend on the working of the application. So, this thing makes the testing of the mobile application important. Different mobile phones run on different operating systems and contain different screen sizes; so, this thing makes the mobile phone's testing is a mandatory process in the software development process. The updating in the app entertains the users. New bug fixes in the app makes sure that nobody uninstalls the app. Testing is essential for apps to stay in the market.

As we know in today's time, mobile phones are popular in the market. We do not want to switch ON our PCs and Laptops. Instead, we want those devices which can be easily handled and can perform the task quickly.

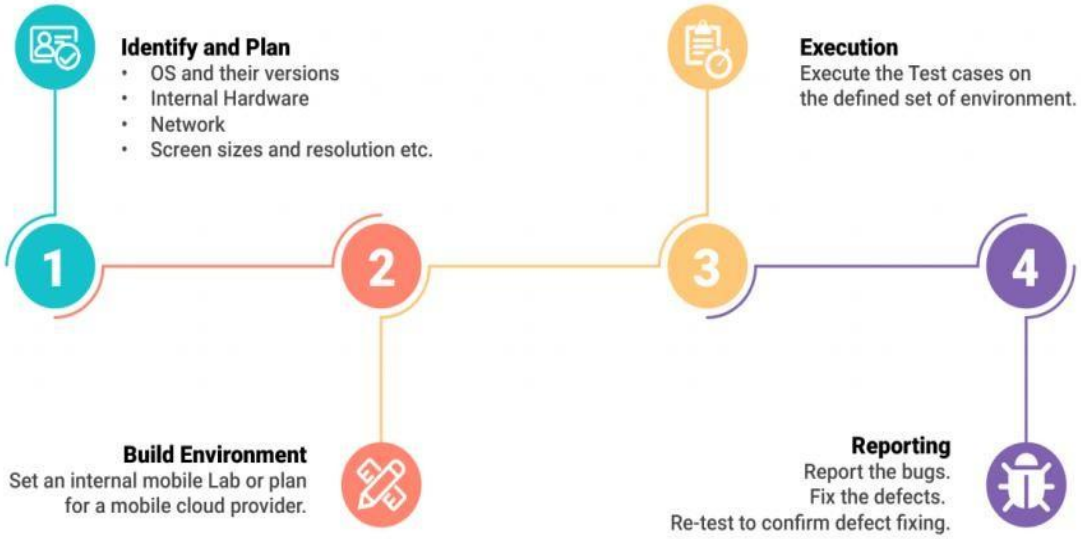
That is why mobile phones should be adequately tested before releasing them to the market.

"Mobile application testing is essential because the millions of users can use a specific product. If there is a bug in the product, that product will not be accepted by the client. The bug in the product can be the loss of memory, legal issue, and the irreplaceable damage in the image".

1. Mobile App Testing Process / Testing Life Cycle

a) Mobile Test Plan and Test Strategy definition.

b) Device identification process.
c) Application specific Test Case development.
d) Lab setup (Browsers/OS/Devices/Emulators).
e) Live Handset Testing i.e., iOS/Android.
f) Monitor Results.
g) Defect Reporting using QC/ALM (or) Bugzilla.



2. Mobile Platform (Mobile Operating System):

- Mobile Platform is software that allows smart phones, tablet PCs and other devices to run applications and programs.
- Mobile platform is Operating system for mobile applications.
- A mobile OS allows application software to operate on mobile devices. It is similar to desktop OS in certain ways, but it is simpler and lighter in comparison.

- Smartphone operating systems include Windows Mobile, Blackberry, Android, iPhone OS.
- Mobile operating systems combine computer and handheld device features. They frequently incorporate a cellular modem and SIM card tray for phone and Internet services. When you acquire a mobile device, it comes preinstalled with a device-specific operating system.

3. Different types of Mobile Platform (OS):

A mobile operating system allows the user to run other different application software on the mobile, tablets, etc. Moreover, we can say that it is a type of operating system which is specially designed for mobiles, tablets, smartwatches, etc. An operating system (OS) is a program that acts as an interface between the system hardware and the user. The mobile operating system will also determine which third-party applications can be used on your device. Some of the more common and well-known mobile operating systems include the following:

Types of Popular Mobile Operating System:

1. Android
2. iOS
3. Windows phone
4. Blackberry (BB)

3.1 Android:

The Android OS is the most common operating system among the mobile operating system developed by Google in 2008. Android is open Source (Free) from Google. This OS is based on the Linux kernel. Android's releases are named after sweets (or) dessert items. Jelly Bean (4.1), (4.2), (4·3).

1. Kit Kat (4.4)
2. LOLLI POP (5.0)
3. Marshmallow 6.0.1
4. Nougat 7.1.2
5. Oreo 8.1
6. Android 12

3.2.ios:

It is an operating system developed by Apple Inc. to be developed for iPhone. Then, it was expanded for iPad, iPad Touch, Apple Touch, etc. It is a secure, commercial operating system and is only available for Apple products as the company does not license it for third-party hardware. The main difference between Android and iOS is that iOS runs only on Apple products. Users can download millions of applications on Apple App Store directly to any device which is running on iOS. Current version of iOS is 15.5

3.3 Windows Phone:

It is an operating system developed by Microsoft and is mainly designed for pocket Personal computers and mobile phones. It has computer-based Windows OS features and some additional features for smartphones. It has a separate corner for kids so that they do not reach any personal data and misuse.

⇒ Windows phone is from Microsoft.
⇒ It is closed source and Property.
⇒ Windows phone devices are made Primarily by Nokia, along with HTC, Samsung.
⇒ Current version of windows phone is 10.0.1

3.4 Blackberry:

The developer of this operating system is Research in Motion i.e., RIM to be used on popular BlackBerry's handheld devices. BlackBerry provides API classes for developers to code applications. It runs only on BlackBerry phones such as Curve, Perl, BlackBerry bold, and Storm. All applications under BlackBerry are written on Java and applications can be installed only from BlackBerry app world through BlackBerry Desktop Manager.

⇒ It is closed source and propriety.
⇒ All phones and tablets are manufactured by Blackberry itself.

4. What is Mobile Application?

A Mobile app if a computer Program designed to run a Smartphones, tablet computers and other mobile devices.

1. Apple App Store.
2. Google play.
3. Windows phone store.
4. Blackberry APP world.

5. Different file format of APP's based on Mobile OS:

Based on mobile platform mobile app computer program is designed

Android	.apk (Android Application Package)
ios	. iPa (iPhone Application Archive)
Windows Phone	.xap (silverlight Application Package)
Black berry	. bbb (Blackberry Backup files)

6. Mobile Build mail from Dev Team to Testing Team:

To our QA Team,

EPPPMC Mobile client (Android version and iOS) Release 9.0 Build 30 is ready for download.

Test Cases for Testing a Mobile App:

In addition to functionality-based test cases, Mobile application testing requires special test cases which should cover the following scenarios.

1. **Battery usage:** It's important to keep a track of battery consumption while running applications on mobile devices.
2. **The speed of the application**: the response time on different devices, with different memory parameters, with different network types, etc.
3. **Data requirements**: For installation as well as to verify if the user with the limited data plan will able to download it.
4. **Memory requirement**: again, to download, install and run
5. **The functionality of the application**: make sure the application is not crashing due to network failure or anything else.

To use this new version of APP:

1. Remove the existing app from your devices (Android and ios)
2. Download mobile client .ipa file from clients Server and Transfer to mobile devices.
3. Install the release 9.0 Build 30 ERPPMC mobile client.

Thanks,

Build Team.

Feature Name	Story ID	Test Case ID	Summary	Precondition	Execution Steps	Expected Result	Actual Result
Install	Mob-1214	Mob-1214_1	Verify that application should be Installed successfully.	1. Select the application from google play.	1. Click on install button. 2. Navigate to the menu and click on the newly installed app.	The application should be Installed successfully.	
Uninstall	Mob-1214	Mob-1214_2	Verify that application should be Uninstalled successfully.	1. Execute test case ID: Mob-1214_1	1. Click on settings. 2. Select the newly added application on Mob-1214_1. 3. Click on Uninstall button. 4. Verify.	The application should be Uninstalled successfully.	
Interruption by Calls	Mob-1214	Mob-1214_3	Verify that user should able to accept Phone calls when application is running and should continue from the same point.	1. Execute test case ID: Mob-1214_1	1. Open the application. 2. Navigate here and there for a moment. 3. Make a call from another device to the device where you have opened the application. 4. Pick up the call. 5. Now disconnect it and verify.	User should able to accept Phone calls when application is running and should continue from the same point.	
Interruption by Messages	Mob-1214	Mob-1214_4	Verify that user should able to accept messages when application is running and should	1. Execute test case ID: Mob-1214_1	1. Open the application. 2. Navigate here and there for a moment. 3. Send a message from another device to the device where you have opened the application.	User should able to accept messages when application is running and should continue from the same point after reading the message.	

9)

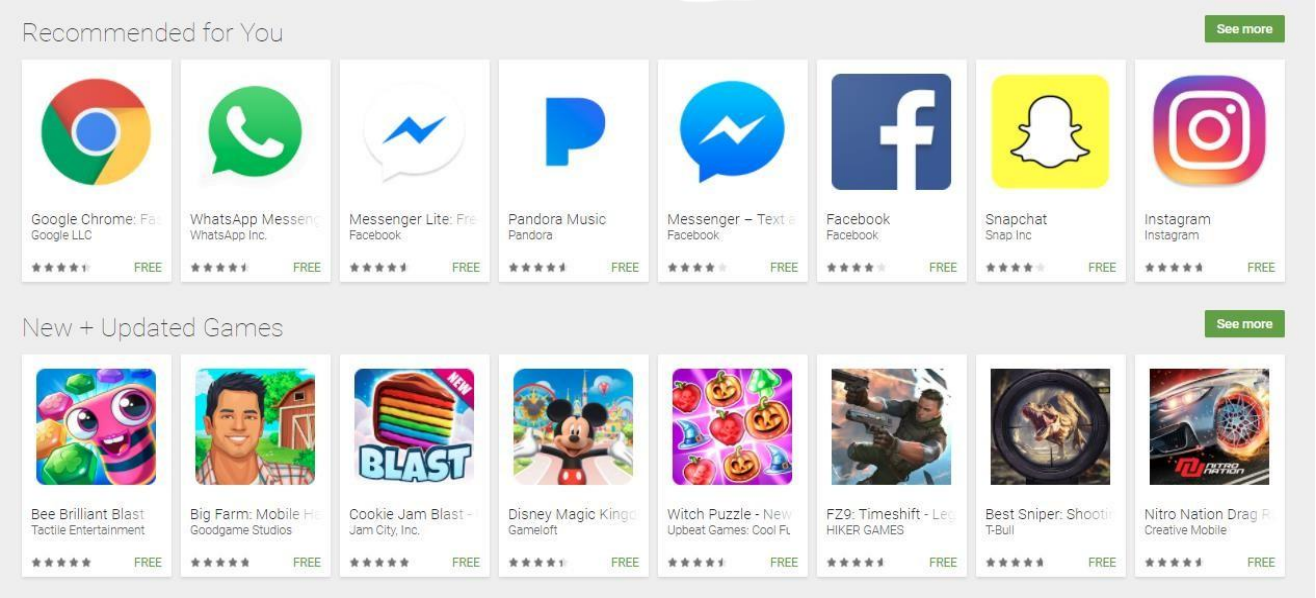
			continue from the same point after reading the message.		4. Read the message. 5. Close the message app and verify.		
Memory	Mob-1214	Mob-1214_5	Verify that user should able to see proper error message when device memory is low.	1. The device memory space should not more than 20 mb. 2. Execute test case ID: Mob-1214_1 Note: Application memory requirement is 25 MB.	1. Go to the app from google play store. 2. Click on the install button. 3. Wait till the application get installed and verify.	Application should display with proper error message when device memory is low.	
Exit application	Mob-1214	Mob-1214_6	Verify that user should able to exit from application if we click on end key.	1. Execute test case ID: Mob-1214_1	1. Click on app and open it. 2. Now press the end key and verify.	User should able to exit from application if we click on end key.	
Battery	Mob-1214	Mob-1214_7	Verify that user should able to see the alert when battery is low.	1. Execute test case ID: Mob-1214_1	1. Click on app and open it. 2. Use the application till you get the low battery indication.	When battery is low the alert should display.	
Battery Consumption	Mob-1214	Mob-1214_8	Verify that application should not consume more battery	1. Execute test case ID: Mob-1214_1 2. Full charge your device.	1. Click on app and open it. 2. Use the application and verify the status of the battery in 15 mins interval of time.	Application should not consume more battery	

Charge	Mob-1214	Mob-1214_8	Verify that application should run when inserting the charger. It will not affect the application	1. Execute test case ID: Mob-1214_1	1. Click on app and open it. 2. Insert the charging pin in between running of the application and verify.	Application should run when inserting the charger. It will not affect the application	

7. What is Third party app:

A third-party app is an application created by a developer that isn't the manufacturer of the device the app runs on or the owner of the website that offers it. You could think of those as first-party apps, although that term isn't used very widely.

⇒ Third party app is not built-in app in mobile platform and not available in stores also.
⇒ Third Party app is directly downloaded from Developer's website then install.



8. Installation process for Third Party app:

Android: (file must be .apk format)

Steps for enable third party apps installation on the Android phone/ device with apk file.

1. Go to Settings.
2. Go to Security.

3. Scroll down and check "Unknown Sources" box.
4. Tap 'OK' when it shows the warning.

All Done, now you can install 3rd party apps through APK file on your Device.

9. Mobile Defects:

Think of a defect as a deviation from expected software behavior. In other words, if a website or app functions differently from what users expect, that particular variation would be considered a defect. In software testing circles, the term defect is often used interchangeably with bug.

Before mobile application goes into production, QA engineers test it for compliance with all the standards and specifications described at inception phases. Along with that, specialists examine app functionality and its ability to compete with similar programs.

In other words, all this refers to product quality that is verified during testing.

10.Mobile Testing Process:

A mobile or web-based application testing process is the set of activities that are conduct by the software tester during or after the development process with the help of Mobile app testing techniques. The usage of mobile application is increasing in all business areas like Schools, Publishers, Retailers, HealthCare providers, etc. Only the development of the mobile application is not enough; it also needs testing before introducing the app into the market.

The testing is essential so that the application can run smoothly without any difficulties.

Sample Mobile Defects:

Tittle: QA-Mobile APP-ios-Android-Junk values are displayed and [Return] is missing.

Steps to Reproduce:

1. Lunch the [EP] app icon.
2. Login with valid user id and password.
3. click on "click the button at the top to start gear icon and Select 'Add menu button.
4. Observe "Add a new value to the list name screen with [Return] button.
5. Enter 'Fixed Name' and 'Tittle' as special characters [Ex: $$$].
6. select [update] -right corner and observe valid popup message and click on [ok].

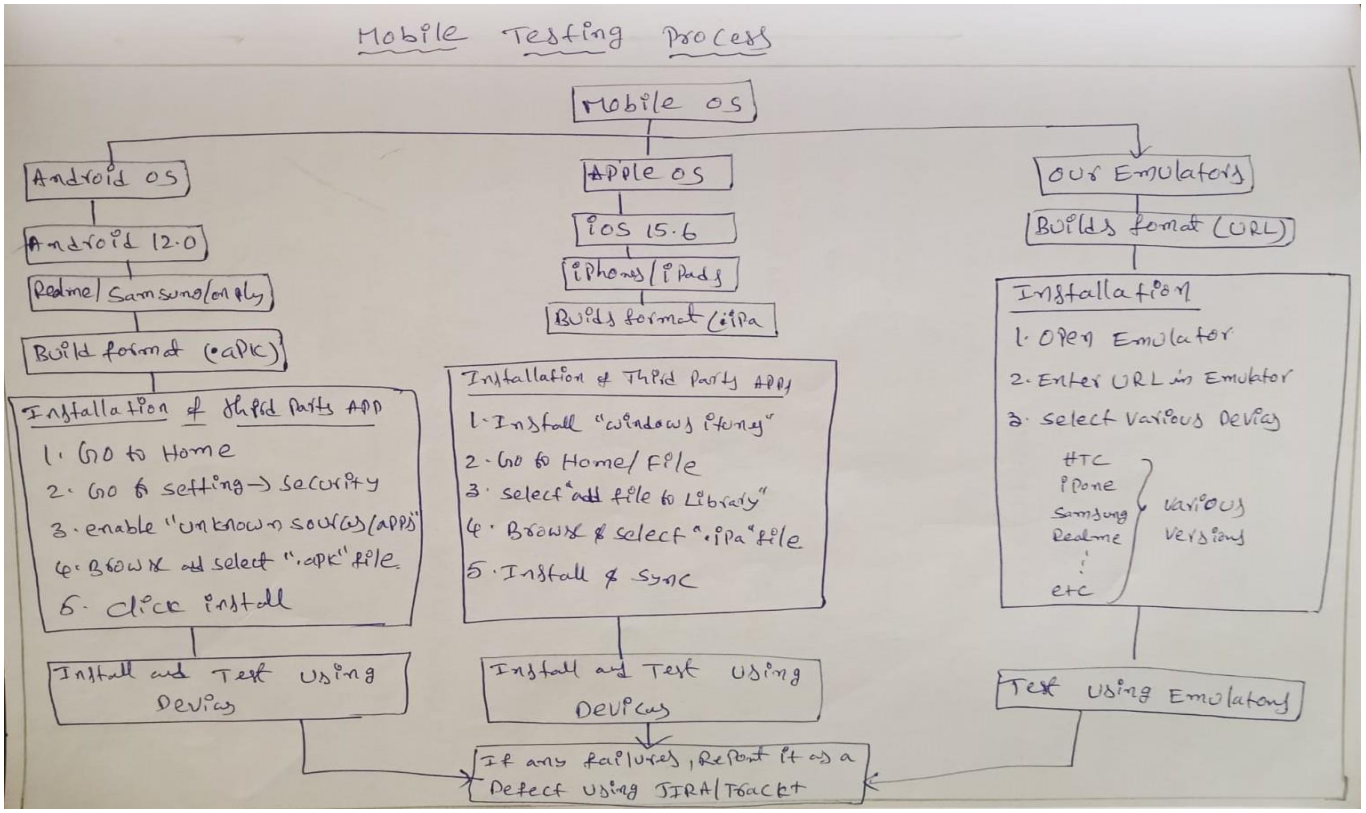
Expected Result:

Junk values should not be displayed.

Actual Result:

Junk values (%. siccy%) is displayed and [Return] is missing.

Please view attached screenshot for more information.



Approaches to Test the mobile application:

We will follow two different approaches for the testing of mobile applications. Here we will follow two approaches to test the mobile application, and those are manual testing and automated testing.

1. Manual Testing:

Manual testing is a human process. The primary focus of manual testing is on the experience of the user. The Analysis and evaluation of the application's functionality can be done through the medium of the user in an explorative process. Manual Testing ensures that the application work on the standard of user-friendliness. Manual testing is generally the time-consuming process because the process is to find out the bugs will take time. Therefore, according to the thumb rule, 20% of applications at the time of releases should be tested with the alpha and beta testing. In the rest part of the application, automated testing should be performed.

2. Automation Testing:

Automated testing is the second approach to test the mobile application. In this process, an array of test cases is set up. Automated Testing covers 80% of the testing process. The percentage is not set up, but this is the general guideline followed by the software industry. We will perform automation testing in the below scenarios:

⇒ When manual test cases are slow, then we will use Automate test cases.

⇒ The test cases which can be easily automated then we will use Automation Testing.

⇒ Automation testing is used for test cases, which is written for the frequently used functionality.

⇒ Automation testing is used for the automated test cases, which we cannot perform manually.

⇒ Automation testing is used to automate the test cases, which gives us predictable results.

Tools for Mobile Automation Testing

1. **Appium**: An open-source automation tool for mobile applications on iOS and Android platforms. It supports native, hybrid, and mobile web applications.

2. **Selendroid**: An open-source mobile app testing framework for Android. It supports both native and hybrid apps.
3. **MonkeyTalk**: An open-source automation tool for mobile testing that supports both iOS and Android platforms. It uses a keyword-driven language for scripting tests.
4. **UI Automator:** An Android testing tool provided by Google for writing functional UI tests. It is used for testing Android applications across different devices.
5. **XCUITest (for iOS) and Espresso (for Android):** Native testing frameworks provided by Apple and Google for testing iOS and Android applications, respectively.
6. **Detox:** A gray box end-to-end testing library for React Native applications. It works with both iOS and Android.
7. **Calabash:** An open-source mobile application automation testing framework that allows writing tests in Cucumber.

Types of Mobile Testing:

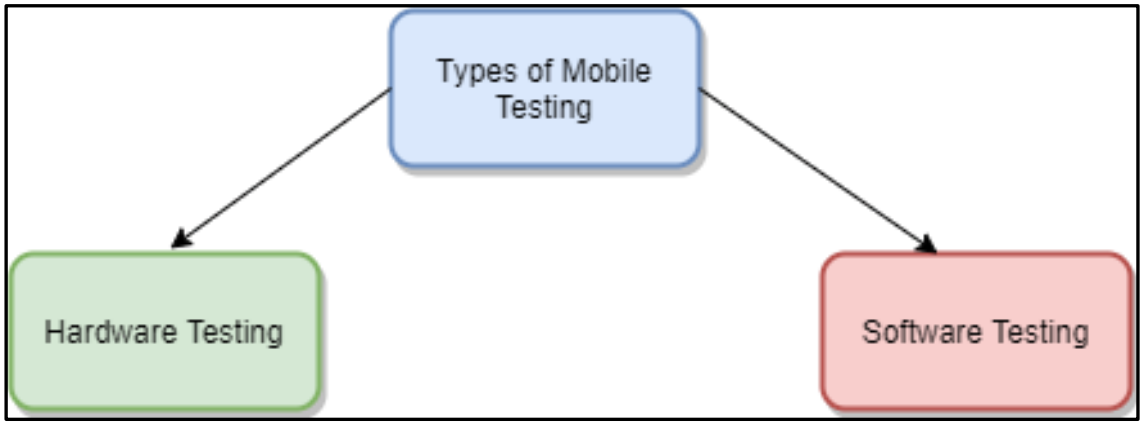
There are broadly 2 kinds of testing that take place on mobile devices:

1. **Hardware testing:**

The device includes internal processors, internal hardware, screen sizes, resolution, space or memory, camera, radio, Bluetooth, WIFI, etc. This is sometimes referred to as, simple "Mobile Testing".

2. **Software or Application testing:**

The applications that work on mobile devices and their functionality are tested. It is called "Mobile Application Testing" to differentiate it from the earlier method. Even in mobile applications, there are a few basic differences that are important to understanding:



There are three types of applications. These are as shown below:

A. **Native Apps:** A native application is created for use on a platform like mobile and tablets.
B. **Mobile web Apps:** Mobile web apps are server-side apps to access website/s on mobile using different browsers like Chrome, Firefox by connecting to a mobile network or wireless network like WIFI.
C. **Hybrid Apps:** Hybrid Apps are combinations of native apps and web apps. They run on devices or offline and are written using web technologies like HTML5 and CSS.

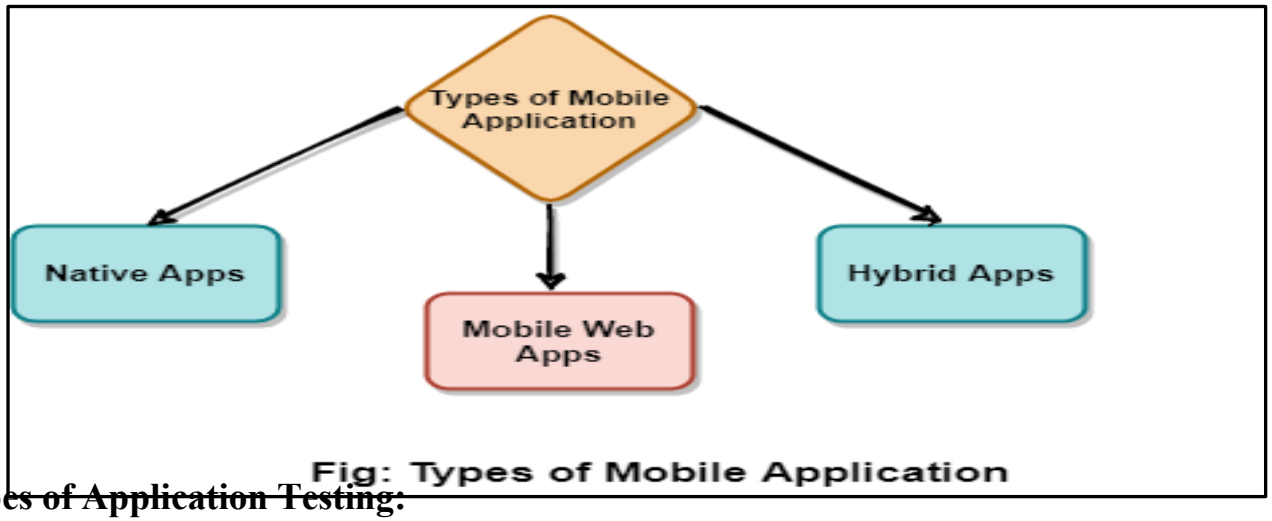


Fig: Types of Mobile Application

Types of Application Testing:

Top different Types of mobile app testing:

1. Usability testing
2. Compatibility Testing
3. Performance testing
4. Security Testing
5. Installation Testing
6. Localisation Testing
7. Functional Testing
8. Acceptance Testing
9. Interruption Testing
10. Recovery Testing

Mobile Application Testing Strategy:

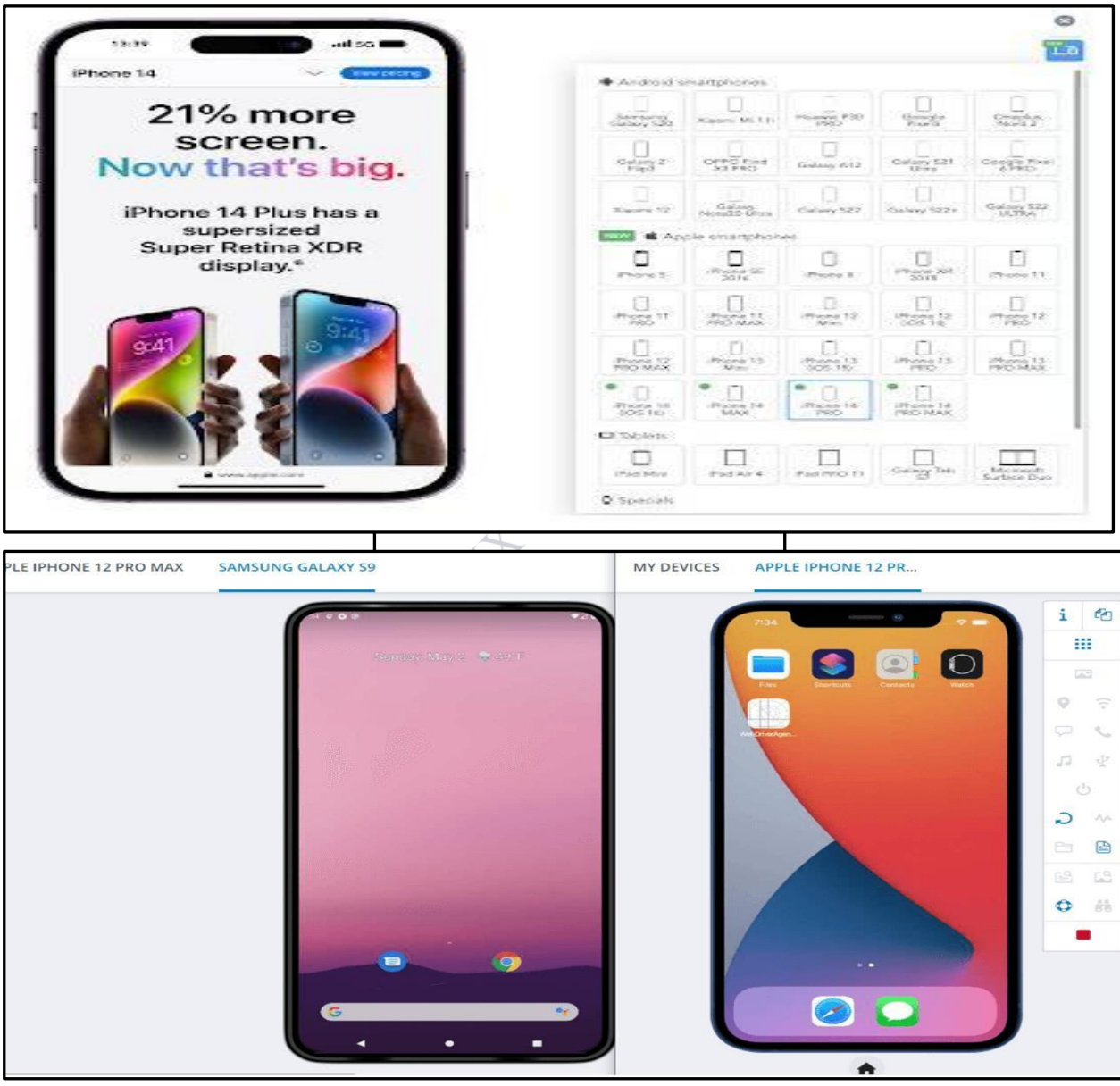
The Test strategy should make sure that all the quality and performance guidelines are met. A few pointers in this area:

1. **Selection of the devices:** Analyze the market and choose the devices that are widely used. (This decision mostly relies on the clients. The client or the app builders consider the popularity factor of certain devices as well as the marketing needs for the application to decide what handsets to use for testing.)

2. **Emulators:** The use of these is extremely useful in the initial stages of development, as they allow quick and efficient checking of the app. The emulator is a system that runs software from one environment to another environment without changing the software itself. It duplicates the features and works on the real system.

Types of Mobile Emulators:

⇒ **Device Emulator**- provided by device manufacturers.
⇒ **Browser Emulator**- simulates mobile browser environments.
⇒ **Operating systems Emulator**- Apple provides emulators for iPhones, Microsoft for Windows phones, and Google for Android phones.



Difference Between Mobile and Desktop Application Testing:

⇒ On the desktop, the application is tested on a central processing unit. On a mobile device, the application is tested on handsets like Samsung, Nokia, Apple, and HTC.
⇒ Mobile device screen size is smaller than a desktop.
⇒ Mobile devices have less memory than a desktop.

⇒ Mobiles use network connections like 2G, 3G, 4G, or WIFI whereas desktop use broadband or dial-up connections.

⇒ The automation tool used for desktop application testing might not work on mobile applications.

Agile Model (or) Agile Methodology (or) Agile Process:

⇒ Agile is basically iterative and incremental Model.

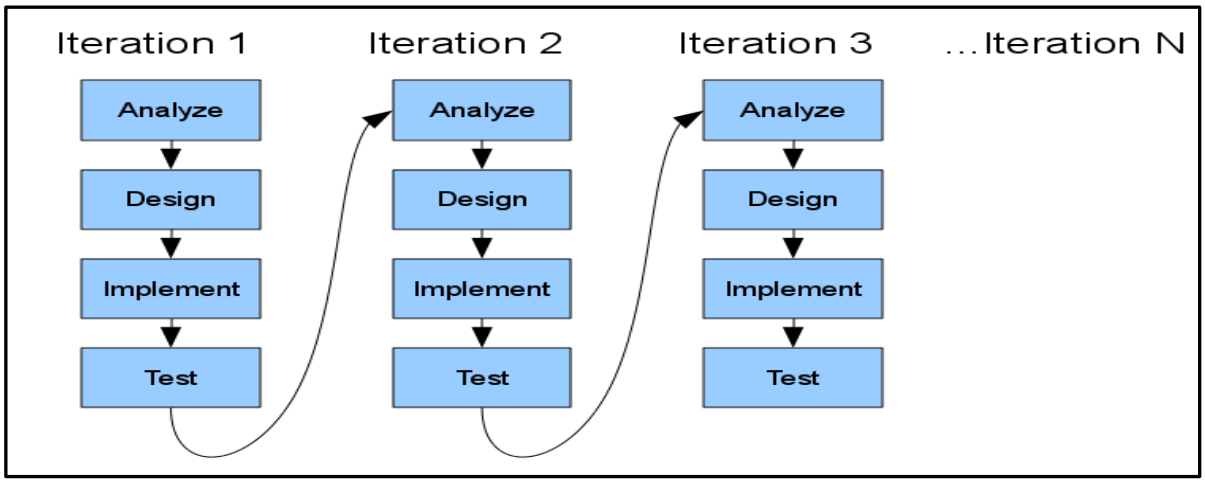
⇒ Iterative means same kind of process we are repeating again and again.

⇒ Incremental means what are the s/w we implemented with few numbers of features we keep on adding new modules and new features into that.

⇒ Agile is one of the world's most widely used and recognized software development framework.

⇒ Agile is an Iterative and Incremental Process and Agile is a software development methodology.

⇒ Agile software development allows the team to work together more efficiently and effectively in developing complex projects.



Agile Principles:

1. Customer no need to wait for long time.
2. We develop, test and release piece of s/w to the customer with few numbers of features.
3. We can accept/accommodate requirement changes.

⇒ There will be good communication b/w customer, Business Analyst (BA), Developers and Testers.

Advantages:

1. Requirement changes are allowed in any stage of development (or) we can accommodate requirement changes in the middle of development.
2. Releases will be very fast (weekly).
3. Customer no need to wait for long time.
4. Good Communication b/w team.

5. It is very easy model to adopt.

Dis-advantages:

⇒ Less focus on design and documentation Science we deliver software very fast.

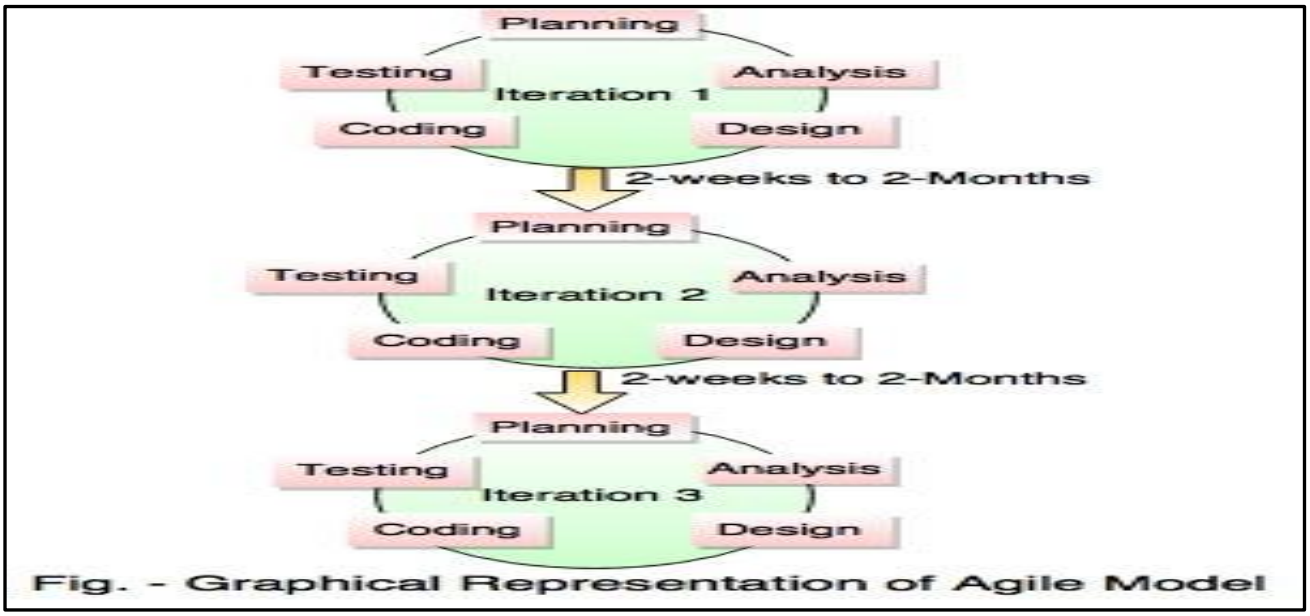
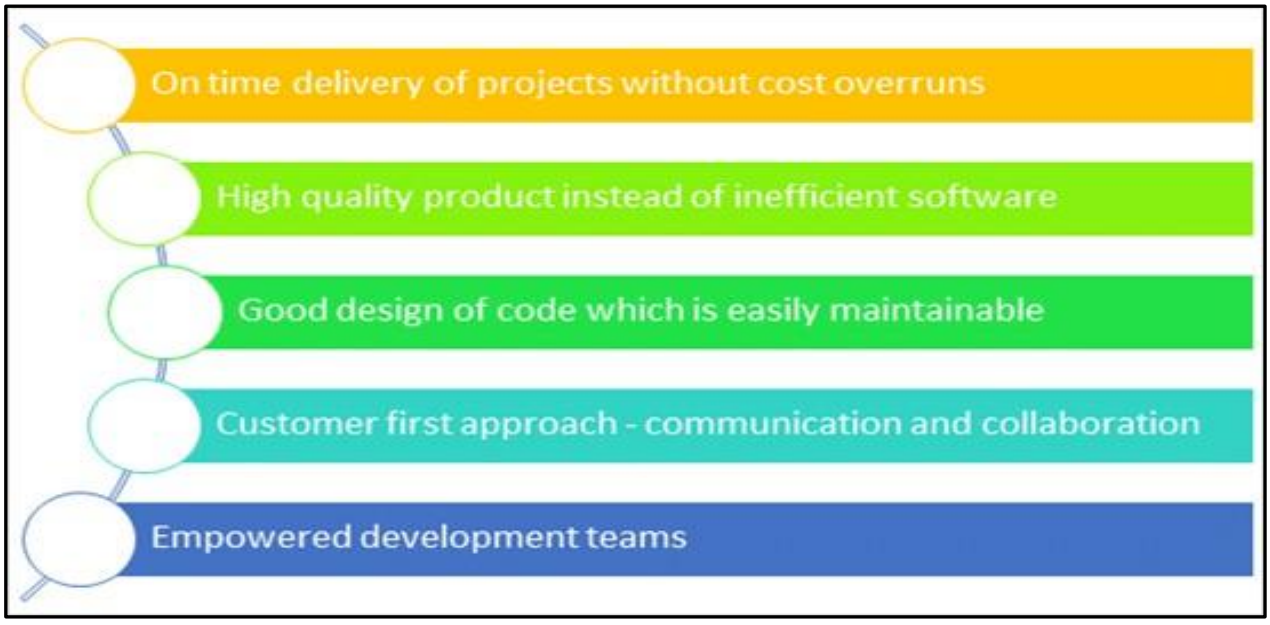


Fig. - Graphical Representation of Agile Model

What are Agile Promises:

Agile is not only about applying the set practices in developing software. It also brings in a change in the Team's mindset which drives them towards building better software, working together and eventually landing them a happy Customer.



Principles of Agile Testing:

Agile Testing includes various different principles that help us to increase the productivity of our software.

1. Constant response

2. Less documentation
3. Continuous Testing
4. Customer Satisfaction
5. Easy and clean code
6. Involvement of the entire team
7. Test-Driven
8. Quick feedback

1. Constant Response:

The implementation of Agile testing delivers a response or feedback on an ongoing basis. Therefore, our product can meet the business needs. In other words, we can say that the Product and business requirements are understood throughout the constant response.

2. Less Documentation:

The execution of agile testing requires less documentation as the Agile teams or all the test engineers use a reusable specification or a checklist. And the team emphases the test rather than the secondary information.

3. Continuous Testing:

The agile test engineers execute the testing endlessly as this is the only technique to make sure that the constant improvement of the product.

4. Customer Satisfaction:

In any project delivery, customer satisfaction is important as the customers are exposed to their product throughout the development process. As the development phase progresses, the customer can easily modify and update requirements. And the tests can also be changed as per the updated requirements.

5. Easy and clean code:

When the bugs or defects occurred by the agile team or the testing team are fixed in a similar iteration, which leads us to get the easy and clean code.

6. Involvement of the entire team:

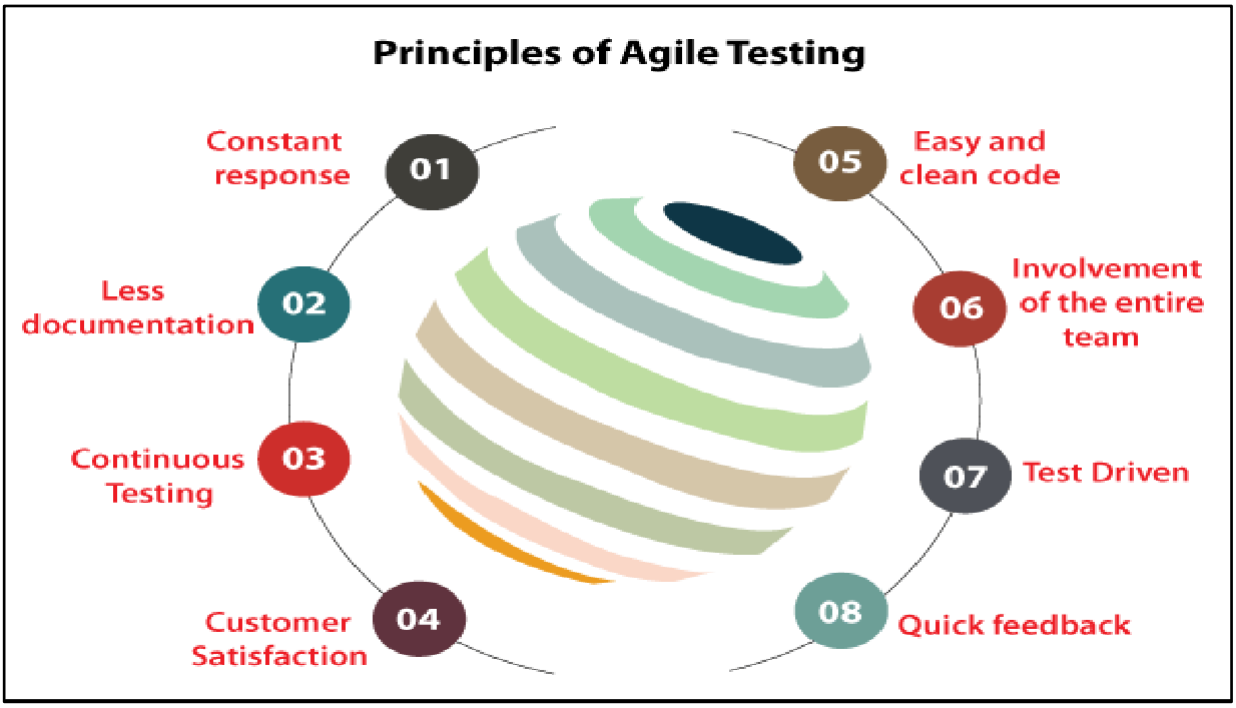
As we know that, the testing team is the only team who is responsible for a testing process in the Software Development Life Cycle. But on the other hand, in agile testing, the business analysts (BA) and the developers can also test the application or the software.

7. Test-Driven:

While doing the agile testing, we need to execute the testing process during the implementation that helps us to decrease the development time. However, the testing is implemented after implementation or when the software is developed in the traditional process.

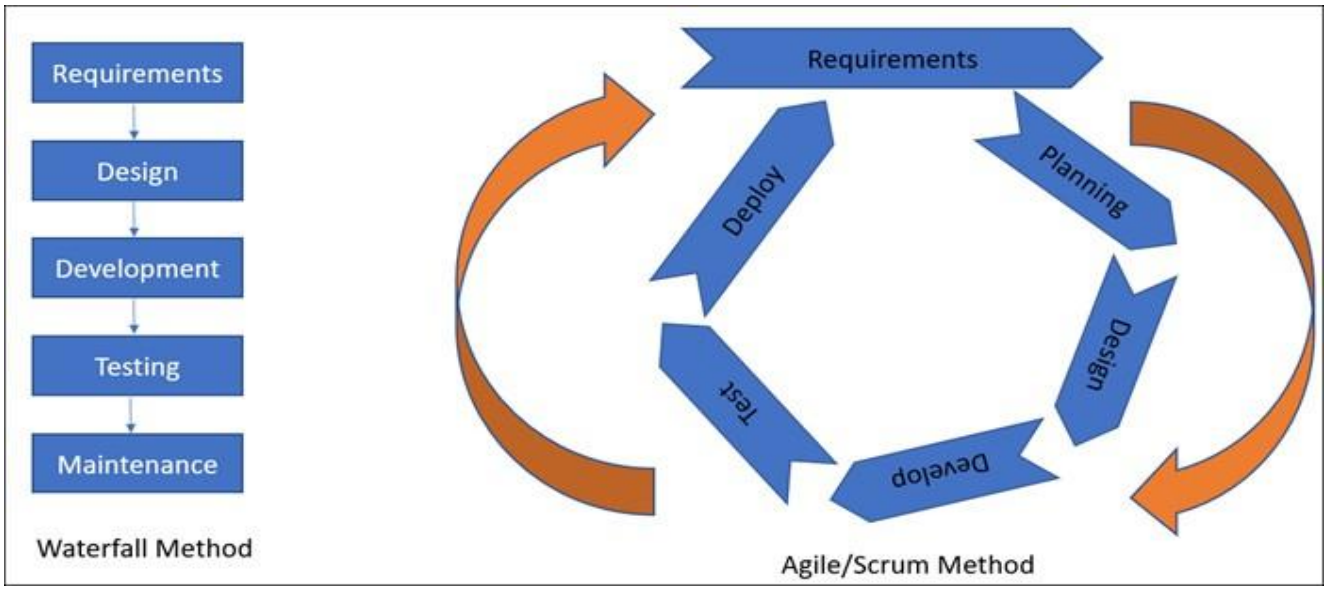
8. Quick response:

In each iteration of agile testing, the business team is involved. Therefore, we can get continuous feedback that helps us to reduces the time of feedback response on development work.



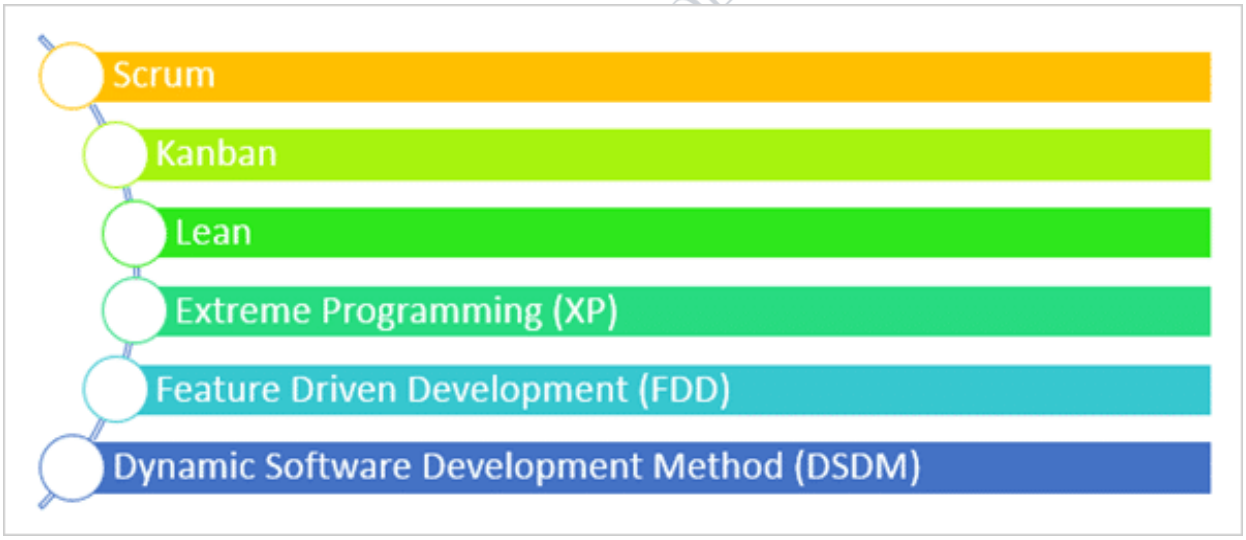
Difference Between Traditional Testing and Agile Testing:

Traditional Testing (Waterfall Method)	Agile Testing
Testing is delinked from development, carried out as separately in the end.	Testing happens along with development, making delivery of projects in shorter cycles a reality.
Testers work independently and they never mingle with developers.	The testing team is part of an agile core team and they mingle with the development team closely.
Tester's experience is not utilized in any development activities.	Testers are involved right from the requirement capturing phase and their inputs are also factored in building a rich User interface (UI) and functionalities.
Progresses as per a firm test plan.	The test plan is flexible to accommodate changes in the requirement.
An exclusive phase of testing consumes time and hence results in delayed delivery.	No delays since coding and testing go together.
Full importance and focus for testing are not visible.	Imbibed into the development process and practiced ruthlessly.
There is the possibility of defects in the software delivered.	Defects in the software-controlled efficiently.



Types of Agile Methodologies

There are several agile methodologies in practice across the world. We are going to learn more in detail about four of the most popular ones.



Scrum:

⇒ Scrum is a framework through which we build software Product by following Agile principles.
⇒ Scrum includes group of people called as scrum team.
⇒ Normally contains 5-9 members.

Main Roles in Scrum Process:

1) Product owner.
2) Scrum Master
3) Dev Team

4) QA Team.

1) Product Owner:

⇒ Define the features of the product. In the form of user stories.
⇒ He is the person always contact (First contact) to the customer/client and get the requirements from customer.
⇒ Prioritize features according to market value.
⇒ Example, here he has 100 features and he decides what are the features are implemented in first cycle, second and continues.
⇒ He will adjust features and priority every iteration, as needed.
⇒ He will assign the features and functionalities of developers and testers.
⇒ Accept or reject work results.
⇒ Once the piece of software is developed and tested, they give demo to the product owner what are the features developed and tested and the product owner likes those features he can accept it suppose he doesn't like it something is different or something is not working reasons he can also reject it.

2) Scrum Master:

⇒ Scrum master is different (Specific) role or different person and he is not a developer, not a tester and he is not manager and any other management activities.
⇒ The main role is facilitating and driving the agile Process.
⇒ He makes sure everybody is following the process or not.
⇒ My Project Manager is my scrum master.
⇒ Scrum master is aware of entire Agile Process.
⇒ Most of the meetings will be conducted or organized by the scrum master.

Responsibilities of the Scrum Master:

Scrum master works with the product owner and follows activities:

⇒ Mentor the members of the team about their roles and contribution to the team, understanding about agile processes and how to implement them while working, sense of product ownership, and self-management in the team.
⇒ Organize daily stand-up meets to get the updates about allocated tasks to the team members–task completed, the task scheduled today, and any obstacles to the member during work.
⇒ Share information about product backlog. This includes a list of work that needs to be accomplished by the team to the product owner.
⇒ Remove obstacles/roadblocks like attending meetings etc. by team members that prevent them concentrate on their task.
⇒ Monitor scrum principles and practices are followed by team members during development activities.

3,4) Dev and QA Team:

⇒ Developers and QA Team is responsible for developing and testing the software.

Scrum Terminology:

Story: A story is representing the Functionality.

User Story: A feature or module in a software.

EPIC:

⇒ Collection of user stories.
⇒ EPIC can be derived in multiple stories.
⇒ EPIC is a large requirement.
⇒ User story is a small requirement
⇒ User stories will be derived from Epic's.
⇒ These things will be Prepared by product owner.

Product Backlog:

⇒ Contains list of user stories.
⇒ Prepared by product owner.
⇒ Product owner will prepare the product backlog by taking all the requirement from the customer and convert them into stories and epic's they will put them of a list as an Excel document or some ware and then we can call is as Product Backlog.

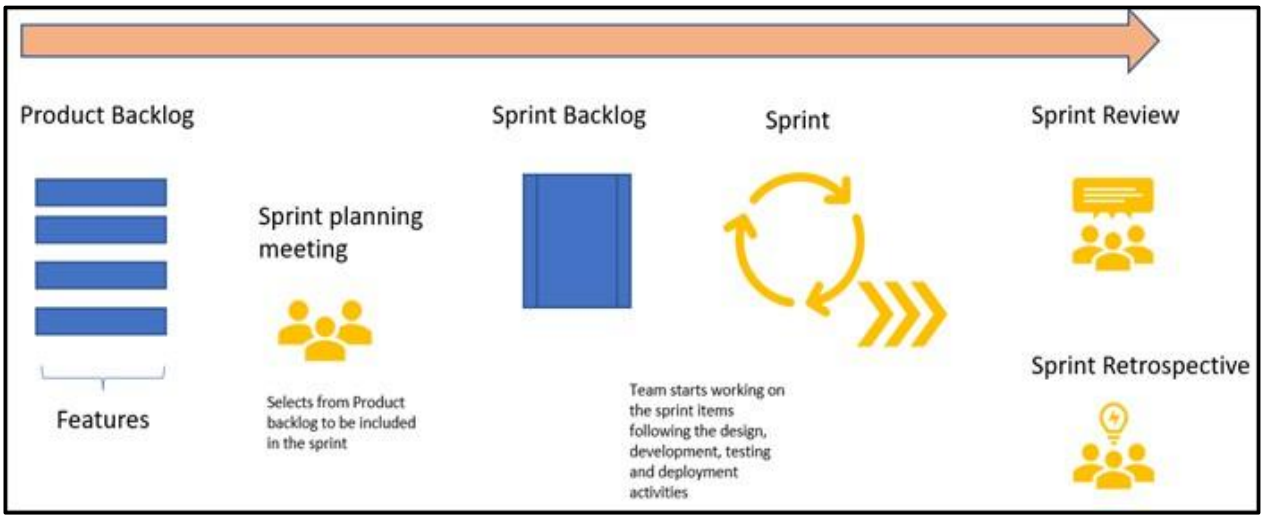
Scrum Events:

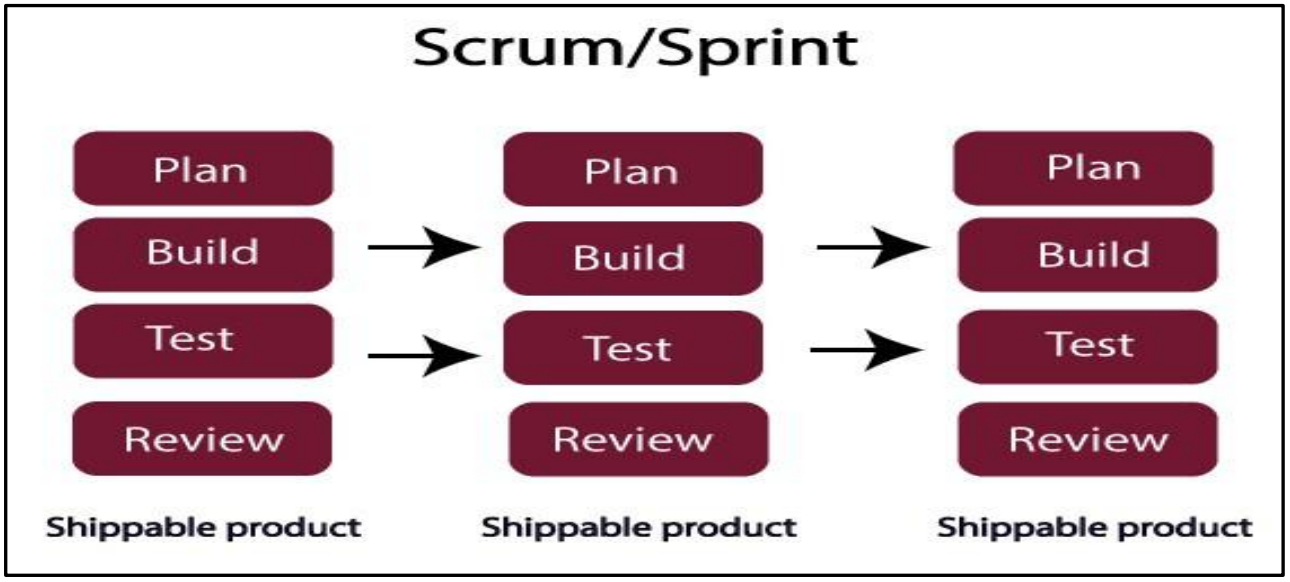
There are five events in a scrum:

1. **Sprint Planning:** This event happens at the start of the sprint where the scrum team discusses the items in the product backlog and decides which ones will form part of

the current sprint. The deliverable of the sprint planning meeting is the sprint backlog.

2. **Daily Scrum:** Daily scrum is timeboxed to 15 minutes and is held usually at the start of the day. In the daily scrum, the scrum team discusses the sprint backlog items to understand their progress or any issues/blockers that need to be resolved.
3. **Sprint**: The sprint itself is the duration in which all the sprint backlog items are delivered. This is usually 2 weeks but can be a month.
4. **Sprint Review:** Sprint review is held at the end of the sprint where the team reviews the product with the stakeholders. The sprint review is a maximum of 4 hours.
5. **Sprint Retrospective:** This meeting reviews the last sprint to answer questions: What went well and what could be improved and then develops a plan to improve. The sprint retrospective is generally 1 hour to 3 hours.

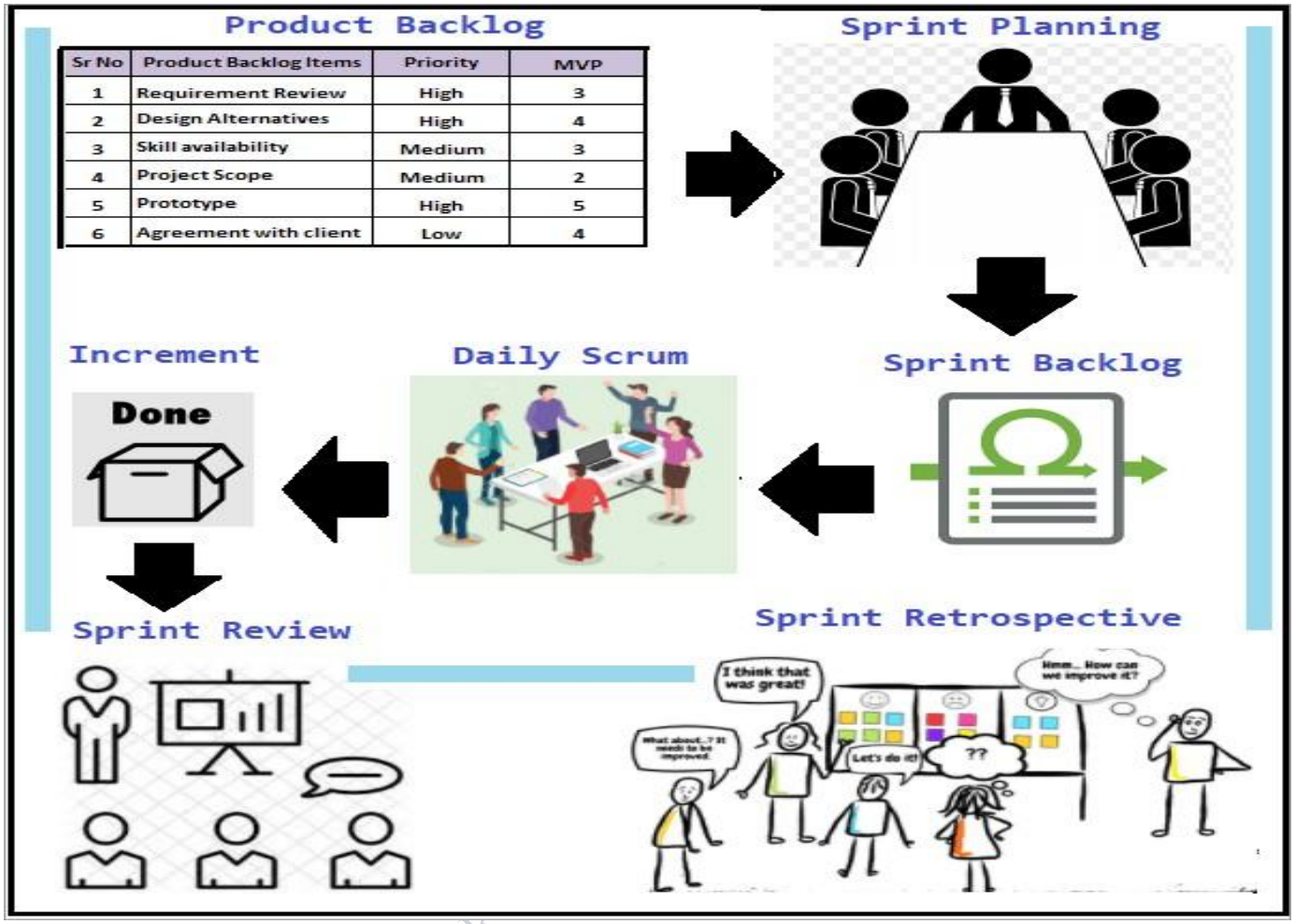




Scrum Framework:

It is an agile project management framework used to develop, deliver and sustain software products to the customers. Scrum framework adapts to changes in client requirements,

develops software through a series of iterations (known as sprint), and delivers a more frequent, quicker, and high-quality product with flexibility for change.



Sprint (Iteration):

⇒ Period of time to complete the user stories Sprint is decided by the Product owner and team, usually 2-4 weeks of time.

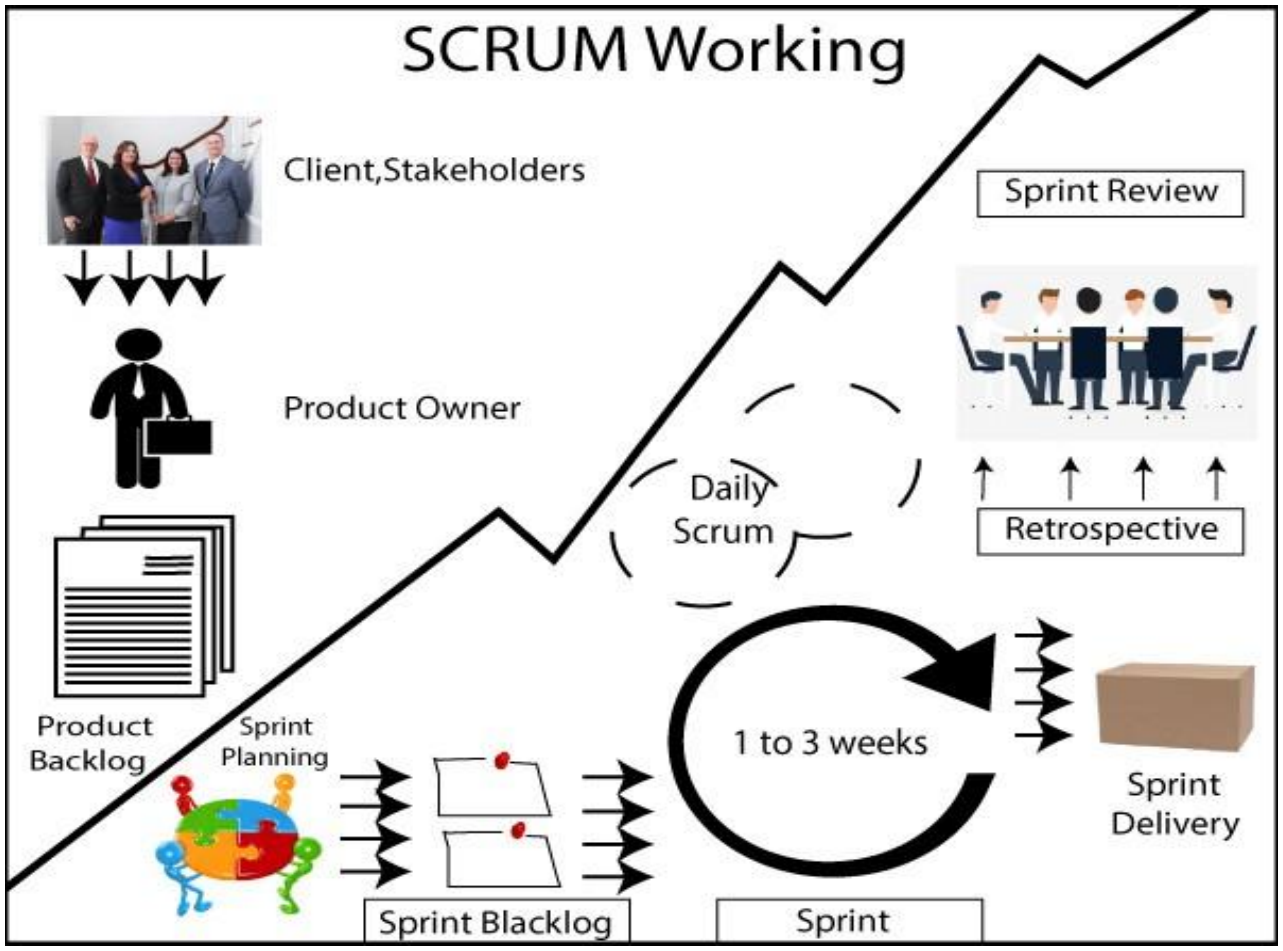
⇒ Product owner will not decide how many stories should be developed in one particular sprint, so he will sit with the team and all the team members together and will decide.

⇒ For example, here we have 100 stories developers and test engineers will not develop all stories with in one sprint, they will pick some stories (5 to 10) and they will develop and test it and at end of the sprint deliver a piece of software to the customer.

⇒ We can call is as first sprint, second sprint or sprint 1, Sprint 2…….

⇒ Period of time to complete the user stories, it means we should complete the design, development and Testing and everything, the entire process should be completed for that particular feature.

⇒ Sprint time is minimum 2 weeks and maximum 4 weeks.



Sprint Planning Meeting:

⇒ Meating conducts with the team (Product owner, Scrum master, Dev Team, Testing Team).

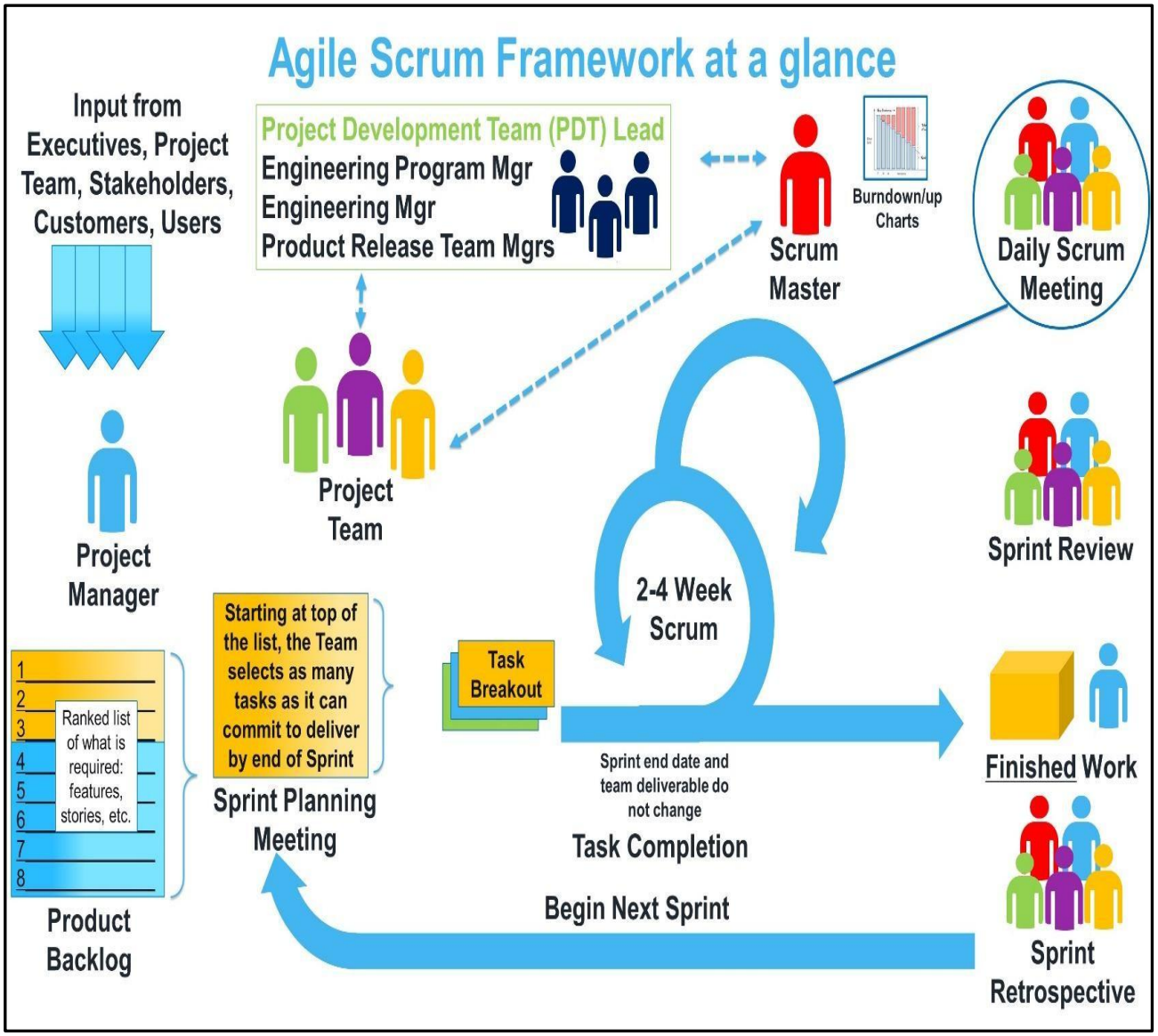
⇒ Define what can be delivered in the sprint ad duration.

⇒ During sprint planning meeting team will decide what are the stories we have to develop and test and release to the customer in that particular sprint.

Two Things will be main focus in sprint Planning meeting:

⇒ First thing is how many stories we have in the backlog and how many stories we are going to develop and Test during the sprint.

⇒ Second Thing is what is the duration of the sprint most of the times it will be one day meeting. whenever we start project in the first day it self you will have sprint planning meeting with the team.



Sprint Back log:

⇒ List of committed stories by Dev or QA for specific sprint.

⇒ What are the stories we are going to develop and Test in the particular sprint those stories are comes under sprint backlog.

⇒ **Product Backlog** contains all the stories from the product. Every story will be written in the product backlog.

⇒ In Sprint backlog we have only few numbers of stories which are committed by Dev and QA.

⇒ Dev and QA accepted so these are stories we are going to develop and test with in the period of time those stories are called sprint backlog.

⇒ For each sprint the sprint backlogs are changed.

Scrum Meeting (Daily Stand-up Meeting):

⇒ Meating conducted by scrum master everyday 15 mins, calls as Scrum call/Stand up meeting.

⇒ Scrum Meeting will be conducted by the scrum master every day 15 mins (Hardly 10 to

15 mins) with the team like including product owner, Developers ad Testers everybody will be involved in the meeting and within the 15 mins everybody should say the status what are the tasks they have completed yesterday and what are tasks plan today and what are tasks plan for tomorrow and if they are any backlogs while doing their task and if they face any challenges, if there is any delay we have to clearly discuss in this particular meeting.

⇒ Scrum meeting discusses status of the project.

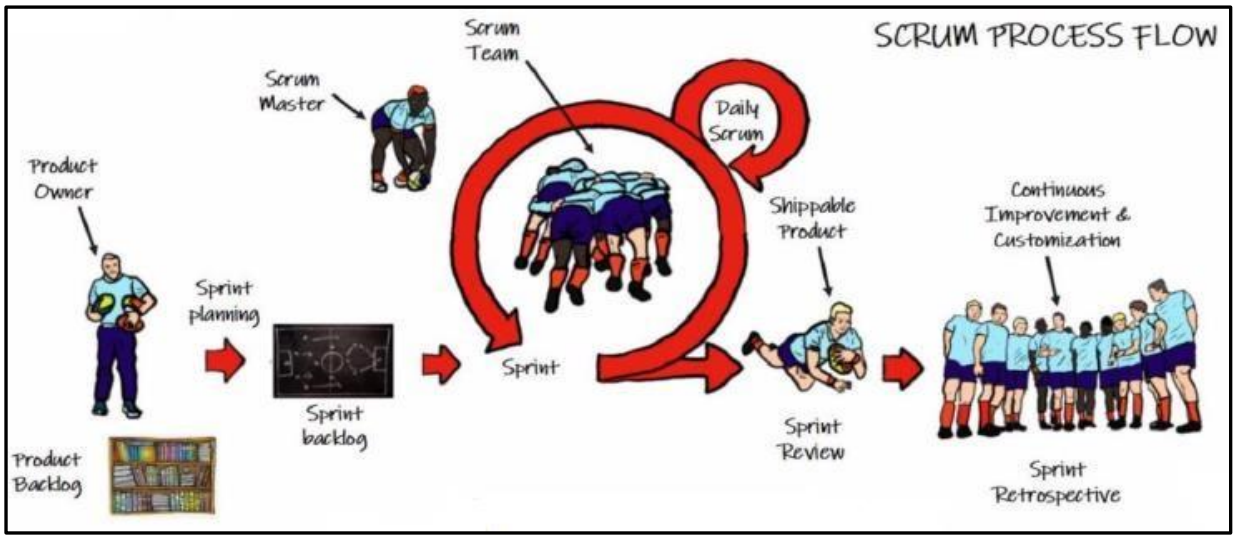
They meeting is mainly focussing on 3 Questions, everybody in the team should tell answer for these 3 questions.

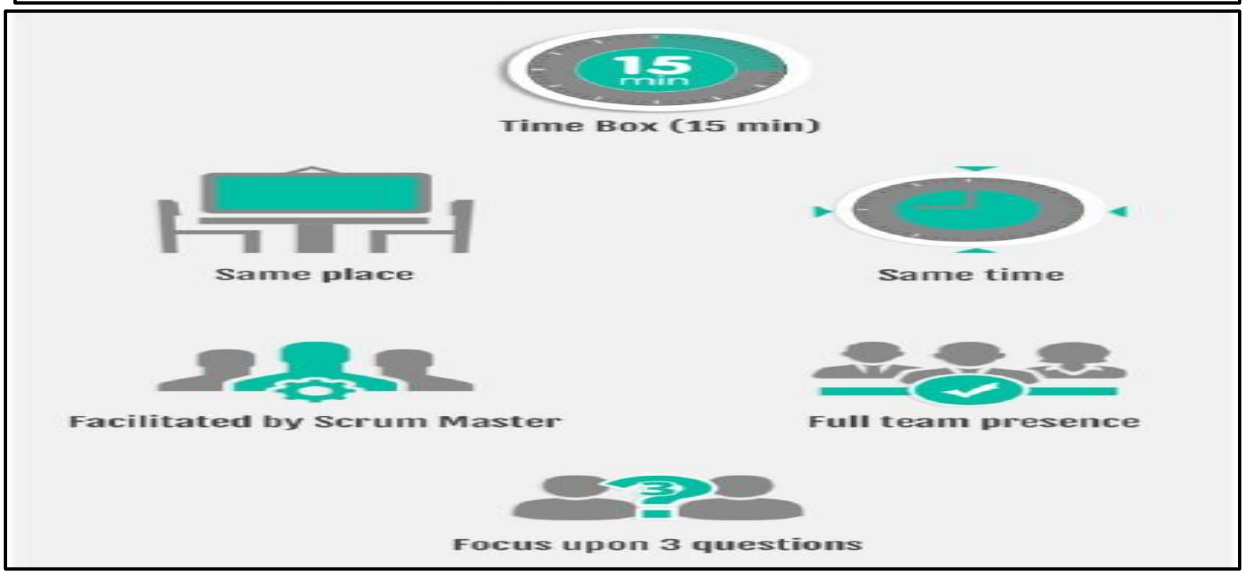
1. What did you do yesterday?
2. What will you do today?
3. What is tasks plan for tomorrow?

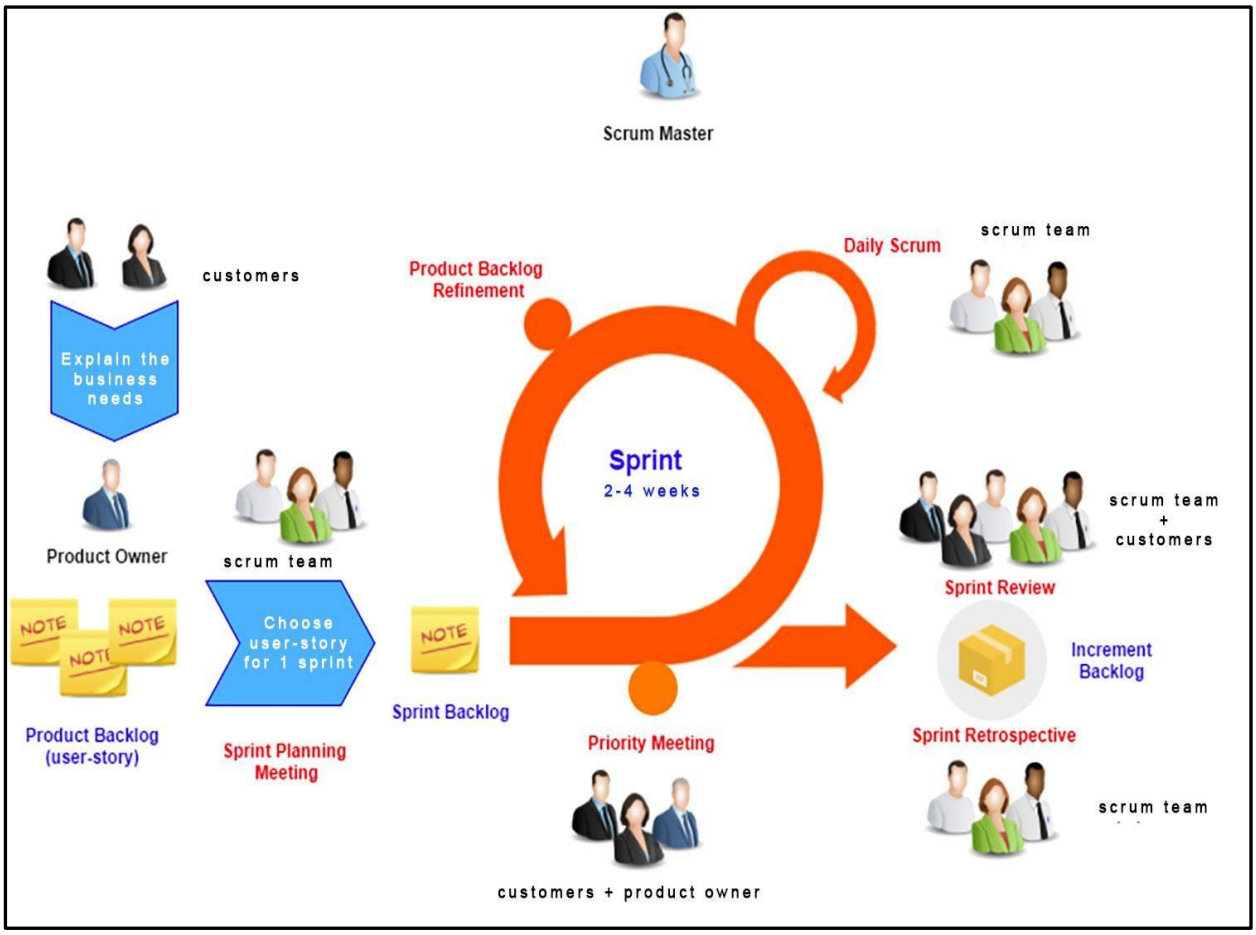
→ Are there any impediments or blockers in your way?

→ Within a sprint there are multiple scrum meetings.

→ Even solutions also we will think in that meeting itself.







Sprint Retrospective Meeting:

⇒ This meeting conducted only once after completion of every sprint.

⇒ Here we exactly discuss about the like what went well, what went wrong and what are the improvements we have to do is next sprint.

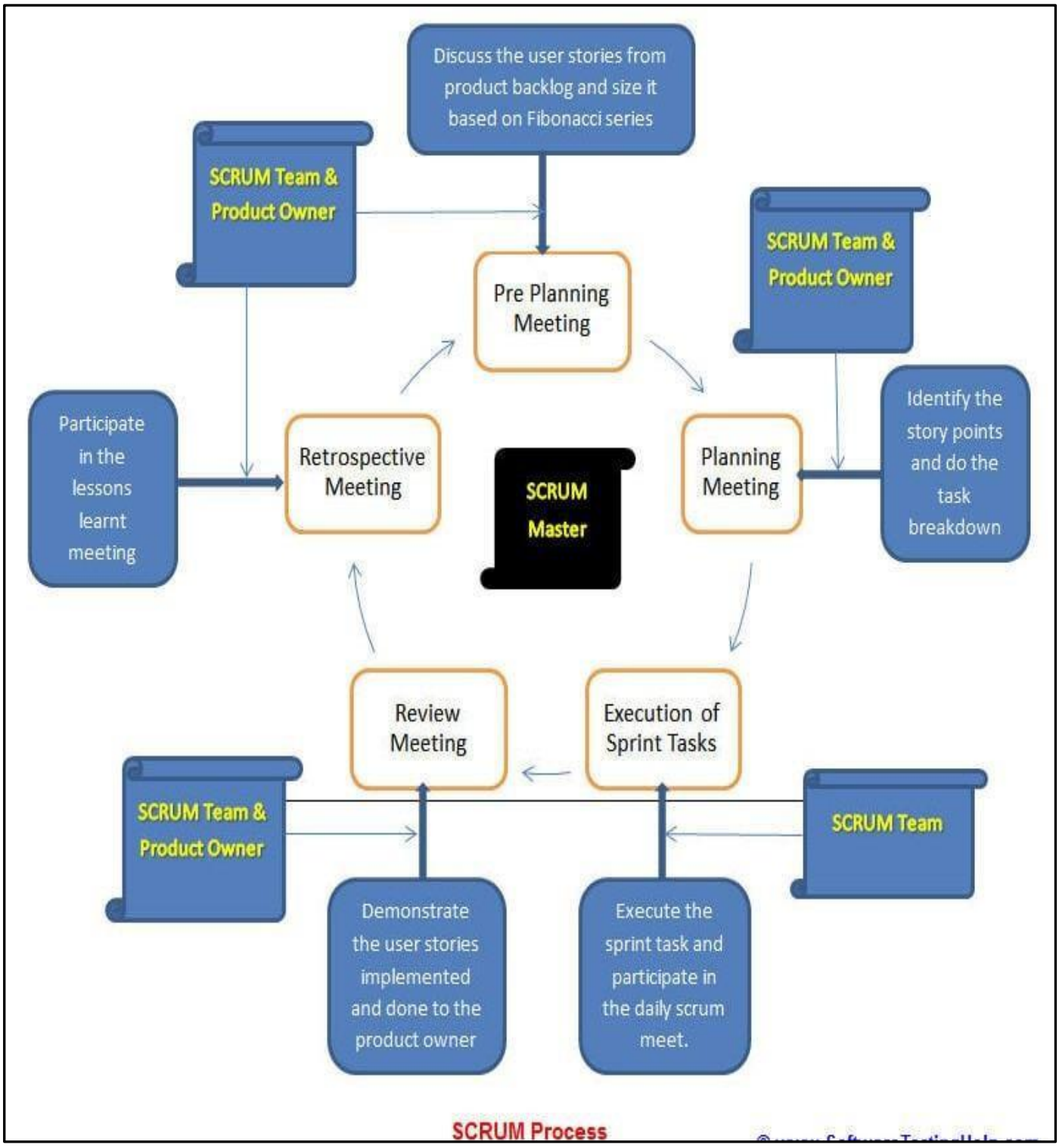
⇒ Conducts meeting, after completion of sprint.

⇒ The entire team, including both the scrum master and the Product owner along with the Dev and QA team everybody will participate this meeting.

Difference between Scrum Meeting and Sprint Retrospective Meeting:

1. The **Scrum meeting** which will happen in everyday 15 mints to discuss about the status of the project.
2. **Sprint retrospective meeting** will happen only one time after completion of the sprint to discuss what went well, what went wrong and what the improvements are needed in the Upcoming sprints.

Note: All these meetings will be conducted by the scrum master and rest of the team will be involved in that meetings.



SCRUM Process

Story Point:

⇒ Rough estimation of user stories, will be given by Dev QA in the form of Fibonacci series.

For example, Login is my story

⇒ For Login Developer says I need this much of time to design and develop the story for the Login.

⇒ QA guys say I need this much of time to write the Testcase and executing the test cases and finding the defects and retesting the defects for thy story.

Fibonacci series:

0 1 1 2 3 5 8

⇒ Some companies 1 story Point is considered which is equal to 1 hour/sometimes 1 story point is consider as 1 day (6 hours) that depends on the Company.

1 Story Point = 1 hour / 1 day (6 hours)

Login: Dev-5, QA-3 = 8 Hours / 1 Day

S.No	User Story	Story Points
1	Story 1	3
2	Story 2	5
3	Story 3	2
4	Story 4	3
5	Story 5	8
6	Story 6	5
7	Story 7	1
8	Story 8	3

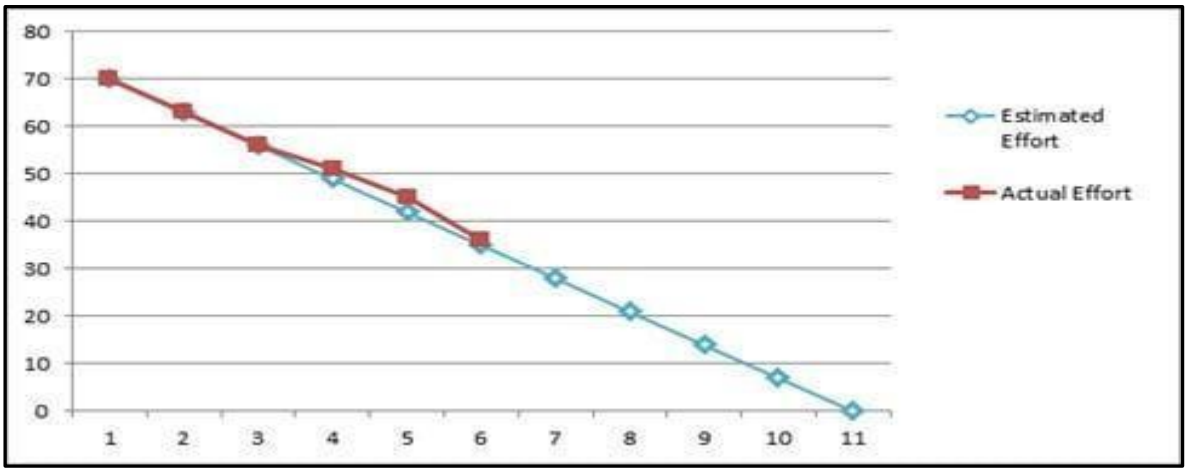
Burndown Chart:

⇒ Shows how much work remining in the sprint, Maintained by the scrum master daily

⇒ By seeing this chart, the scrum master will know exactly how much work is remaining in the sprint.

1 story --> 3 days (18 hours) (Estimation)

⇒ How much work is completed in the sprint, how much work is still pending, how the scrum master will know that, he will prepare a chart called of Burndown chat.



Sprint Review:

In sprint review we will check all the tasks we have planned during the sprint completed or not, is there any open bugs, all bugs are closed or not all test cases are executed or not these things we will review once.

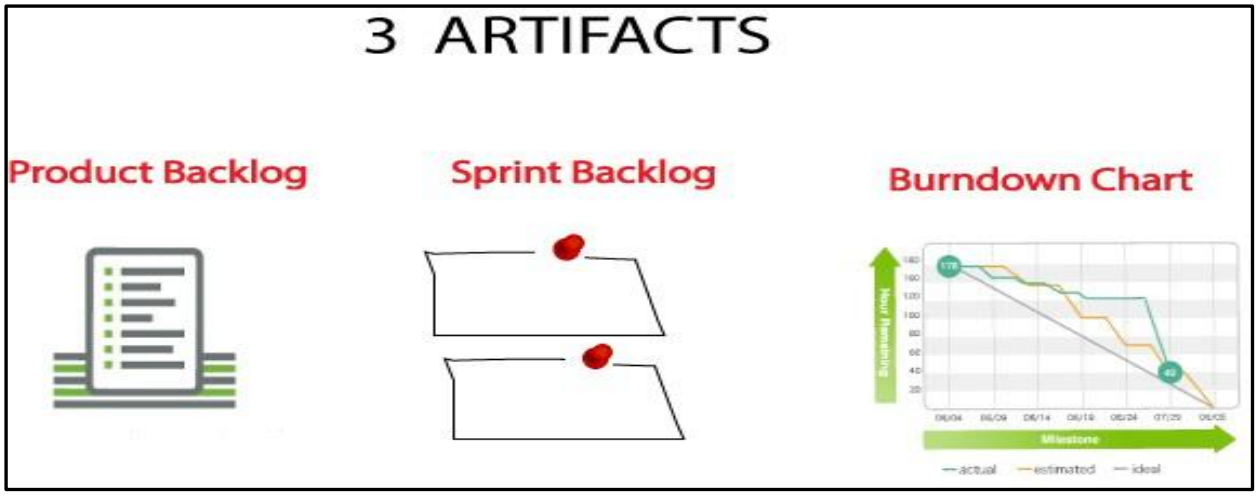
Here we will provide demo to the product owner, If the demo is excepted by the product owner then we will deliver the piece of software to the customer.

Roles:

1. Product Owner
2. Scrum Master
3. Team (Dev & QA)

Artefacts:

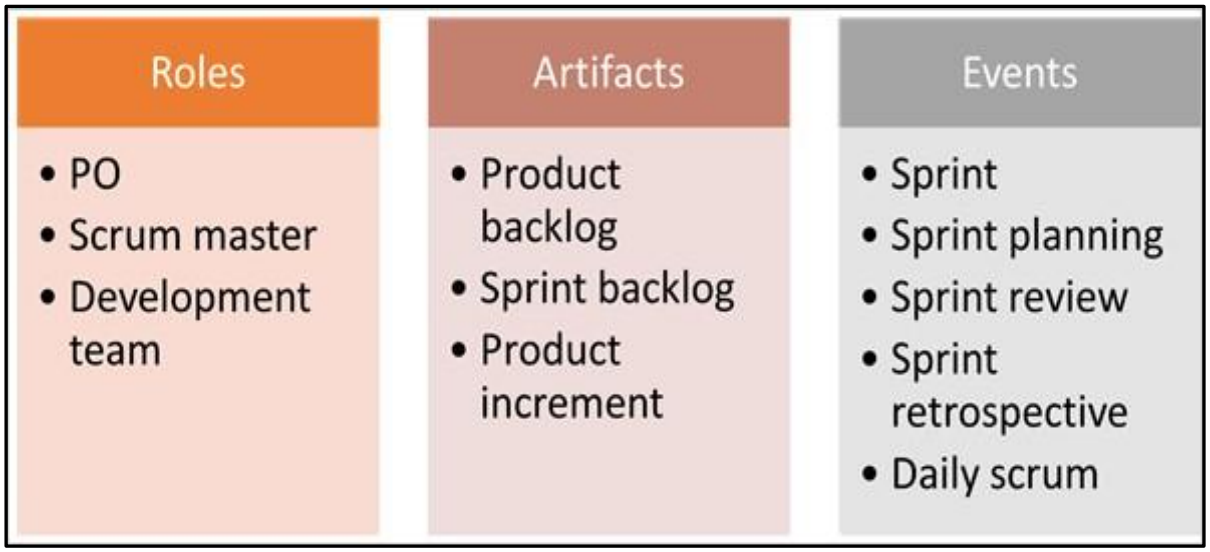
1. Product Backlog
2. Sprint Backlog
3. Burndown Chart



Ceremonies or Events:

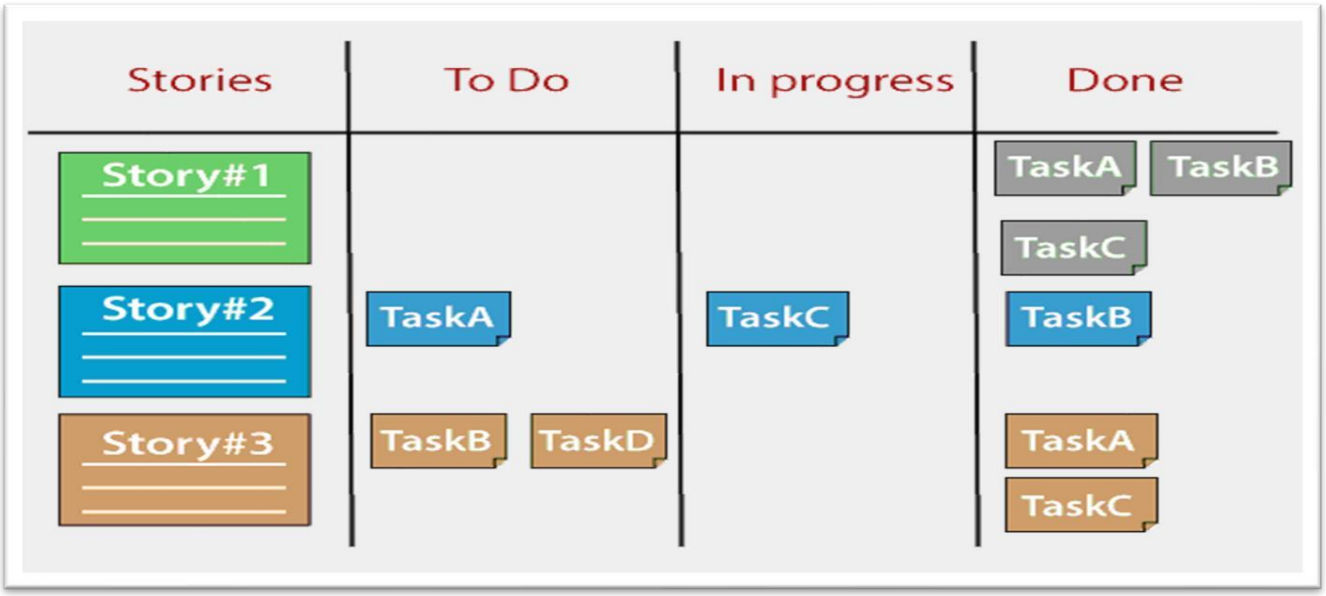
1. Sprint Planning

2. Daily scrum
3. Sprint Review



Scrum Board:

Scrum Board is a board that shows the status of all the activities that need to be done within this sprint.



Scrum Board consists of 4 statuses:

Open:

The 'Open' status means that the tasks which are available in 'Open' are not yet started.

In progress:

The 'In progress' status means the developers completed their tasks.

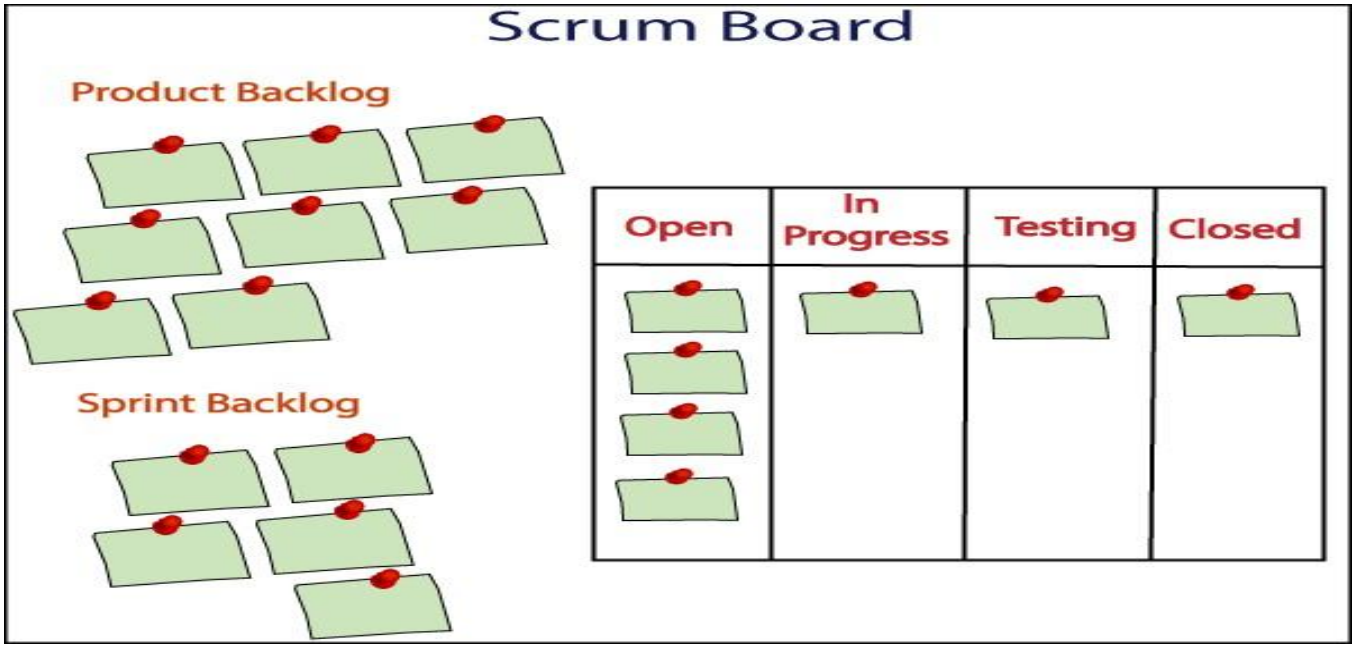
Testing:

The 'testing' means that the task is in a testing phase.

Closed:

The 'closed' means the task has been completed.

Stories	TODO	In Progress	Testing	Done
1. 2. . . . Total Stories	3 Stories (Up Coming)	2 Stories (Dev move to testing) 2 Stories (Now Process)	2 Stories (In testing)	1 story (Completed)



Definition of Ready (DOR) & Definition of Done (DOD):

DOR	DOD
User story is clear	Code Produced (all to do items in code completed)
User story is testable	Code commented, checked in and you against current version in source control
User story is feasible	Peer reviewed (or produced with pair programming) and meeting development standards
User story defined	Builds without errors
User story acceptance criteria defined	Unit tests written and passing
User story dependencies identified	Deployed to system test environment and passed system tests
User story sized by development team	Passed UAT (User Acceptance Testing) and signed off as meeting requirements

Scrum team accepts user experience artefacts	Any build /deployment / configuration changes are implemented/documented/communicated
Performance criteria identified where appropriate	Relevant documentation /diagrams produced and /or updated
Team has a good idea what it will mean to demo the user story	Remaining hours for task set to zero and task closed.

Product Backlog Sample Template:

Project Name: OpenCart (Frontend)

Client: OpenCart

Created by: Name of the product owner

Creation Date: DD-MM-YYYY

Approval Date: DD-MM-XXXX

Epic	User Story ID	Feature/ Tittle	User Story	Status	Acceptance Criteria
OpenCart Epic -001: For a new e-commerce website to launch, the highest business value will be when a new user is able to buy an item from the website	US001	Registration	As a first-time visitor to the e-Commerce website, I want register my account, so that I can login to application	New	New User should able to Register account with valid data
	US002	Login	As a Registered user, I want to login to the website, so i can see my account details etc	New	System must validate user credentials and allow login if credentials are correct
	US003	Logout	As a Registered user, I want to logout from website. so that no one else can't access my account.	New	System must logout after login

	US004	User search products	As a user, I want to be able to search items. So that I can add them to cart and do payment	New	User should able to search products and add them to cart

Sprint Planning Sample Template:

Project Name: OpenCart (Frontend)

Client: Open Cart

Created by: Name of the scrum Master

Attendees: Scrum Team

Creation Date: DD-MM-YYYY

EPIC	User Story ID	Feature/ Tittle	User Story	Story Points	Sprint
OpenCart Epic -001: For a new e-commerce website to launch, the highest business value will be when a new user is able to buy an item from the website	US001	Registration	As a first-time visitor to the e-Commerce website, I want register my account, so that I can login to application	8	1
	US002	Login	As a Registered user, I want to login to the website, so I can see my account details etc	5	1
	US003	Logout	As a Registered user, I want to logout from website. so that no one else can't access my account.	3	1
	US004	User search products	As a user, I want to be able to search items. So that I can add them to cart and do payment	5	3

Story Points	Hours
1	I Hour/Day (Depends on Company)
0,1,1,2,3,5,8	Fibonacci series

Tasks:

Developer Tasks	QA Tasks
Understanding Requirements	Understanding Requirements
Designing	Writing Test scenarios
Coding	writing Test cases
Unit Testing	Test Case Reviews
Integration Testing	Test Data Preparation
Code Review	Test Environment setup
Bug Fixes	Test Execution
Team Meetings	Re-Testing Bugg
Any other	Team Meetings
	Automation
	Any other....

Test/Project Management Tool: JIRA

JIRA Introduction:

Jira is a software development tool developed by Atlassian, primarily used for issue tracking and project management. Jira is a versatile tool that can be used by developers, testers, project managers, and other members of software development teams. It can be used in various industries and in different types of projects. Jira is easy to use and can be customized to fit the needs of any organization, thus making it a popular choice among many teams. It is widely used in software development teams to plan, track, and release software. JIRA is the main source of information for future software release. Developers can plan new features to be added and bugs to be fixed in the next release.

Jira is a cloud- and subscription-based issue tracking tool but also designed to handle team coordination in agile software development as well. Atlassian JIRA is an issue and project tracking software to plan, track and manage your projects. JIRA is mainly used by agile development teams to customize your workflows, team collaboration, and release software with confidence. It is a Defect tracking/Project Management tool by Atlassian, Inc. It is a platform-independent software.

Important Points to Note:

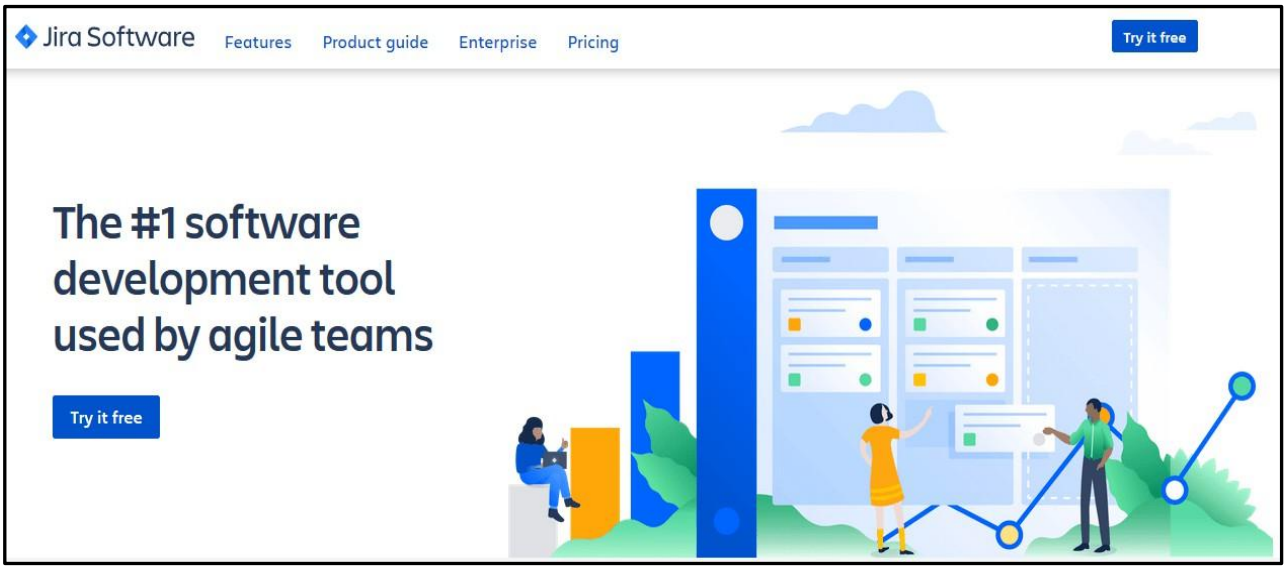
The following points explain some interesting details of JIRA.

- JIRA is an incident management tool.
- JIRA is developed by Atlassian Inc., an Australian Company.
- JIRA is a platform independent tool; it can be used with any OS.
- JIRA is multi-lingual tool − English, French, German, Japanese, Spanish, etc.
- JIRA supports MySQL, Oracle, PostgreSQL and SQL server in the backend.
- JIRA can be integrated with many other tools − Subversion, GIT, Clearcase, Team Foundation Software, Mercury, Concurrent Version System and many more.

Features of Jira:

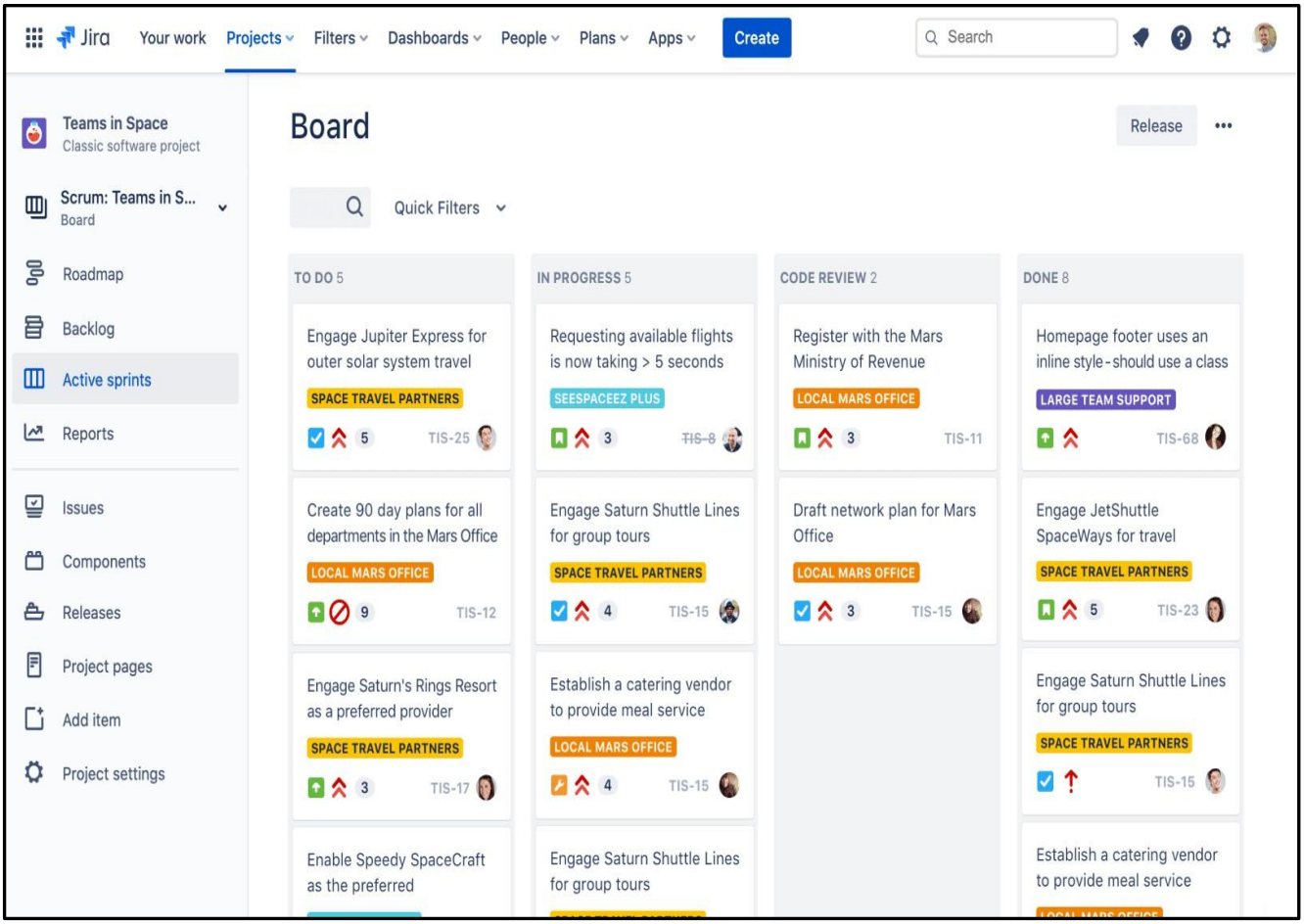
1. Agile Development:

Agile is the Jira's fundamental application, and it offers the smooth utilization of all the features of Scrum boards and Kanban boards. Therefore, it can be used for a Scrum, Kanban and hybrid method like Scrumban as well. Whenever you begin a project, Jira asks you to select your project type: Kanban or Scrum. The moment you select, Jira creates a Scrum or a Kanban board to make your project continues.



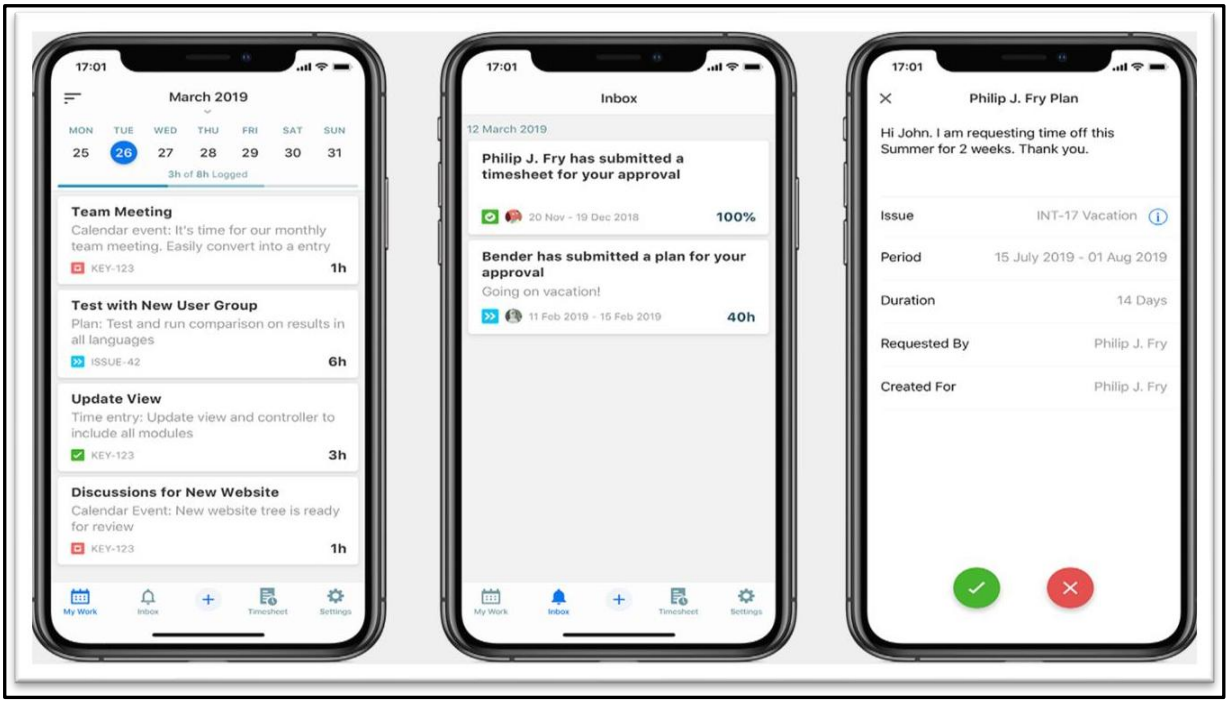
2. Jira Project Tracking:

This issue tracking software tracks ongoing project at any stage. Using JQL, the customized query language of Jira allows you to filter or sort issues based on the various criteria. The sidebar allows accessing immediate details about planning, releasing, tracking, and reporting. With this flexible planning tool, you can create tasks and stories from any screen. In addition, the drag and drop feature makes it simple to create sprints and epics in the backlog.



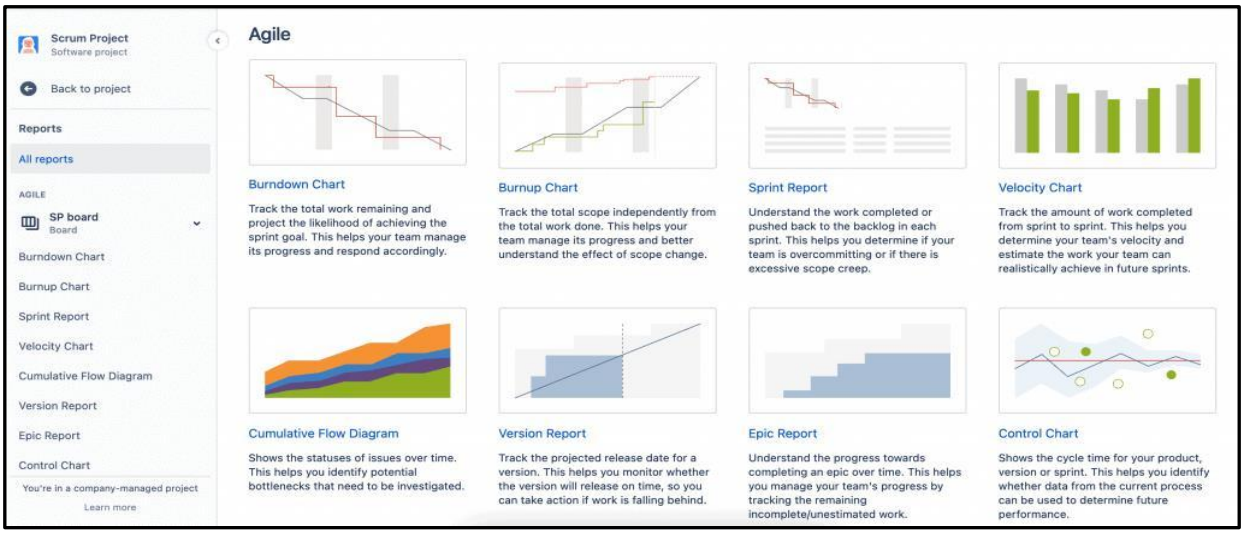
3. Mobile Applications:

Jira provides a mobile app which allows users to access and update their issues on the go. In addition to desktop and on-premise system, the tool supports remote teams on diverse locations. The Jira project management tool comes with native mobile applications that are compatible with Android and iOS devices. Hence, users can stay online as well as engaged anytime.

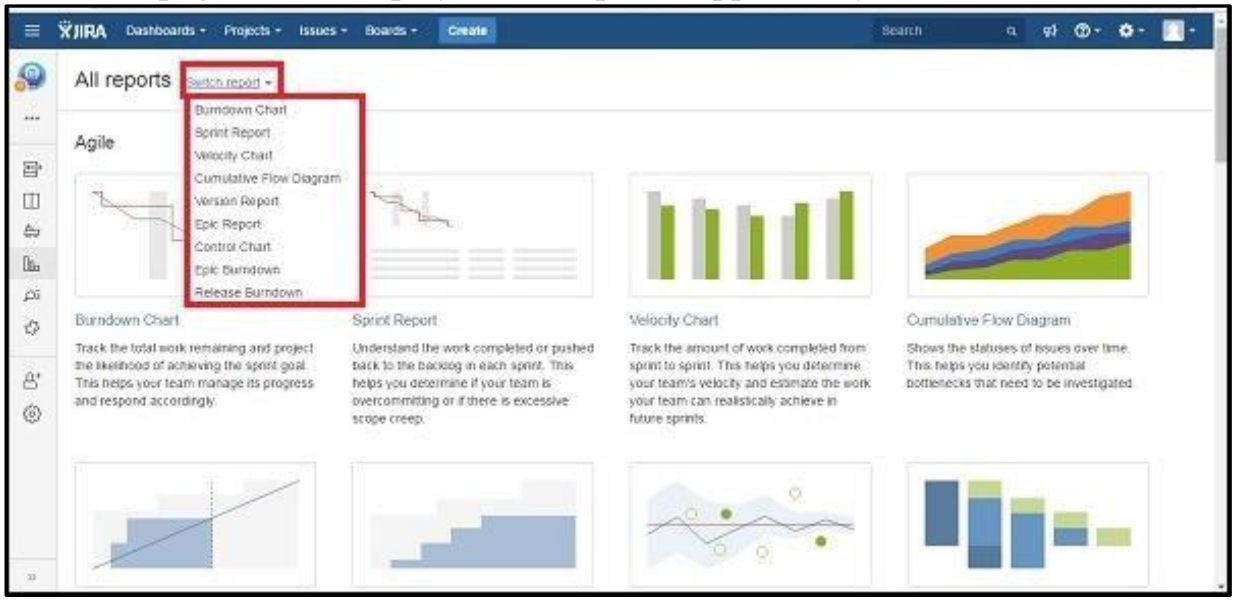


4. Reports in Jira:

Provides a wide range of reports and metrics to give teams visibility into project progress and performance. Jira delivers the relevant information in a convenient format called reports. There are numerous reports available in JIRA, which enables you to gain visibility of the situation. In addition, these reports offer project statistics throughout the entire lifecycle. For example, the Burndown chart displays the actual as well as the estimated amount of work to be finished in the sprint.



To access reports in JIRA, the user should go to Project → choose Specific project. The following screenshot shows how to navigate to a specific project. Click on the Reports icon on the left side of the page. It will display all the reports supported by JIRA.



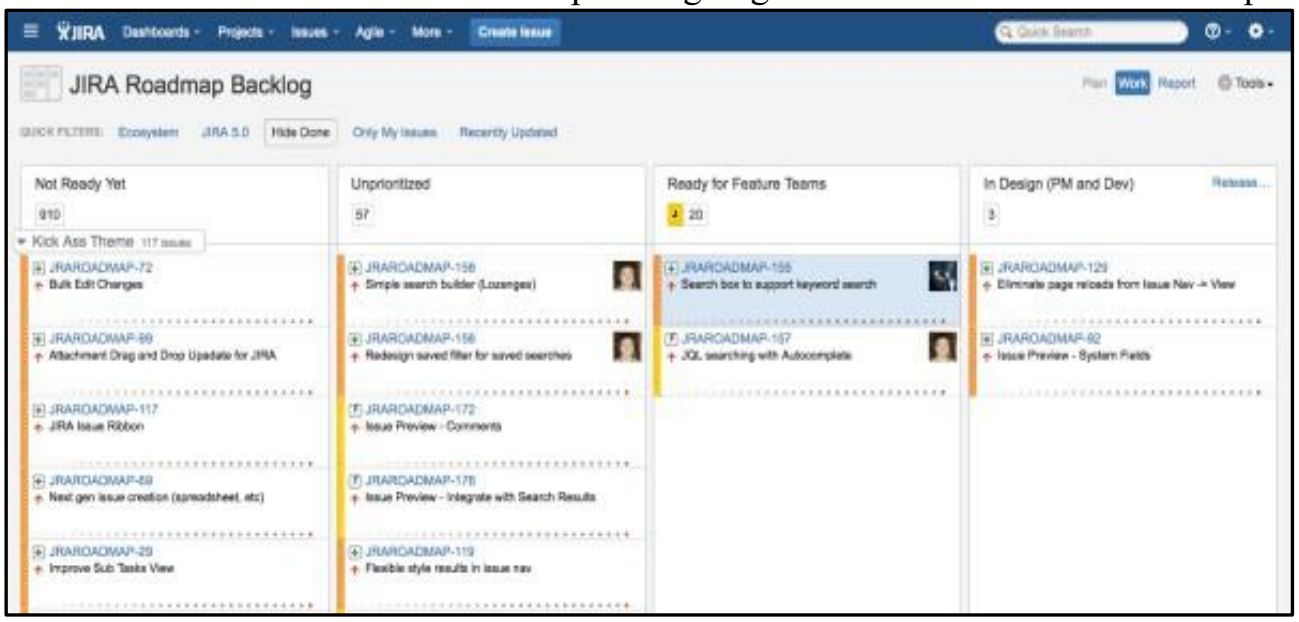
Following are the list of Agile Reports:

1. Burn down Chart: Track the total work remaining, also whether sprint is achieving the project goal or not.
2. Sprint Chart: Track the work completed or pushed back to the backlog in each sprint.
3. Velocity Chart: Track the amount of work completed from sprint to sprint.

4. Cumulative Flow Diagram: Shows the statuses of issues over time. It helps to identify high-risk issues or unresolved important issues.
5. Version Report: Track the projected release date for a version.
6. Epic Report: Shows the progress towards completing an epic over a given time.
7. Control Chart: Shows the cycle time for the product, its version or the sprint. It helps to identify whether data from the current process can be used to determine future performance.
8. Epic Burn Down: Track the projected number of sprints required to complete the epic.
9. Release Burn Down: Track the projected release date for a version. It helps to monitor whether the version will release on time, so mandatory action can be taken if work is falling behind.

5.Roadmaps:

Jira project management includes roadmap where users can create, handle, and visualize epics of their team. It enables you to comprehend what is outstanding and when bugs are scheduled to be fixed. This feature is beneficial for planning large sets of stories across various sprints.



6. Jira Security:

Jira provides advanced security features such as two-factor authentication, IP restriction and more, to secure the data and access. The security settings of Jira bug tracking software restricts the access of certain bug to only those people who are allowed to work on the bug or a team member of the given security level. You can set your bug's security level when it is created or when it is being edited.

7.Real-Time Notification:

Equipped with notification features, Jira ensures to offer the required information to its users when they indeed need it. There are configurable email alerts when the issues are updated and there are optional emails to send the remainder for overdue tasks.

8.Jira Dashboard:

The dashboard is the first thing that you can see once you log onto your Jira software. The admin can customize the dashboard's view and the things displayed on it. A dashboard typically displays apps and gadgets that expose various sorts of information to support the team members to track their project's progress.

9. Templates:

Allows users to create and use templates to quickly create issues with the same fields and settings.

10. Multi-languages support:

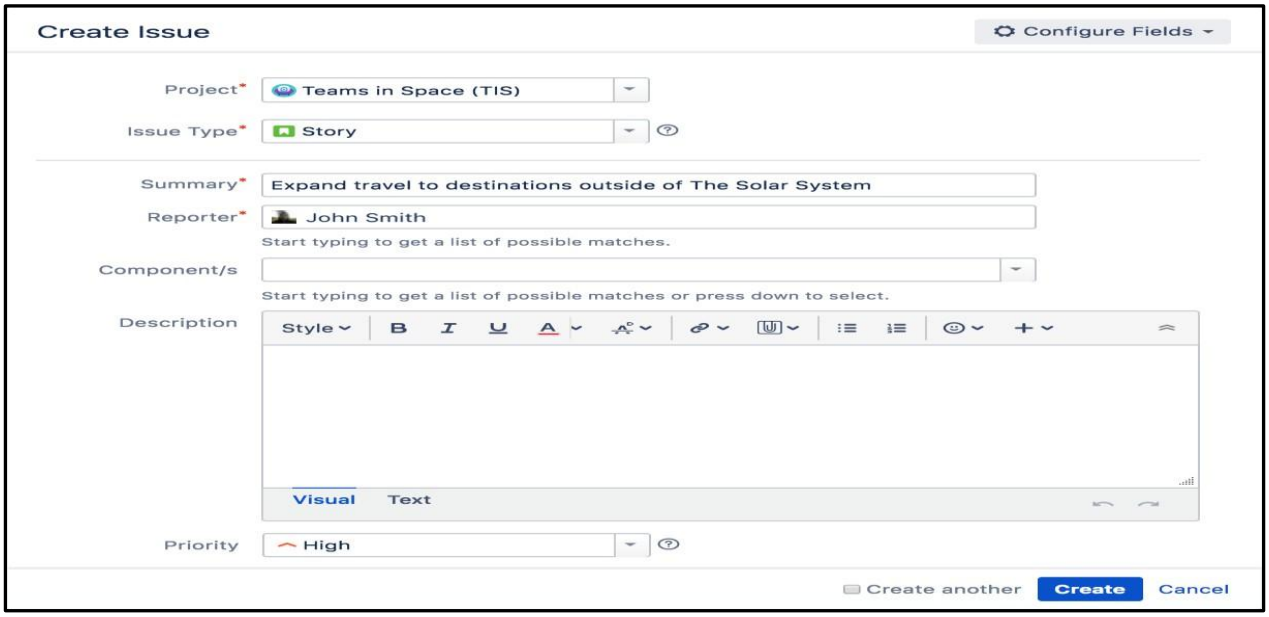
JIRA supports more than 10 languages those are widely used as English (US, UK, India), French, German, Portuguese, Spanish, Korean, Japanese and Russian.

These are some of the key features of Jira, but depending on the version and add-ons, there may be other features available.

Defect Reporting using Jira:

Jira is widely used for defect reporting in software development teams. The process of reporting a defect using Jira typically involves the following steps:

1. **Create a new issue:** Users can create a new issue by clicking on the "Create" button in Jira. They will then be prompted to select the issue type, which in this case would be "Defect."

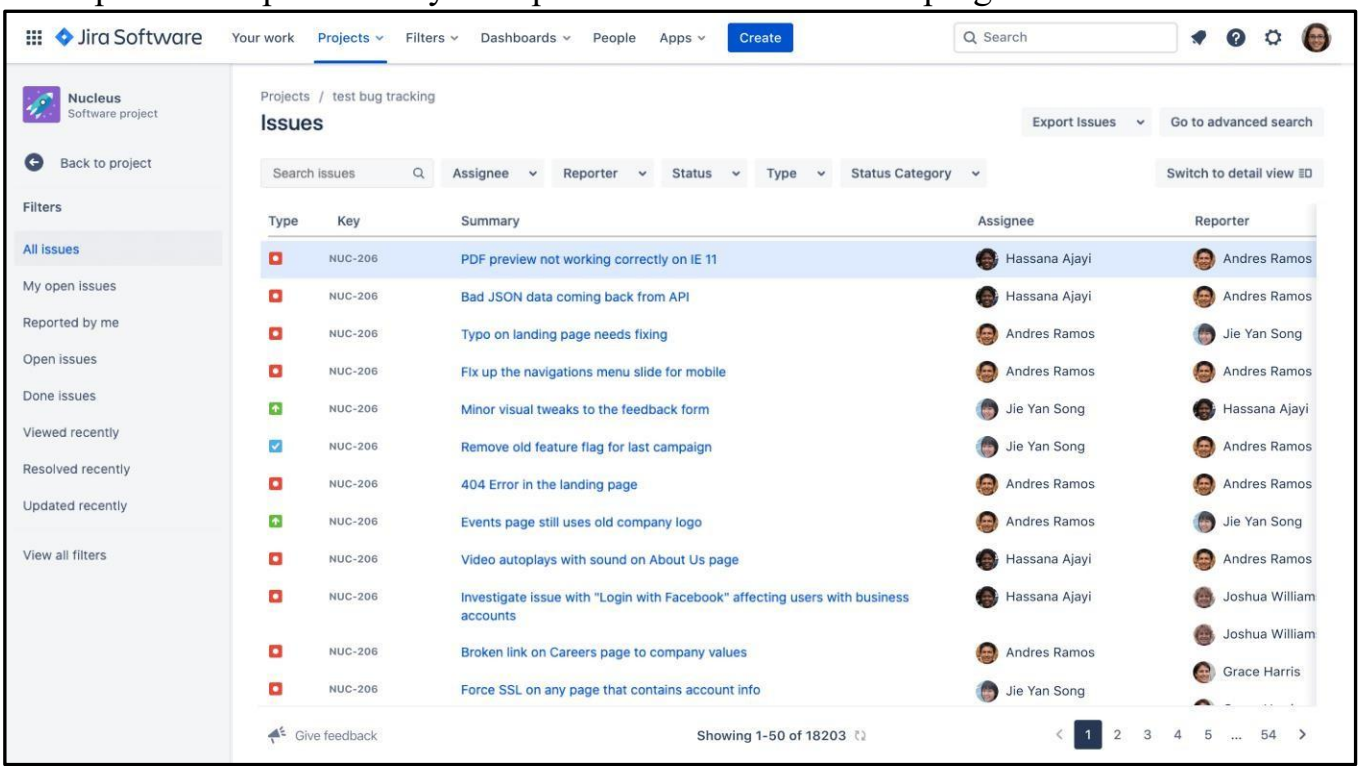


2. **Enter the details of the defect:** Users will need to enter details such as the summary, description, and affected version of the software. They can also add attachments such as screenshots or log files to provide more context.
3. **Assign the issue:** Users can assign the issue to a specific team member or team for investigation and resolution.
4. **Set the priority**: Users can set the priority of the issue to indicate its level of urgency.
5. **Add relevant labels, components, or versions**: Users can add labels, components, or

versions to the issue in order to help with organization and tracking

6. **Add comments**: Users can add comments to the issue to provide additional context or to discuss the issue with other team members.
7. **Track the issue:** Once the issue has been reported, users can track its progress by viewing the issue's details, comments, and change history.
8. **Close the issue**: Once the defect has been fixed, users can close the issue and mark it as resolved.

Jira provides a range of tools to help teams manage defects effectively and efficiently, including reporting, tracking, and resolution. It also provides a variety of filters, dashboards, and reports to help teams stay on top of their issues and track progress.



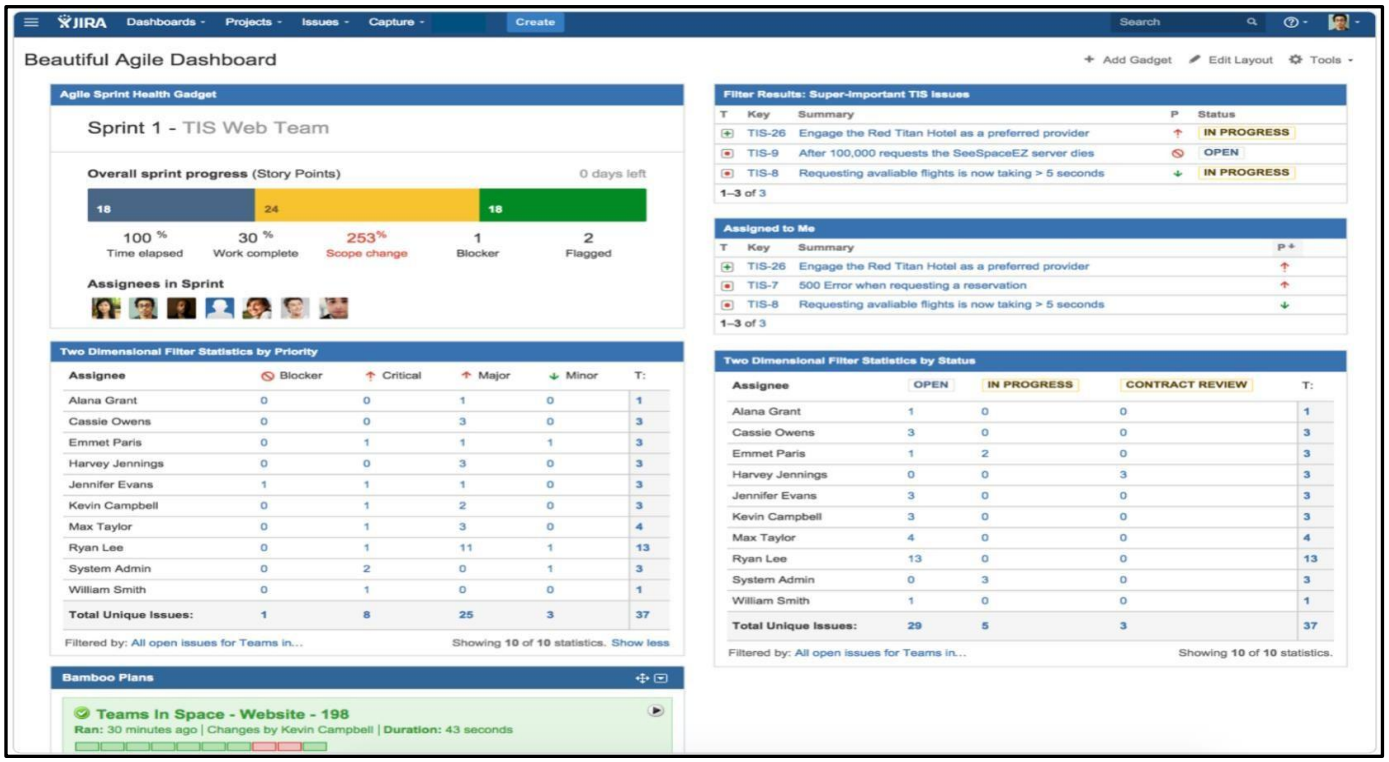
Jira Dashboard:

A Jira dashboard is a customizable and configurable page in Jira that provides an overview of the work being done by a team. It allows users to view and track the progress of their projects, issues, and tasks in a single place. Some of the key features of a Jira dashboard include:

1. Gadgets: Jira dashboards include a variety of gadgets, such as charts, reports, and filters, that can be added, removed, and configured to display relevant information.
2. Customizable layout: Users can customize the layout of their dashboard by adding, removing, and rearranging gadgets to create a layout that works best for their team.
3. Shared or personal: Jira dashboards can be shared among team members or kept as a personal dashboard for an individual user.
4. Filters: Users can create and apply filters to their gadgets to display specific information or issues based on certain criteria, such as issue type, status, or assignee.
5. Dashboard permissions: Users can set permissions for their dashboard to control who can view, edit, and administer it.

6. Reports: Users can add reports to their dashboard, such as burndown charts, to track the progress of their projects and issues.
7. Time tracking: Users can view the time tracking information of the issues on the dashboard.
8. Search: Users can search for issues on the dashboard and add search results to the dashboard.
9. Customizable color schemes: Users can customize the color scheme of their dashboard to match their team's branding or personal preference.

A Jira dashboard is a useful tool for teams to stay on top of their work and track progress. It provides a central location for team members to view and manage their tasks and projects, and helps teams to stay organized and informed.



Creating Product backlog in JIRA:

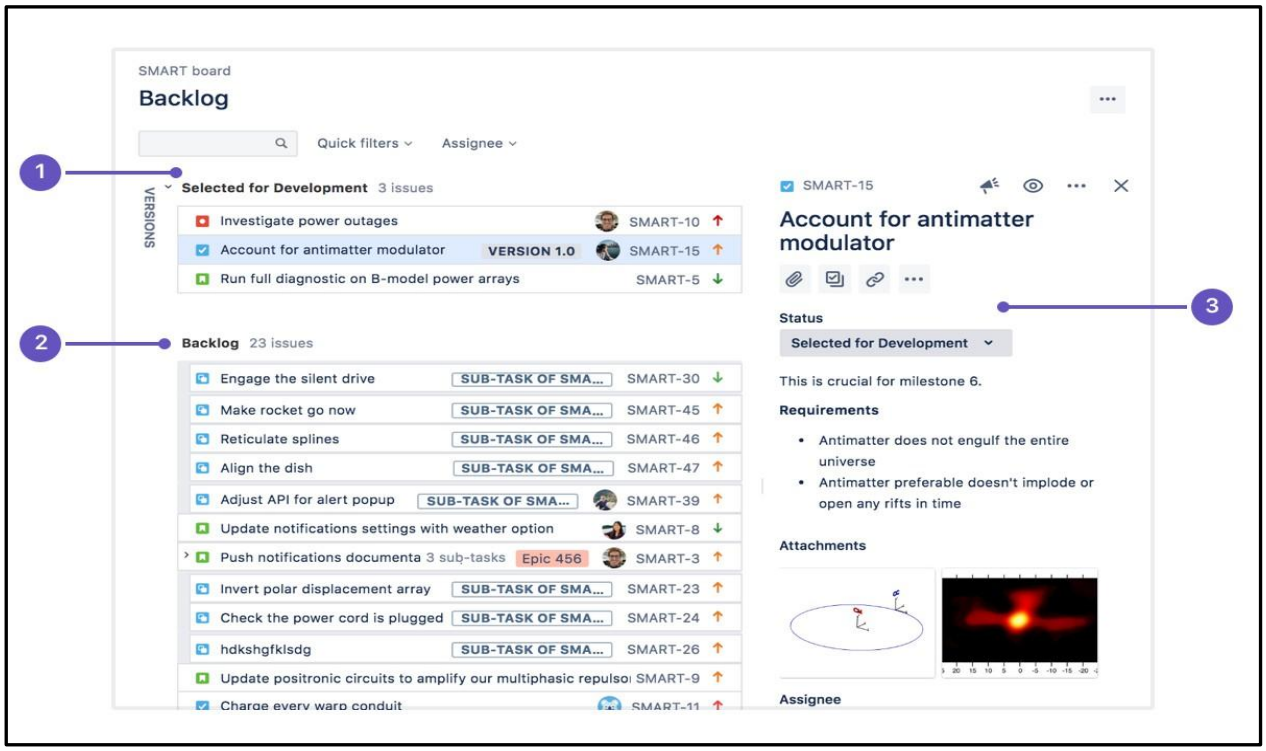
Creating a product backlog in Jira involves the following steps:

1. Create a new project: To create a product backlog, you will first need to create a new project in Jira. This can be done by going to the Jira home page and clicking on the "Create" button.
2. Define the issue types: Once the project is created, you will need to define the issue types that will be used in the product backlog. In Jira, the most common issue types used for product backlog items are "User Story" and "Epic".
3. Create the product backlog items: Once the issue types are defined, you can start creating the product backlog items. Each item should represent a piece of work that needs to be done. You can add more details to each item like description, acceptance criteria, and more.
4. Prioritize the backlog items: The product backlog should be prioritized, with the most

important items at the top and the least important items at the bottom. You can use Jira's built-in prioritization tools to reorder items in the backlog.

5. Assign the items to sprints: As items are completed, they can be assigned to sprints. This will help you track the progress of the work and ensure that the most important items are completed first.
6. Use filters and tags: You can use filters and tags to organize the product backlog and make it easier to find specific items.
7. Use the backlog board: You can use the Jira backlog board to see the entire product backlog and move items from the backlog to active sprints.
8. Use reports and metrics: Jira provides a wide range of reports and metrics to give teams visibility into product backlog progress and performance. You can use them to track the progress of the backlog items.

Creating a product backlog in Jira helps teams to organize and prioritize their work, ensuring that the most important items are completed first. It also provides a central location for teams to view and manage their tasks and projects, helping them stay organized and informed.



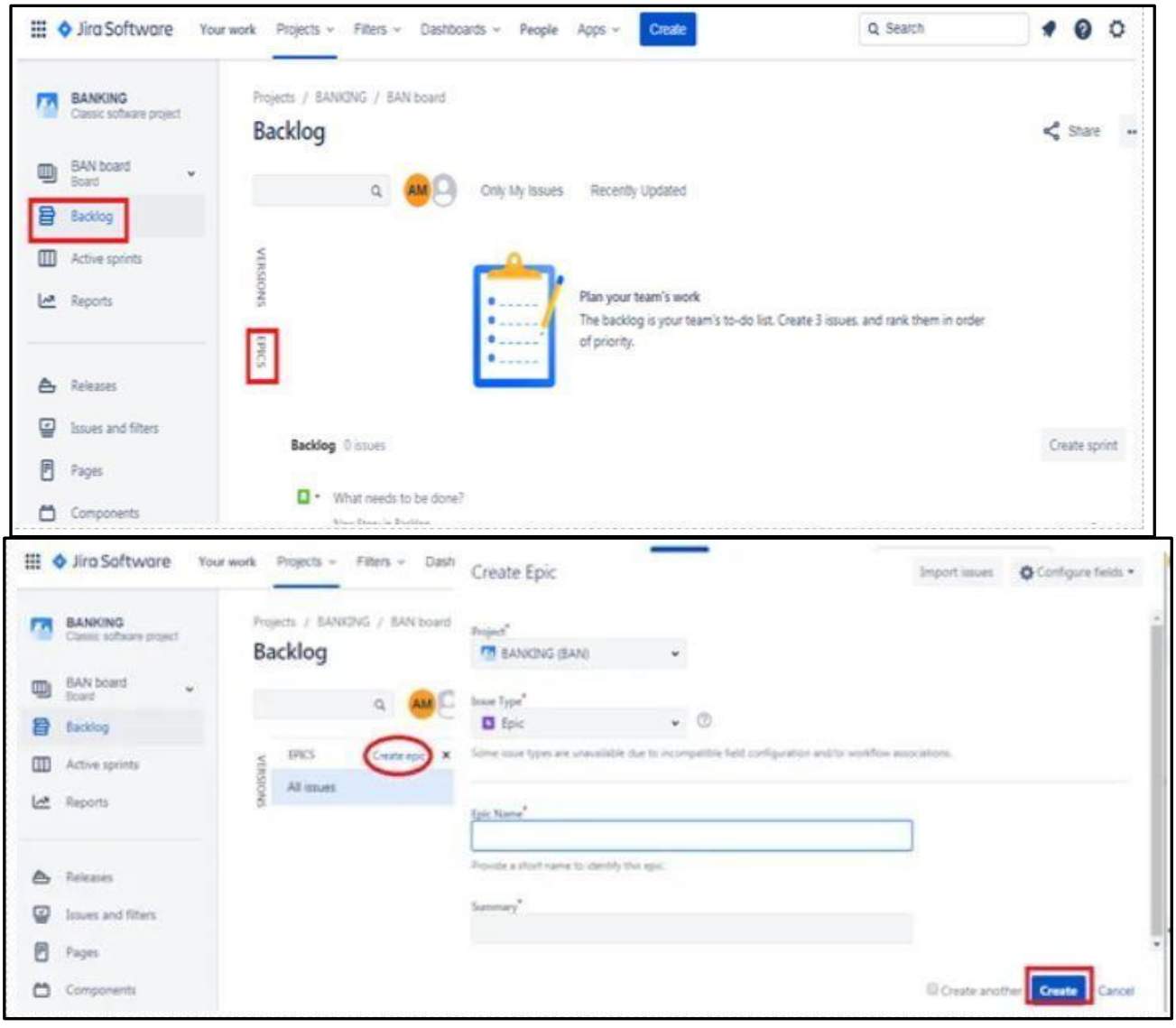
Creating EPICS in JIRA:

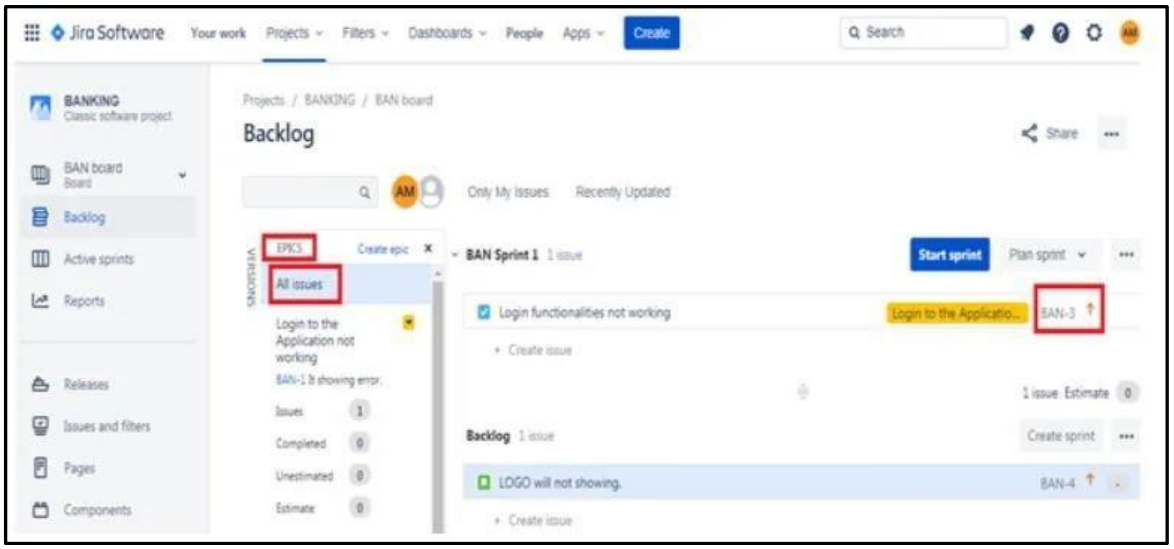
Creating epics in Jira involves the following steps:

1. Create a new issue: To create a new epic, you will need to create a new issue in Jira. You can do this by going to the Jira dashboard and clicking on the "Create" button.
2. Select the issue type: When creating a new issue, you will need to select the issue type. In Jira, the issue type for epics is typically "Epic."
3. Enter the details of the epic: Once the issue type is selected, you will need to enter the details of the epic, such as the summary, description, and affected version of the software.

4. Assign the epic: You can assign the epic to a specific team member or team for investigation and resolution.
5. Set the priority: You can set the priority of the epic to indicate its level of urgency.
6. Add relevant labels, components, or versions: You can add labels, components, or versions to the epic in order to help with organization and tracking.
7. Add comments: You can add comments to the epic to provide additional context or to discuss the epic with other team members.
8. Link the epic with related issues: Epics can be linked to related user stories, bugs or tasks, by using the link issue option in Jira.
9. Track the epic: Once the epic is created, you can track its progress by viewing the epic's details, comments, and change history.
10. Close the epic: Once the epic is completed, you can close the epic and mark it as resolved.

Creating epics in Jira helps teams to organize and prioritize their work by grouping related user stories and tasks together. It also provides a way to track the progress of large and complex projects, and helps teams stay organized and informed.



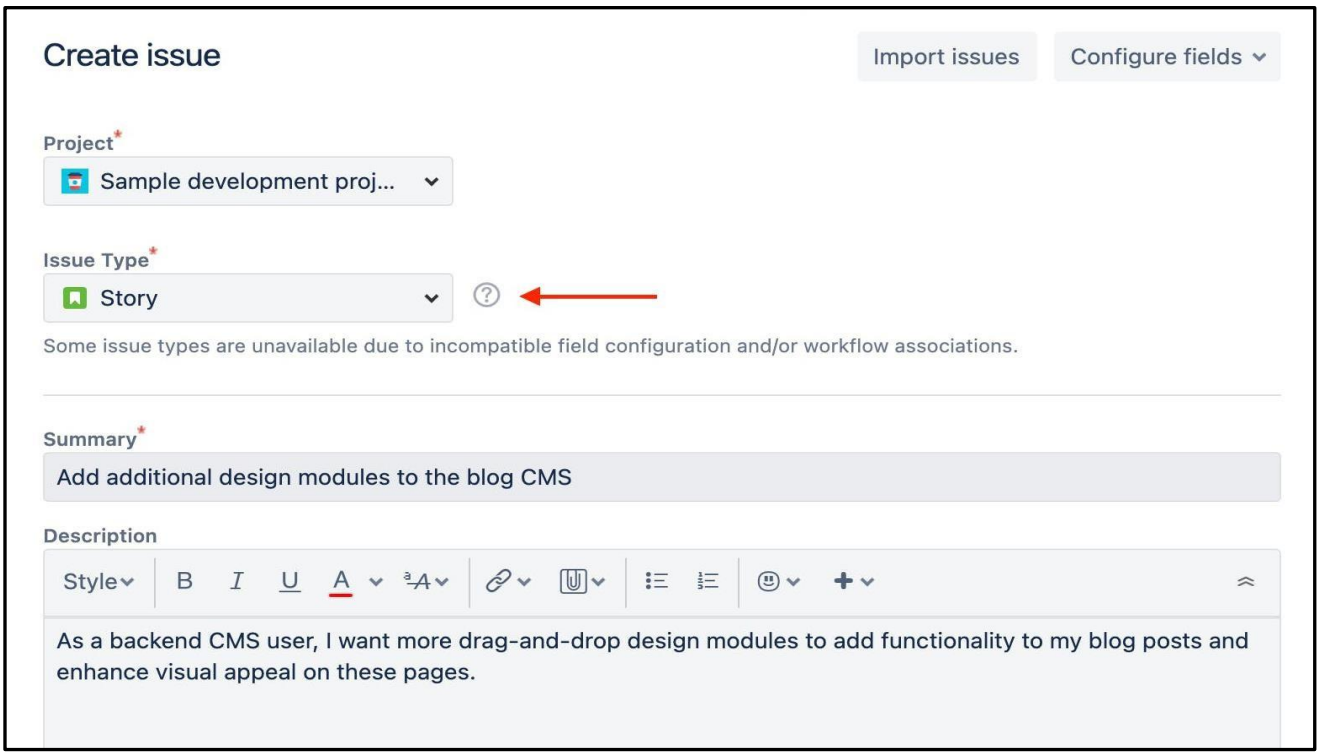


Creating User Stories/Issue in JIRA:

Creating user stories in Jira involves the following steps:

1. Create a new issue: To create a new user story, you will need to create a new issue in Jira. You can do this by going to the Jira dashboard and clicking on the "Create" button.
2. Select the issue type: When creating a new issue, you will need to select the issue type. In Jira, the issue type for user stories is typically "User Story."
3. Enter the details of the user story: Once the issue type is selected, you will need to enter the details of the user story, such as the summary, description, and acceptance criteria.
4. Add a related epic: If the user story is related to an epic, you can link it to the related epic by using the link issue option in Jira.
5. Assign the user story: You can assign the user story to a specific team member or team for investigation and resolution.
6. Set the priority: You can set the priority of the user story to indicate its level of urgency.
7. Add relevant labels, components, or versions: You can add labels, components, or versions to the user story in order to help with organization and tracking.
8. Add comments: You can add comments to the user story to provide additional context or to discuss the user story with other team members.
9. Track the user story: Once the user story is created, you can track its progress by viewing the user story's details, comments, and change history.
10. Close the user story: Once the user story is completed, you can close the user story and mark it as resolved.

Creating user stories in Jira helps teams to organize and prioritize their work by breaking down requirements into small and manageable tasks. It also provides a way to track the progress of specific tasks and helps teams stay organized and informed.



Writing TestCases in JIRA:

Create a new issue: To create a new test case, you will need to create a new issue in Jira.

You can do this by going to the Jira dashboard and clicking on the "Create" button.

Select the issue type: When creating a new issue, you will need to select the issue type. In Jira, the issue type for test cases is typically "Test Case."

Enter the details of the test case: Once the issue type is selected, you will need to enter the details of the test case, such as the summary, description, test steps and expected result.

Link the test case to a related issue: If the test case is related to a user story or a bug, you can link it to the related issue by using the link issue option in Jira.

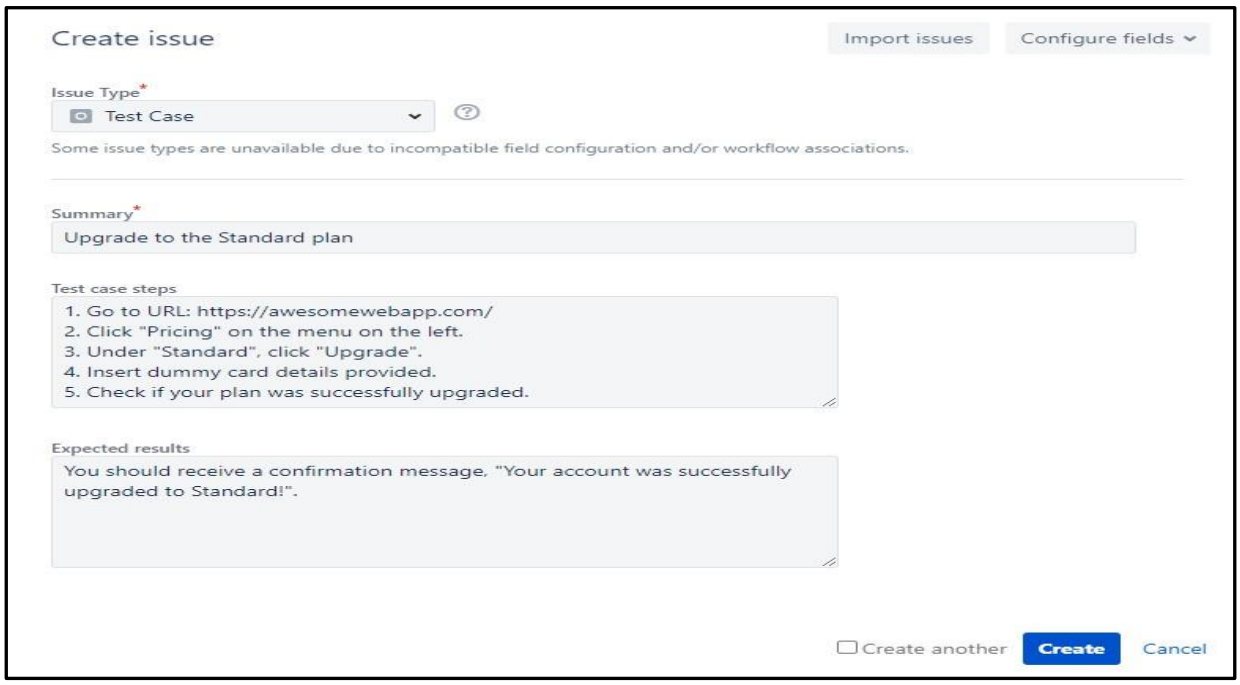
Assign the test case: You can assign the test case to a specific team member or team for execution and validation.

Set the priority: You can set the priority of the test case to indicate its level of importance.

Add relevant labels, components, or versions: You can add labels, components, or versions to the test case in order to help with organization and tracking.

Add comments: You can add comments to the test case to provide additional context or to discuss the test case with other team members.

Track the test case: Once the test case is created, you can track its execution status and results by viewing the test case's details, comments, and change.



Executing TestCases from JIRA:

Executing test cases from Jira involves the following steps:

Find the test case: Go to the Jira dashboard and use the search or filter functionality to find the test case you want to execute.

Update the status: Before starting the execution, update the status of the test case to indicate that it is in progress.

Execute the test: Follow the steps and expected results described in the test case and document any observations or results.

Update the status: After the execution, update the status of the test case to indicate whether it passed or failed.

Add comments and attach files: If necessary, add comments to the test case and attach any relevant files, such as screenshots or log files.

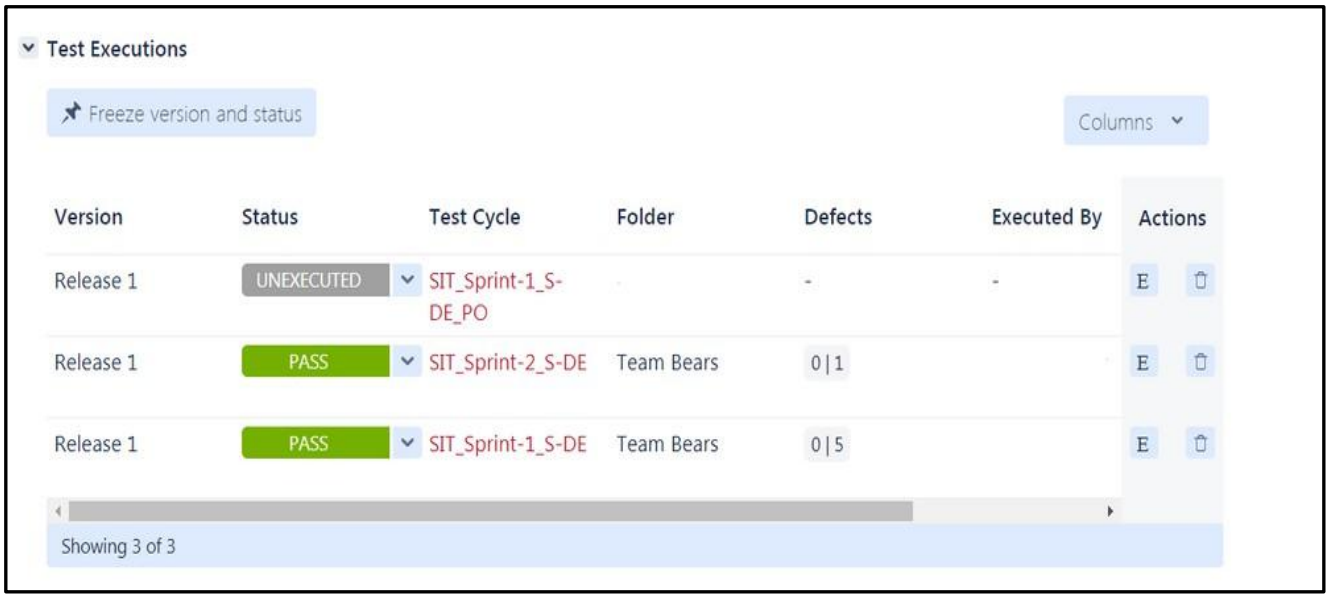
Link defects: If any defects were found during the execution, you can link them to the test case in Jira.

Use test management add-on: Jira also provides some test management add-ons like Xray, Zephyr and others, that can be used to execute test cases and record the results.

Reporting: Use the built-in reports and metrics of Jira or the test management add-on to track the progress and results of test execution.

Close the test case: Once the test case is executed and the results are recorded, you can close the test case and mark it as completed.

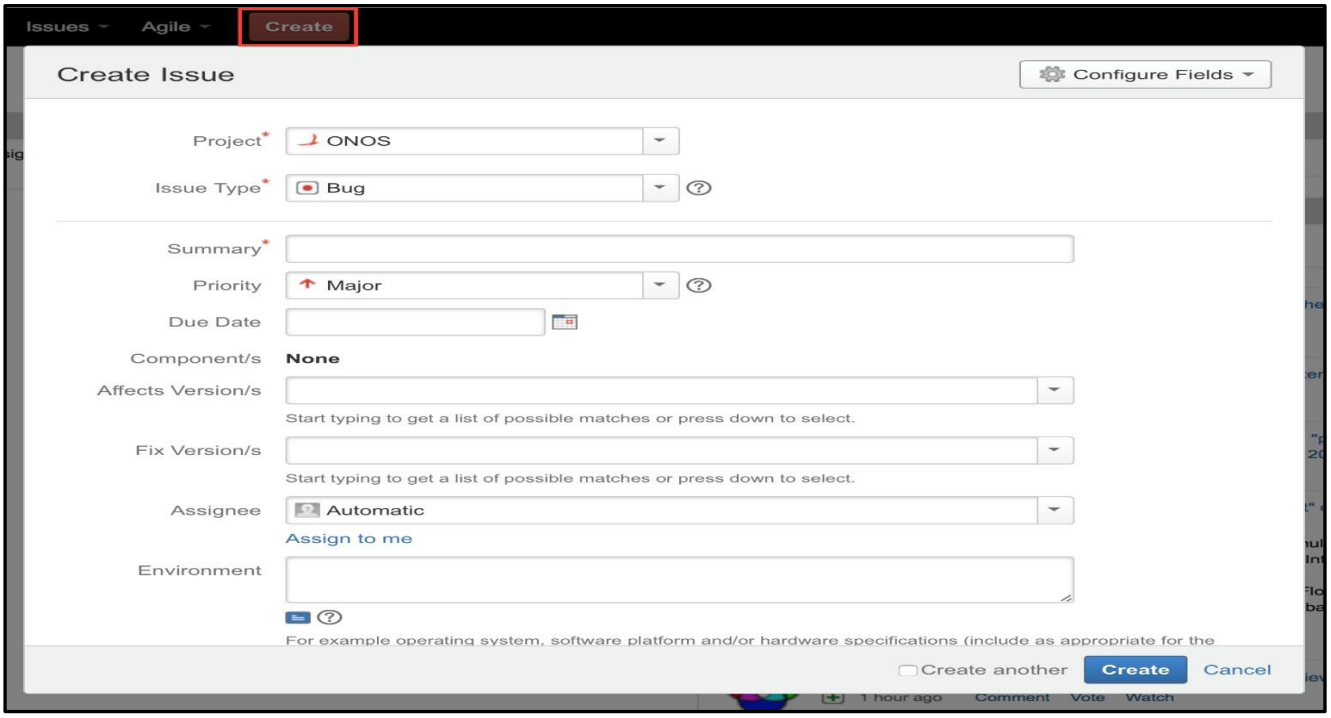
By executing test cases from Jira, teams can ensure that their software is thoroughly tested and that any defects are identified and resolved in a timely manner. The test execution results are also recorded and can be accessed later for reference or further analysis.



Adding Bugs to the JIRA Project:

Adding bugs to a Jira project involves the following steps:

- Create a new issue: To create a new bug, you will need to create a new issue in Jira. You can do this by going to the Jira dashboard and clicking on the "Create" button.
- Select the issue type: When creating a new issue, you will need to select the issue type. In Jira, the issue type for bugs is typically "Bug."
- Enter the details of the bug: Once the issue type is selected, you will need to enter the details of the bug, such as the summary, description, and affected version of the software.
- Add attachments: If necessary, add attachments such as screenshots or log files to provide more context for the bug.
- Assign the bug: Assign the bug to a specific team member or team for investigation and resolution.
- Set the priority: Set the priority of the bug to indicate its level of urgency.
- Add relevant labels, components, or versions: Add labels, components, or versions to the bug in order to help with organization and tracking.
- Add comments: Add comments to the bug to provide additional context or to discuss the bug with other team members.
- Link the bug with related issues: Bugs can be linked to related user stories, tasks or other bugs, by using the link issue option in Jira.



I would like to thank you for taking the time to read Software Testing Revealed.

I hope that you enjoyed reading this book as much as I had enjoyed while I were writing it. It is my biggest pleasure if we by any means manage to help you build a strong Software Testing foundation for yourself.

Thank you:

N. Krishna

Manual Interview Questions and Answers

1. What is manual testing?

Ans: Manual Testing is a type of software testing in which test cases are executed manually by a tester without using any automated tools. The purpose of Manual Testing is to identifythe bugs, issues, and defects in the software application. Manual software testing is the most primitive technique of all testing types and it helps to find critical bugs in the software application. Any new application must be manually tested before its testing can be automated.

2. Black box testing and white box testing?

Ans: It is an Interface testing by Test Engineer. Internal system testing is not considered in this type of testing. Tests are based on customer requirements and Functionality. Other than Unit Level testing, the remaining testing types belongs to Black box testing.

The White Box Test method is the one that looks at the code and structure of the product to be tested and uses that knowledge to perform the tests. This method is used in the Unit Testing phase, although it can also occur in other stages such as Integration Tests. For the execution of this method, the tester or the person who will use this method must have extensive knowledge of the technology used to develop the program.

3. What is data base testing?

Ans: Database Testing is a type of software testing that checks the schema, tables, triggers etc. of the database under test. It involves creating complex queries for performing the load or stress test on the database and check its responsiveness. It checks integrity and consistency of data. Database testing is important for ensuring that the data in the database is accurate, consistent, and secure. The database is a critical component of any software application, and errors in the database can result in serious consequences for the organization, including data loss, data corruption, and security breaches.

Some of the commonly used techniques for database testing include data integrity testing, performance testing, security testing, and data migration testing. Testing tools and frameworks

such as SQL Server Management Studio, MySQL Workbench, and Apache JMeter are often used for database testing.

4. Software testing types?

Ans: Accessibility testing **Acceptance testing** **Black box testing**

End to end testing Functional testing Interactive testing
Integration testing Load testing Non-functional testing
Performance testing Regression testing Sanity testing
Security testing Single user Performance testing
Smoke testing Stress testing Unit testing
White-box testing

5. What is Test cases?

Ans: A test case is a document which contains set of input values, execution preconditions, expected results and developed for a particular objective, such as to verify compliance(fulfilment) with a specific requirement. Test cases define how to test a system, software or an application. A test case is a singular set of actions or instructions for a tester to perform that validates a specific aspect of a product or application functionality.

If the test fails, the result might be a software defect that the organization can triage.

6. Defect life cycle?

Ans: Defect cycle or defect life cycle is ride of a defect from discovering defect to closure of defect. Defects management in defect cycle is important to ensure the software quality. Preventing, identifying, rectifying defect is important to improve the quality.

Defect Life Cycle States:

New - Potential defect that is raised and yet to be validated.

Assigned - Assigned against a development team to address it but not yet resolved.

Active - The Defect is being addressed by the developer and investigation is under progress. At this stage there are two possible outcomes; viz - Deferred or Rejected.

Test - The Defect is fixed and ready for testing.

Verified - The Defect that is retested and the test has been verified by QA.

Closed - The final state of the defect that can be closed after the QA retesting or can be closed if the defect is duplicate or considered as NOT a defect.

Reopened - When the defect is NOT fixed, QA reopens/reactivates the defect.
Deferred - When a defect cannot be addressed in that particular cycle it is deferred to future release.

Rejected - A defect can be rejected for any of the 3 reasons; viz - duplicate defect, NOT a Defect, Non-Reproducible.

7. Tell me about your project?

Ans: My Project name is "ERPPMC". It has different modules and each module having different pages.

Modules are:

1. Customer Module,
2. Purchase Module,
3. Sales Module,
4. Vendor Module,
5. Inventory Module and Finance Module.

8. What is difference between manual and automation?

Ans: There are some major differences between manual testing and automation testing. In manual testing, a human performs the tests step by step, without test scripts. In automated testing, tests are executed automatically via test automation frameworks, along with other tools and software.

9. Why you choosing manual testing?

Ans: Any software or product is incomplete without its testing. Without testing, there could be some bugs in the product and customers will not get satisfied according to the requirements. For testing, all companies need to have software tester for complete and satisfied software or product without any errors or bugs.

10. Smoke (or) Quick Testing and Sanity Testing (or) Build Test?

Ans: In smoke testing verify the build readiness for testing.

Here we are verifying whether build is ready for testing or not by testing all major functionalities (with valid data and without test cases).

→Sanity testing is a software testing technique in which a particular functionality of the application is verified instead of carrying out regression on the complete application. In this way, we can say that sanity testing is a subset of regression testing. Just like regression testing, in the case of sanity testing, we check if a fix has not affected the previously working functionalities of the application. But in lesser time and with a lesser number of test cases, focusing on some critical parts only.

11. What is block box testing?

Ans: It is an Interface testing by Test Engineer. Internal system testing is not considered in this type of testing. Tests are based on customer requirements and Functionality. Other than Unit Level testing, the remaining testing types belongs to Black box testing.

Black box testing Two types:

1.Functional Testing

2.Non-functional Testing

12. What is Regression Testing?

Ans: It is Re-execution of some are all test cases of a testing activities for each build to verify that fixes or changes made have not introduced new defects.

Regression testing is done in three situations.

1. After fixing the bug.

2. New change request will come from client.

3. When environment changes.

Here we have to verify whether already existing functionality can get defects or not.

13. Why we choose testing?

Ans: One company is developing any application the company wants to know whether the

developed application is working as per our requirements or not. So immediately to test the application. Testing is the process of verifying the defects of the developed application.

14. SDLC Process?

Ans: SDLC Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality software's. SDLC consists of a detailed plan which explains how to plan, build, and maintain specific software. Every phase of the SDLC life Cycle has its own process and deliverables that feed into the next phase.

15. A tester has found a defect where the defect is important to such that it has to be fixed the current release select the priority of the from the below options.

Ans. P1 Priority

16. Positive testing and negative testing?

Ans: **Positive Testing** is a type of testing which is performed on a software application by providing the valid data sets as an input. It checks whether the software application behaves as expected with positive inputs or not. Positive testing is performed in order to check whether the software application does exactly what it is expected to do.

Negative Testing is a testing method performed on the software application by providing invalid or improper data sets as input. It checks whether the software application behaves as expected with the negative or unwanted user inputs. The purpose of negative testing is to ensure that the software application does not crash and remains stable with invalid data inputs.

17. What is latent defect?

Ans: Defects that are not identified during Testing phase and are later identified after the software has been released into the market by real users using a specific scenario and specific data set. Latent defect is a term used in software engineering and quality assurance to refer to a defect or problem in the software application that is not immediately apparent or visible. It is a defect that remains hidden or dormant until a particular set of conditions are met or until the application is used in a specific way.

18. What is test case design techniques?

Ans: Test case design techniques are broadly divided into two types.

1. Black box testing (Functional).
2. White box testing (Structural).
   Black box Test case design techniques
   1. Equivalence Class Partitioning (ECP).
   2. Boundary Value Analysis (BVA).
   3. State transition testing.
   4. Cause-effect testing.

19. What is Data Testing?

Ans: In testing process, we need test data for validating the application. In this process, checking the prepared test data is suitable for testing the application or not is called Data Testing. Data testing is a type of software testing that is performed to verify that the data in the software application is accurate, complete, and consistent. It is an important aspect of software testing as data is a critical component of any software application, and errors or discrepancies in the data can have significant consequences for the organization.

Some of the commonly used techniques for data testing include data integrity testing, performance testing, security testing, and data migration testing. Data testing can also involve the use of test data management tools to generate test data and simulate different types of user input.

20. What is a Bug?

Ans: Problem which is identified on Test Engineer machine i.e called Bug (or) Defect. Expected result is not matching with the application actual result is called Bug. Any flaw (or) imperfection in software work product (Deliverable).

Bugs can be categorized into different types based on their impact and severity, such as critical bugs, major bugs, minor bugs, and cosmetic bugs. Critical bugs are the most severe type of bugs and can result in data loss, system crashes, or other serious issues. Major bugs can cause significant functionality issues or system errors, while minor bugs can cause minor issues that do not significantly impact the user experience. Cosmetic bugs, also known as visual bugs, are minor bugs that only affect the appearance of the application and do not impact its functionality. Effective bug management is a critical component of software development and involves the use of bug tracking tools to identify, report, and track bugs throughout the

development process. Proper bug management helps to ensure that bugs are quickly identified and resolved, resulting in a more stable and reliable software application.

21. What is compatibility testing?

Ans: Compatibility Testing is a type of Software testing to check whether your software is capable of running on different hardware, operating systems, applications, browsers, versions, network environments or Mobile devices.

Hardware: It checks software to be compatible with different hardware configurations.

Operating Systems: It checks your software to be compatible with different Operating Systems like Windows, Unix, Mac OS etc.

Software: It checks your developed software to be compatible with other software. For example, MS Word application should be compatible with other software like MS Outlook, MS Excel, VBA etc.

Network: Evaluation of performance of a system in a network with varying parameters such as Bandwidth, Operating speed, Capacity. It also checks application in different networks with all parameters mentioned earlier.

Browser: It checks the compatibility of your website with different browsers like Firefox, Google Chrome, Internet Explorer etc.

Devices: It checks compatibility of your software with different devices like USB port Devices, Printers and Scanners, Other media devices and Bluetooth.

Mobile: Checking your software is compatible with mobile platforms like Android, iOS etc.

Versions of the software: It is verifying your software application to be compatible with different versions of the software. For instance, checking your Microsoft Word to be compatible with Windows 7, Windows 7 SP1, Windows 7 SP2, Windows 7 SP3.

22. What is STLC?

Ans: STLC stands for Software Testing Life Cycle. STLC is a process used to test software and ensure that quality standards are met. The STLC is a process that follows a sequential set

of steps that start with planning and end with closure. The main objective of STLC is to ensure that the software product is of high quality, meets the customer requirements, and is delivered on time and within budget. Tests are carried out systematically over several phases.

STLC Phases:

1. Requirement Analysis.
2. Test Planning.
3. Test Design.
4. Test Environment Setup.
5. Test Execution.
6. Defect Logging.
7. Defect Retesting
8. Test Closure.

23. Severity, priority with examples?

Ans: **Severity** defines the importance of defect with respective to functional point of view…. means criticalness of defect with respective to application.

Responsibility of "Testing Team".

Classification could be:

*CRITICAL *MAJOR *MODERATE *MINIMAL

Priority defines the importance of defect with respective to client point of view… means how soon it should fix.

Responsibility of developing team (DL, Tech Lead)

Classification could be:

*P1(Urgent) *P2(High) *P3(Medium) *P4(Low)

24. What is SDLC Principles?

Ans: Implementing a SDLC is all about quality, reducing costs and saving time. Embracing the SDLC principles will improve your quality assurance practices, increase your project

success rate, reduce rework and provide deliverables that meet or exceed your stakeholders' expectations.

Principle #1: An effective organizational change management strategy is essential to the success of the SDLC.

Principle #2: Investment opportunities cannot be estimated accurately until the requirements are known.

Principle #3: Build systems with minimal defects.

Principle #4: Always conduct acceptance testing.

Principle #5: Complete all project artifacts before implementation.

25. Test design techniques?

Ans: Test design techniques are methods used to create a set of effective and efficient test cases that will help in achieving maximum test coverage. These techniques are designed to help testers create comprehensive test cases, optimize test coverage, and minimize the number of test cases required to achieve thorough testing. Some of the commonly used test design techniques include:

1. **Equivalence Partitioning:** This technique involves dividing the input data into different classes or partitions based on similar characteristics, and then selecting test cases from each partition.
2. **Boundary Value Analysis:** This technique involves selecting test cases that focus on the boundary values of the input data, as they are more likely to reveal defects than other values.
3. **Decision Table Testing:** This technique involves creating a table that lists all possible combinations of inputs and expected outputs, and then selecting test cases from this table.
4. **State Transition Testing:** This technique is used for testing systems that have different states or modes, and involves selecting test cases that exercise the transition between different states.
5. **Exploratory Testing:** This technique involves testing the application in an unscripted

manner, where the tester explores the application and identifies defects that may not be discovered by other techniques.

6. **Error Guessing:** This technique involves using the tester's experience and intuition to identify areas of the application that may be more error-prone, and then selecting test cases to exercise these areas.

26. Which is better among manual and automation Testing?

Ans: Manual testing and automation testing both have their own advantages and disadvantages, and the choice between the two depends on various factors, such as the project requirements, timelines, budget, and the expertise of the testing team. In some cases, a combination of both manual and automation testing may be necessary to achieve maximum test coverage. A skilled testing team should be able to evaluate the requirements of the project and determine the best approach to achieve the desired results.

Manual testing involves the execution of test cases manually by a tester without the use of any automated tools. Manual testing is generally recommended for small projects or when the test cases require frequent updates, as manual testing allows testers to quickly adapt to changes in the requirements. Manual testing is also beneficial when it comes to exploratory testing, where the tester can use their expertise and intuition to identify defects in the software.

Automation testing involves the use of automated testing tools to execute test cases. Automation testing is recommended for large projects or when the test cases require repetition, as it saves time and effort by automating the testing process. Automated testing is also beneficial for testing large and complex systems where manual testing can be time-consuming and error-prone.

27. How to identify the bugs?

Ans: As a software tester, identifying bugs involves several steps, including:

Test case execution: Run a set of test cases and compare the expected results with the actual results to identify any discrepancies.

Review of requirements: Compare the requirements to the actual product to identify any discrepancies or missing functionality.

Reproduce the issue: Try to recreate the issue multiple times to make sure it's a bug and not a one-time issue.

Gather information: Collect information about the environment, steps to reproduce the issue, and any error messages or logs.

Defect logging: Document any issues found during testing by creating a defect report, including steps to reproduce and expected vs actual results.

Exploratory testing: Ad-hoc testing that aims to uncover issues by randomly trying out different scenarios and combinations of inputs.

Logs and error messages: Review logs and error messages to identify any issues or exceptions thrown by the system.

User feedback: Collect feedback from end-users to identify any issues or pain points with the product.

Regression testing: Re-run previously passing tests to ensure new changes have not introduced new bugs.

Test automation: Use automated testing tools and scripts to identify bugs in a systematic and consistent manner.

28. Explain about critical issue faced in your defects?

Ans: A defect that has completely blocked the functionality of an application where the user or the tester cannot proceed or test anything. If the whole application's functionality is inaccessible or down because of a defect, such a defect is categorized as a critical defect. Critical issues are typically considered high-priority and require immediate attention and resolution. A critical defect is a show stopper which means the functionality cannot be delivered unless it is fixed. A critical defect has to be fixed immediately and ASAP since it can block testing of an entire product.

⇒ User cannot log in to the account.

⇒ A feature developed is missing in the build 'or' complete failure of a feature.

⇒ Internal Server error on accessing the application.

⇒ Error while customer is making the payment in case of eCommerce site, as the business loses its money.

⇒ Incorrect calculation or reporting of critical business data

⇒ Failures in key functionality, such as a shopping cart not calculating total cost correctly.

29. Which of the following is not a non-functional testing?

Ans: Here is a list of common types of non-functional testing:

Performance testing: Testing the system's performance, response time, and scalability under different loads and conditions.

Load testing: Testing the system's ability to handle expected levels of traffic and concurrent users.

Stress testing: Testing the system's behavior and stability under extreme loads and conditions.

Endurance testing: Testing the system's behavior and performance over an extended period of time.

Scalability testing: Testing the system's ability to accommodate increased traffic and load.

Compatibility testing: Testing the system's compatibility with different platforms, devices, browsers, and configurations.

Usability testing: Testing the ease of use and user experience of the system.

Security testing: Testing the system's security features and protections against potential threats and attacks.

Recovery testing: Testing the system's ability to recover from failures and errors, and return to normal operation.

Localization testing: Testing the system's behavior and display of content in different languages and cultural environments.

30. What is vulnerability testing?

Ans: Vulnerability testing is a type of security assessment that identifies and evaluates the security weaknesses or vulnerabilities in a system, network, or application. The purpose of

vulnerability testing is reducing the possibility for intruders/hackers to get unauthorized access of systems. Vulnerability testing is important because it helps organizations to proactively identify and address security risks, rather than waiting for an attack to occur. This can help to prevent data breaches, loss of confidential information, and other security incidents that can have serious consequences for the organization and its customers. It is performed to identify potential security risks and to evaluate the effectiveness of security controls in place.

31. What is Alpha Testing and Beta Testing?

Ans: Alpha Testing: Testing of the whole computer system before rolling out to the UAT. Alpha testing is carried out by the testers who are internal employees of the organization. Alpha testing performed by Testers who are usually internal employees of the organization.

Deliverable: Stable application

32.Differences between Smoke and Sanity, Retesting and Regression testing?

Smoke Testing	**Sanity Testing**	**Regression Testing**	**Re-testing**
performed before any detailed testing and is used as a quick check to determine if the application is stable enough to proceed with further testing.	used to ensure that the application is still functioning as expected after a minor change.	performed after changes have been made to the code and is used to ensure that the application is still functioning as expected after these changes.	bug has been completely fixed and that the application is functioning as expected.
Performed on initial builds	Performed on stable builds	Performed on stable builds	Performed on stable builds
Executed by testers & sometimes also by developers	Executed by testers	Executed by testers, mostly via automation	Executed by testers

Defect verification is not the part of smoke testing	Defect verification is not the part of sanity testing	Defect verification is not the part of regression testing	Defect verification is the part of re-testing
The purpose of smoke testing is to verify the "stability" of the system in order to proceed with more rigorous testing	The purpose of sanity testing is to verify the "rationality" of the system in order to proceed with more rigorous testing	The purpose of regression testing is that new changes should not have any side effects to existing functionalities	Re-testing is done on the basis of the defect fixes

33. Write test cases for email id?

Ans: Design or UI related Test Cases:

- Verify email field is present on the page.
- Verify label text is shown with the email field or not.
- Verify label text email align with the email field.
- Verify that the placeholder text in the email field is added or not.
  Functional Test Cases:
- Verify email address field is accessible by clicking on the email field.
- Check users can type email in the email field.
- Verify user can paste the email address in the field by keyboard keys Ctrl + v.
- Verify that the user can paste the email address with the mouse by right-clicking in the email field and pasting the email address.
- Verify validation for the email field is implemented or not.
- Verify an error message should be shown in case if the user adds an invalid email address or not.

  Here are some test cases for validating an email ID:

1. Test for a valid email ID format:
   Input: user@example.com
   Output: Email ID is valid

2. Test for an invalid email ID format (missing @ symbol):

   Input: userexample.com

   Output: Email ID is invalid

3. Test for an invalid email ID format (missing domain name):

   Input: user@.com

   Output: Email ID is invalid

4. Test for an invalid email ID format (missing user name):

   Input: @example.com

   Output: Email ID is invalid

5. Test for an invalid email ID format (missing. in domain name):

   Input: user@examplecom

   Output: Email ID is invalid

6. Test for an invalid email ID format (using spaces):

   Input: user @example.com

   Output: Email ID is invalid

7. Test for an invalid email ID format (using special characters other than . and @):

   Input: user#example.com

   Output: Email ID is invalid

8. Test for a maximum length email ID:

   Input:user12345678901234567890123456789012345678901234567890123@example.com

   Output: Email ID is valid

9. Test for an email ID with multiple dots in the domain name:

   Input: user@example.com.au

   Output: Email ID is valid

34. How you start testing process?

Ans: Starting the testing process involves the following steps:

1. Requirements gathering: Gather the requirements and specifications of the software to be tested. This will help to understand what needs to be tested and what the expected outcome should be.

2. Test planning: Create a test plan that outlines the scope, objectives, and approach for testing. The test plan should also include the resources, timeline, and budget for testing.
3. Test case development: Develop test cases based on the requirements and test plan. Test cases should include steps for testing, expected results, and any additional information needed to execute the tests.
4. Test environment setup: Set up the test environment, including hardware, software, and any necessary tools. This may involve installing the software, configuring the test environment, and preparing test data.
5. Test execution: Execute the test cases and document the results. This may involve manual testing, automated testing, or a combination of both.
6. Defect tracking: Track any defects that are found during testing and report them to the development team for resolution.
7. Test closure: Close the testing phase when all the tests have been executed and the results have been documented. This may involve preparing a final test report and documenting any lessons learned during the testing process.

35. What is test coverage?

Ans: Test coverage is defined as a statistic that indicates the quantity of testing completed by a collection of tests. Test coverage is defined as a metric in Software Testing that measures the amount of testing performed by a set of tests. It will include gathering information about which parts of a program are executed when running the test suite to determine which branches of conditional statements have been taken.

Test coverage (TC)=Test cases executed / total number of Test cases *100

36. what is Exhaustive testing?

Ans: Exhaustive testing is a testing method in which all possible combinations of inputs, conditions, and scenarios are tested. The goal of exhaustive testing is to ensure that all possible scenarios have been considered and that no defects or bugs exist in the software. Exhaustive testing is sometimes known as ‘complete testing’ and aims to reduce the chance of undiscovered faults in a software package. A team of software ‘testers’ will test as many possible input combinations and conditions, looking for bugs and glitches. Exhaustive testing is typically used in situations where high reliability and safety are required, such as in the

aerospace, defense, and medical industries. In most other cases, exhaustive testing is not practical or necessary, and other testing methods, such as risk-based testing or scenario-based testing, are used instead.

Exhaustive testing should be used in combination with other testing methods, such as unit testing, integration testing, and system testing, to provide a comprehensive testing approach.

37. What is Design?

Ans: Design refers to the process of planning and creating a solution to meet a particular need or requirement. It involves visualizing, conceptualizing, and defining the structure, form, and function of a product or system.

Design in software testing refers to the process of creating a plan or blueprint for testing a software application. It involves determining what tests need to be performed, what test cases will be used, and what testing tools and techniques will be employed.

1. The software requirements: The testing plan should ensure that all the requirements of the software are covered in the testing process.
2. The software architecture: The testing plan should take into account the software architecture and how it will affect the testing process.
3. The software environment: The testing plan should consider the software environment, including the hardware and software configurations, network connections, and data sources.
4. The testing goals: The testing plan should clearly define the goals of the testing process and how they will be achieved.
5. The testing schedule: The testing plan should include a schedule for when each test will be performed, who will perform it, and what resources will be required.

38.100 cases are there, then how these cases are executed?

Ans: The best approach to executing test cases will depend on the specific needs and constraints of the project. It may be necessary to use a combination of these approaches to ensure that all the test cases are executed effectively and efficiently.

1. Test prioritization: Prioritizing the test cases based on their criticality and risk, and executing the high-priority test cases first, can help to focus the testing efforts on the

most important issues.

2. Test parallelization: Running multiple tests at the same time on different testing environments can help to reduce the overall testing time.
3. Test optimization: Optimizing the test cases by reducing the number of test steps, reducing the time required to set up the testing environment, and reusing test data can help to reduce the time required to execute the tests.
4. Test environment management: Properly managing the testing environment, such as ensuring that it is stable, well-configured, and up-to-date, can help to reduce the time required to execute the tests.
5. Test data management: Using optimized and well-managed test data can help to reduce the time required to execute the tests, as well as increase the consistency and accuracy of the results.
6. Manual execution: In this approach, the test cases are executed manually by a tester, who follows the steps outlined in the test cases and records the results. This is often done for smaller projects or when a high degree of human judgment is required.
7. Automated execution: In this approach, test cases are executed using automated testing tools, such as test scripts or test frameworks. This can help to reduce the time and effort required to execute the tests, as well as increase the consistency and accuracy of the results.

By using these strategies, you can significantly reduce the time required to execute 100 test cases and ensure that the tests are executed efficiently and effectively.

39. How do you experienced your test?

Ans: The experience of software testing can vary depending on the specific software being tested, the testing techniques being used, and the goals of the testing process. In general, software testing can be a time-consuming and meticulous process, but it is an essential step in the software development cycle to ensure the quality and reliability of the final product.

In software testing, the goal is to validate and verify that a software application meets its specified requirements and works as intended. This is usually done through a combination of manual and automated testing techniques. The software is subjected to various tests, such as functional testing, performance testing, security testing, etc. The results of these tests are then

analyzed to identify any bugs or issues that need to be fixed.

40. Testing techniques?

1. Manual testing – Involves manual inspection and testing of the software by a human tester.
2. Automated testing – Involves using software tools to automate the testing process.
3. Unit Testing: It is a method of testing individual units or components of a software application.
4. Integration Testing: It involves testing the interactions between different components or systems to ensure they work together as expected.
5. Functional Testing: This type of testing verifies that the software functions according to the specified requirements.
6. End-to-end Testing: It tests the complete software system from start to finish to ensure that all the components work together seamlessly.
7. Regression Testing: It is performed to verify that changes made to the software have notintroduced new bugs or affected existing functionality.
8. Performance Testing: This type of testing is performed to evaluate the speed, stability, and scalability of the software under different conditions.
9. Security Testing: It is a type of testing that is performed to evaluate the security of the software and identify any vulnerabilities.
10. User Acceptance Testing (UAT): This type of testing is performed by the end-users of the software to verify that it meets their requirements and expectations.

41---- is to check the coverage of requirements in the test design?

Ans: Testing coverage of requirements is an important part of the software development process because it helps identify any gaps or missing requirements and ensures that the software is of high quality and meets the expectations of the end-users.

Yes, checking the coverage of requirements in the test design is an important goal of software testing. The aim is to ensure that all relevant requirements, both functional and non-functional, have been included in the test design and are being adequately tested.

To achieve this goal, testers often create a test plan that outlines the requirements that need to be tested and the methods that will be used to test them. The test plan serves as a roadmap for

the testing process and helps ensure that all of the requirements are thoroughly tested.

42. What is bug life cycle?

Ans: The Defect Life Cycle, also known as the Bug Life Cycle, is a cycle of defects from which it goes through covering the different states in its entire life. This starts as soon as any new defect is found by a tester and comes to an end when a tester closes that defect assuring that it won't get reproduced again.

1. Detection: A bug is identified and detected during testing.
2. Report: The bug is reported and documented in a bug tracking system.
3. Prioritization: The bug is prioritized based on its severity and impact on the system.
4. Investigation: The bug is investigated by the development team to determine its root cause.
5. Fix: A solution is developed to resolve the bug.
6. Verification: The solution is tested to verify that the bug has been fixed.
7. Closure: The bug is marked as resolved and closed in the bug tracking system.

43.Verification &Validation?

Ans: Verification in software testing refers to the process of evaluating the software design and implementation to determine if they meet the specified requirements. It is a process of checking that the software product is designed and developed correctly and meets the quality standards. The main goal of verification is to catch errors and mistakes early in the development cycle, before they become more costly and time-consuming to fix. This process includes activities such as code review, walk-through, inspection, and testing prototypes. Verification helps ensure that the software is built to meet the user's needs and expectations.

Validation in software testing is the process of evaluating a software product during or at the end of the development process to determine whether it satisfies the specified requirements. The main objective of validation is to assess the software's compliance with the user's needs and expectations, and to ensure that it is fit for its intended use. This process includes activities such as functional testing, integration testing, system testing, and acceptance testing. The goal of validation is to identify any defects or issues that may affect the software's functionality and performance, and to ensure that the software meets the necessary quality standards before it is released to the end-user. Validation helps to reduce the risk of releasing a software product

that does not meet the customer's requirements, leading to customer dissatisfaction and increased costs for fixing the defects.

44. Write test cases on login page?

Ans: Test cases that can be performed on any login page:

Positive test cases:

1. Test the login functionality with a valid username and password combination.
2. Test the case-insensitivity of the username field.
3. Test the maximum and minimum length of the username and password fields.
4. Test the successful login with special characters in the username and password.
5. Test the "remember me" feature by checking if the login credentials are saved after logging out.

Negative test cases:

1. Test the login functionality with an invalid username and password combination.
2. Test the login functionality with an empty username and password.
3. Test the login functionality with a username that does not exist.
4. Test the login functionality with an incorrect password for a valid username.
5. Test the login functionality with an account that has been disabled or locked.
6. Test the error messages displayed for invalid login attempts.

45. Write test cases for Gmail login page?

Ans: Here are some test cases that can be performed on the Gmail login page:

Positive test cases:

1. Test the login functionality with a valid Gmail account.
2. Test the case-insensitivity of the username field.
3. Test the successful login with special characters in the username and password.
4. Test the "Stay signed in" feature by checking if the login credentials are saved after logging out.
5. Test the "Forgot password" feature and verify that the reset password process works.
6. Test the "Two-step verification" feature and verify that the process works as expected.

Negative test cases:

1. Test the login functionality with an invalid Gmail account.
2. Test the login functionality with an empty username and password.
3. Test the login functionality with an incorrect password for a valid Gmail account.
4. Test the error messages displayed for invalid login attempts.
5. Test the login functionality with an account that has been disabled or locked.

These are just some of the basic test cases that can be performed on the Gmail login page.

46. What is performance testing, types

Ans: Checking the behavior of an application by applying some load is known as performance testing. Performance testing is a non-functional testing. While doing performance testing onthe application, we will concentrate on the various factors like Response time, Load, and Stability of the application.

Types of Performance Testing:

1. Load testing
2. Stress testing
3. Scalability testing
4. Stability testing

1.Load testing: To check the performance of an application by applying some load which is either less than or equal to the desired load is known as load testing.

2.Stress Testing: The stress testing, checks the behavior of an application by applying load greater than the desired load.

3.Scalability Testing: Checking the performance of an application by increasing or decreasing the load in particular scales (no of a user) is known as scalability testing.

4.Stability Testing: Checking the performance of an application by applying the load for a particular duration of time is known as Stability Testing.

47. What is defect components?

Ans: Defect Components

1. Defect ID.

2. Defect Description.
3. Version.
4. Action Steps.
5. Expected Result.
6. Environment.
7. Actual Result.
8. Severity.
9. Detected By
10. Fixed by
11. Date Raised
12. Priority
13. Date Closed

48. WhatsApp test scenarios?

Ans: **WhatsApp Test Scenarios:**

⇒ Verify that on downloading the WhatsApp application, users can register using a new mobile number.

⇒ Verify that for a new mobile number user will get a verification code on his mobile and filling the same verifies the new user account.

⇒ Check the maximum number of incorrect attempts allowed while filling the verification code.

⇒ Verify that registering an existing mobile number for new user account registration is not allowed.

⇒ Verify that on successful registration all the contacts in the user's contact directory get imported to the WhatsApp contact list.

⇒ Verify that the user can set DP and status on WhatsApp.

⇒ Verify that the user can update the existing DP and WhatsApp status.

⇒ Verify that the user can send messages to any individual selected from his contact list.

⇒ Verify that 'Chats' window contains all the chat list with DP and name and last message preview of the other person with whom chat was initiated.

⇒ Verify that clicking a chat in the chat list opens a new window containing all the chats

received and sent with the other person.

⇒ Verify that the user can check the message delivered and read time for a message in the

⇒ 'Message Info' section.

⇒ Verify that the user can share or receive contact with the other person.

⇒ Verify that the user can create a group adding multiple people from his contact list.

⇒ Verify that the user can send and receive the message in group chats.

⇒ Verify that users can send and receive images, audio, video, emoticons in chat to individuals.

⇒ Verify that users can send and receive images, audio, video, emoticons in group chats.

⇒ Verify that the user can send and receive chats in secondary languages available.

⇒ Verify that users can delete text, images, audio, video messages within a chat.

⇒ Verify that users can clear complete chat history in an individual or group chat.

⇒ Verify that users can archive chats in an individual or group chat.

⇒ Verify that users can block a user to prevent any message from getting received from the blocked contact.

⇒ Verify that the user makes WhatsApp calls to the person in his contact list.

⇒ Verify that the user can receive WhatsApp calls from a person in his contact list.

⇒ Verify that users can mark chats as favorite and access all chats marked as favorite from the 'Favorites' section.

49. WhatsApp test cases?

Ans: Some of important test cases for whatsapp.

1. First Verify that the after successfully downloading the application user able to register mobile number or not.

2. Verify that the user able to verify the number successfully or not.

3. Verify that when user enter the mobile number in whatsapp app which is already register.

4. Verify that after the successfully verification all the element properly visible / present or not. (Contact list, menu bar, chat status, calls etc.).

5. Verify that all the whatsapp contact list showing or not.

6. Check that user able to set the dp or not.

7. Check that user able to set the status or not on whatsapp app. 8. Check that user able to send the whtasapp message to any whatspp contact.

9. Check that user able to send the message by selecting attaching the photo, video, file etc through the whatsapp app.

10. Verify that by default Chat option selected or not.

11. Verify that all the chat list showing or not .(whith last message) on chat window.
12. Verify that on clicking on status window, Status of contact sowing or not.

13. Check that on clicking on calls window all the incoming and outgoing audio video call list showing or not.

14. Check that when user click on gallery option it redirects user to the phone gallery or not.

15. Check that user able to select the image from the gallery or not.

16. Check that after the selection of picture resize option showing or not.

17. Check that user able to resize the image or not.

18. Check that when user click on done option then selected image get updated as dp or not and confirmation message showing or not.

19. Check that when user click on cancel button then it redirects user to in gallery or not.

20. Check that when user select the camera option then mobile phone camera get on or not.

50. Difference between Test design and Plan?

Ans: Test design focuses on the creation of test cases, while the test plan focuses on the overall strategy and plan for testing the software application. The test design provides the details for executing the tests, while the test plan provides the overall framework and direction for the testing effort.

Test design is the process of creating and documenting the test cases that will be used to validate the functionality of a software application. The goal of test design is to ensure that the software is tested thoroughly and that all potential issues are identified and addressed. Test design involves identifying the requirements, defining the test objectives, developing the test

cases, and choosing the appropriate testing methods.

Test plan, on the other hand, is a document that outlines the approach, resources, and schedule for testing a software application. It provides a high-level overview of the testing process and includes information such as the scope of testing, the testing methodologies to be used, the resources required, the schedule for testing, and the criteria for success. The test plan is a livingdocument that is updated throughout the testing process as changes are made to the software application or the testing approach

51. Write Test case template?

Ans: Here is a sample test case template that you can use:

Test Case ID: [Unique identifier for the test case]

Test Case Title: [Brief description of the test case]

Test Designed by: [Tester's Name]

Date of test designed: [Date when test was designed]

Test Executed by: [Who executed the test- tester]

Date of the Test Execution: [Date when test needs to be executed]

Test Steps: [Mention all the test steps in detail and write in the order in which it requires to be executed. While writing test steps ensure that you provide as much detail as you can]

Test Data: [List of any necessary data required to execute the test case, if applicable]

Test Environment: [Description of the environment in which the test case will be executed, if applicable]

Expected Result: [Description of the expected result of the test case]

Actual Result: [Result of the test case, filled after execution]

Status (Fail/Pass): Mark this field as failed, if actual result is not as per the estimated result

A	B	C	D	E	F	G	H	I	J	K	L	M	N
TEST CASES TEMPLATE													
Componay Logo	Project Code							Total No of Tc'S					
	Project Name							Passed					
	Prepared By		Date Prepared					Failed					
	Reviewed By		Date Reviewed					Not Executable					
	Approved By		Date Approved					Defects Reported					
	TestCase Name		TestCase Objective					Tc'S Automated					
	Executed Name		Date Executed										
	Version		Build										
TC#	TS#	Test Design/Test Steps	Test Data	Expected Result	Actual Result	Pass	Fail	Not Executable	Severity				Defect ID
									Critical	Major	Modarate	Minar	

52. what is testing methodology?

Ans: Testing methodology is a systematic approach to software testing that outlines the tasks, techniques, and processes involved in testing. It serves as a blueprint for software testing and helps ensure that the testing process is systematic, efficient, and effective. The main objective of testing methodology is to find as many defects as possible in the software, within the given time and resources.

There are several different testing methodologies, each with its own strengths and weaknesses. Some of the most commonly used testing methodologies include:

Agile Testing: An iterative and incremental approach to testing that is commonly used in Agile software development.

Waterfall Testing: A sequential testing methodology that follows the Waterfall software development model.

V-Model Testing: A graphical representation of the testing process that maps each stage of the software development life cycle to its corresponding test activities.

Exploratory Testing: An approach to testing that focuses on discovering and exploring the software to uncover defects.

The choice of testing methodology depends on various factors, such as the size and complexity of the software, the goals of the testing process, and the available resources. The most important factor is that the methodology should be suitable for the software and should help the testing team achieve its objectives.

53. Difference between Smoke and Sanity?

Ans:

Smoke Testing	Sanity Testing
The main purpose of smoke testing is to quickly determine if the most critical functions of a software application are working correctly.	Sanity testing, on the other hand, is performed to verify if a small change or modification to the software has not affected its other functionalities.
Smoke testing is used to test all over function of the system/product.	Sanity testing is used in the case of only modified or defect functions of system/products.
This testing is performed by the developers or testers.	Sanity testing in software testing is usually performed by testers.
Smoke testing can be performed either manually or by using automation tools.	Sanity testing is commonly executed manually, not by using any automation approach.
Smoke Testing firstly performs on the initial build. smoke testing is done first.	Sanity Testing is done on stable builds or for the introduced new features in the software.

54. How to design Testcases and Scenarios?

Ans: How to create a Test Scenario:

⇒ Carefully study the Requirement Document – Business Requirement Specification (BRS), Software Requirement Specification (SRS), Functional Requirement Specification (FRS) pertaining to the System Under Test (SUT).

⇒ Isolate every requirement, and identify what possible user actions need to be tested for it. Figure out the technical issues associated with the requirement. Also, remember to analyze and frame possible system abuse scenarios by evaluating the software with a hacker's eyes.

⇒ Enumerate test scenarios that cover every possible feature of the software. Ensure that these scenarios cover every user flow and business flow involved in the operation of the website or app.

⇒ After listing the test scenarios, create a Traceability Matrix to ensure that every requirement is mapped to a test scenario.

⇒ Get the scenarios reviewed by a supervisor, and then push them to be reviewed by other stakeholders involved in the project.

How to create a Test Cases:

⇒ Simple and clear: Test cases need to be very concise, clear, and transparent. They should be easy and simple to understand not only for oneself but for others as well.

⇒ Maintain the uniqueness: While writing the test cases, it's necessary to make sure that they aren't being written over and over again and each case is different from the other.

⇒ Zero Assumptions: Test cases should not contain assumed data, don't come up with features/modules that don't exist.

⇒ Traceability: Test cases should be traceable for the future reference, so while writing it's important to keep that in mind,

⇒ Different input data: While writing test cases, all types of data must be taken into consideration.

⇒ Strong module name: The module name should be self-explanatory while writing the test case.

⇒ Minimal Description: The description of a test case should be small, one or two lines are normally considered good practice but it should give the basic overview properly.

⇒ Maximum conditions: All kinds of conditions should be taken into consideration while writing a test, increasing the effectiveness.

⇒ Meeting requirements: While writing the test case it's important that the client/customer/end-user requirements are met.

⇒ Repetitive Results: The test case must be written in such a way that it should provide the same result.

55. Test Case vs Test Scenario?

Ans: Test scenario is "What is to be tested". Identify all the possible areas to be tested.

Test case is “How, what is to be tested”.

Test Scenario	Test Case
A test scenario contains high-level documentation which describes an end-to-end functionality to be tested.	Test cases contain definite test steps, data, expected results for testing all the features of an application.
Test scenarios are derived from test artifacts like BRS, SRS, etc.	Test case is mostly derived from test scenarios. Multiple Test case can be derived from a single Test Scenario
It focuses on more “what to test” than “how to test”.	A complete emphasis on “what to test” and “how to test.”.
Comparatively less time and resources are required for creating & testing using scenarios.	More resources are needed for documentation and execution of test cases.
Test Scenario provides a small description, mostly one-line statements.	Test cases are more detailed with a number of parameters.

56. Difference between Defect, Error and Failure?

Ans:

Defect	**Error**	**Failure**
A Defect is a variance between expected and actual results. An Error that the tester finds is known as Defect.	The Error is a human mistake. An Error appears not only due to the logical mistake in the code made by the developer.	Failure is a consequence of a Defect. It is the observable incorrect behaviour of the system. Failure occurs when the software fails to perform in the real environment.

The Testers identify the defect. And it was also solved by the developer in the development phase or stage.	The Developers and automation test engineers raise the error.	The End-Users or Realtime users are identify Failures.

The Defect is the difference between the actual outcomes and expected outputs.	An Error is a mistake made in the code; that's why we cannot execute or compile code.	If the software has lots of defects, it leads to failure or causes failure.
The variation between the actual results and expected results is known as defect.	We can't compile or run a program due to coding mistake in a program. If a developer unable to successfully compile or run a program then they call it as an error.	Once the product is deployed and customers find any issues then they call the product as a failure product. After release, if an end user finds an issue then that particular issue is called as failure.

57. What is BVA and ECP?

Ans: **BVA:** Boundary Value Analysis is based on testing the boundary values of valid and invalid partitions. The behavior at the edge of the 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.

Boundary value analysis is a testing technique used in software testing to identify errors in the boundary or edge values of input parameters. It is based on the assumption that most of the errors in a software system occur at the boundaries of the input domain.

The testing technique involves testing the system with input values that are on the boundary, just above the boundary, and just below the boundary. For example, if the input domain of a system is 1 to 100, the boundary values will be 1 and 100, and the values just above and below the boundary will be 2 and 99.

ECP: ECP (Equivalence Class Partitioning) is a software testing technique used to reduce the number of test cases required to test the software while maintaining the test coverage. It is based on the principle that a large number of test cases can be reduced to a smaller set of test cases by grouping the input data into equivalence classes. In ECP, the input domain is

dividedinto different equivalence classes, where each class represents a set of input values that are expected to behave in the same way by the software. The objective of ECP is to identify a representative value from each equivalence class and test it as a test case. This approach helps to reduce the number of test cases required to test the software while ensuring that each class is tested at least once.

For example, if the input domain of a software is a range of numbers from 1 to 100, it can be divided into several equivalence classes, such as valid numbers, invalid numbers, and boundary numbers. The valid numbers class can be further divided into sub-classes such as odd numbers and even numbers. The test cases can be selected from each of these equivalence classes and sub-classes to ensure that the software is tested for all possible input values.

58. What is functional testing and non-functional testing?

Ans: **Functional Testing** is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements. Manual Testing or automation tools can be used for functional testing. The testing is done by providing sample inputs, capturing resulting outputs, and verifying that actual outputs are the same as expected outputs. Functional testing mainly involves black box testing and it is not concerned about the source code of the application. This testing checks User Interface, APIs, Database, Security, Client/Server communication and other functionality of the Application Under Test.

Non-Functional Testing is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per non-functional parameters which are never addressed by functional testing. The primary purpose of non-functional testing is to test the reading speed of the software system as per non-functional parameters. The parameters of non-functional testing are never tested before the functional testing. Non-functional testing is also very important as functional testing because it plays a crucial role in customer satisfaction.

59. What are different types of SDLC models?

Ans: Software Development Life Cycle Models:

There are various software development life cycle models defined and designed which are followed during the software development process. These models are also referred as Software Development Process Models".

1. Waterfall Model
2. V Model:
3. Spiral Model
4. Agile Model

60. Name two parameters that can be useful to check the quality of test execution.

Ans: Two parameters required to check the quality of test execution include:

Defect leakage ratio: It represents the ratio of total potential rejections to the total overall production.

Defect reject ratio: It represents the ratio of total rejections to the total overall production.

61. When to choose manual testing over automation testing and vice versa?

Ans: Choosing Manual Testing over Automation Testing:

→ When test cases need to be run for a short duration of time (once or twice).
→ When one needs to perform exploratory testing, usability testing, or ad-hoc testing.
→ When assessing an application's user-friendliness.
→ Whenever flexibility is needed.
→ Whenever one wants to better manage complex situations/scenarios.

Choosing Automation Testing over Manual Testing

→ Whenever test cases have to be run repeatedly over a long period of time.
→ When one needs to perform performance testing, load testing, or regression testing.
→ Whenever one wishes to record the testing process.
→ When one has a limited amount of time to complete the testing phase.
→ When tests are needed to be executed in a standard runtime environment.

→ When tests involve repetitive steps.

→ When there are multiple and quick deployments for the product, the manual becomes very time taking and redundant.

62. In what way will you determine when to stop testing?

Ans: Testing can be quite challenging when it comes to determining when to stop. In the modern world, many software applications are so complex and run in so many interdependent environments, that complete testing is impossible. The following factors are often considered when deciding when to stop testing:

→ If deadlines are met (release deadlines, testing deadlines, etc.) and there are no high-priority issues left in the system.

→ Completion of test cases with a certain passing percentage.

→ As soon as the test budget is depleted.

→ The mean time between two inherent failures is known as the MTBF (Mean Time Between Failure). When the MTBF is quite high, the testing phase may be stopped depending on stakeholder decisions.

→ As soon as the automated code coverage meets a specified threshold value and there are no critical bugs.

→ If the bug rate drops below a certain level.

→ After the Beta or Alpha testing period has ended.

63. Can 100% testing coverage be achieved? How do you ensure test coverage?

Ans: Testing a product 100% is considered impossible. You can, however, get closer to your goal by following the steps below.

- Developing an effective testing strategy.
- Prepare a checklist for all activities related to testing.
- Establish a priority list for the application's critical areas.
- List all application requirements.
- Identify the risks associated with the application.
- Utilize automated testing

Agile Model Interview Questions and Answers

60. What is Agile Testing?

Ans: Agile Testing takes place in an environment where requirements keep changing according to the need of the customer. In this model, the testing and development activities are concurrent and the testing takes place throughout the development of the software. The testing team keeps receiving frequent small codes from the development team for testing.

61. Define the roles in Scrum?

Ans: There are mainly three roles that a Scrum team have:

1. Project Owner has the responsibility of managing the product backlog. Works with end-users and customers and provides proper requirements to the team to build the proper product.

2. Scrum Master works with the scrum team to make sure each sprint gets completed on time. Scrum master ensures proper workflow for the team.

3. Scrum Team: Each member of the team should be self-organized, dedicated and responsible for the high quality of the work.

62. What is Product Backlog & Sprint Backlog?

Ans: The Product backlog is maintained by the project owner which contains every feature and requirement of the product.

Sprint backlog can be treated as the subset of product backlog which contains features and requirements related to that particular sprint only.

63. Explain the difference between a traditional Waterfall model and Agile testing?

Ans: Agile testing is done parallel to the development activity whereas a traditional waterfall model testing is done at the end of the development.

As done in parallel, agile testing is done on small features whereas, in a waterfall model, testing is performed on the whole application.

64. Explain the Iterative and Incremental Development in Agile?

Ans: Iterative Development: Software is developed and delivered to the customer and based on the feedback again developed in cycles or releases and sprints. Example: Release 1 software is developed in 5 sprints and delivered to the customer. Now, the customer wants some changes, then the development team plan for 2nd release which can be completed in some sprints and so on.

Incremental Development: Software is developed in parts or increments. In each increment, a portion of the complete requirement is delivered.

65. What qualities should a good Agile tester have?

- He should be able to understand the requirements quickly.
- He should know Agile concepts and principals.
- As requirements keep changing, he should understand the risk involved in it.
- The agile tester should be able to prioritize the work based on the requirements.
- Communication is a must for an Agile tester as it requires a lot of communication with developers and business associates.

66. What is the difference between Epic, User stories & Tasks?

Ans:

1. User Stories: It defines the actual business requirement. Generally created by the business owner.
2. Task: To accomplish the business requirements development team create tasks.
3. Epic: A group of related user stories is called an Epic.

67. What is a Taskboard in Agile?

Ans: Taskboard is a dashboard that shows the progress of the project.

It contains:

User Story: It has the actual business requirement.

To Do: Tasks that can be worked on.

In Progress: Tasks in progress.

To Verify: Tasks pending for verification or testing

Done: Completed tasks.

68. What is Scrum ban?

Ans: It is a software development model that is a combination of Scrum and Kanban. Scrumban is considered for maintaining projects in which there are frequent changes or unexpected user stories. It can reduce the minimum completion time for user stories.

69. What is the Zero sprint in Agile?

Ans: It can be defined as a pre-preparation step to the first sprint. Activities like setting development environment, preparing backlog, etc need to be done before starting the first sprint and can be treated as Sprint zero.

70. What is the importance of daily stand up meetings?

Ans: Daily stand-up meeting is essential for any team in which team discuss,

How much work has been completed?

What are the plans to resolve technical issues?

What steps need to done to complete the projects etc?

71. How the velocity of the sprint is measured?

Ans: If capacity is measured as a percentage of a 40 hours weeks then, completed story points * team capacity

If capacity is measured in man-hours, then Completed story points/team capacity

72. How long the Scrum cycle last?

Ans: Basically, the Scrum cycle depends on the project size and team size. Team size may vary from 3 members to 9 members. Normally, it takes 3 to 4 weeks to complete a Scrum sprint. On an average, a scrum sprint ends in 4 weeks.

73. What is the scrum of scrums?

Ans: Suppose there are 7 teams working on a project and each team has 7 members. Each team leads its own particular scrum meeting. Now to coordinate among the teams a separate meeting has to be organized, that meeting is called Scrum of Scrums.

Scrum of Scrums:

An ambassador (a designated person who represents team) represents its team in the scrum of scrums.

Few points discussed in the meeting are:

The progress of the team, after the last meeting.

The task to be done before the next meeting.

Hindrance which the team had faced while completing the last task.

74. What are the principles of agile testing?

Ans: Some major principles of agile testing are:

- Customer satisfaction
- Bug-free clean code
- Changes are welcome by customer
- Whole team, business people and developers work collectively
- Instead of lengthy documentation, focus on the essence
- It focuses on face-to-face conversation
- It promotes sustainable development
- Deliver value to the customer
- Deliver working software frequently
- Simplified and clean code
- Practice continuous improvement
- Respond to change
- Focus on face-to-face conversation
- Less documentation
- Customer and developers work together collectively

- Promote sustainable development

75. What are the disadvantages of the agile model?

Ans: Some of the disadvantages of using the agile model are as follows:

- Not easy to predict: When you encounter a large project, it is not easy to get an idea of how much effort will it require.
- If the guidelines given by the customers are not properly grasped, then final outcome of the project is not as per customer satisfaction.
- Sometimes focusing on design and documentation is not proper
- High-level decisions are under the hand of Veterans, if not combined with non-experienced ones, freshers have little scope to grasp proper knowledge.

76. In what way does agile testing(development) methodology differ from the other testing(development) methodologies?

Ans: In Agile methodology, the code is broken into small parts and at a time, only that particular code is worked or tested. Continuous communication on the particular code part is done by a team so that the focus is only on that particular code. This makes the agile process more flexible and focused.

77. Explain what is a story point in the scrum?

Ans: It can be considered as a unit to estimate the total efforts required to complete or to do the particular task or implementing a backlog.

78. What are the main roles in the scrum?

Ans:

1. Scrum Team: Scrum team is made by an individual person who works collectively to achieve a particular task. The team works in a bond to deliver committed and requested products.
2. Scrum Master: Scrum Master is responsible for the proper execution or working of the scrum team. Being a servant – leader and a coach, he ensures the proper productivity of a team towards scrum sprint goal.
3. Product Owner: The product owner has the responsibility to deliver a complete picture

of what to build and to convey that idea to the team.

79. What is a product burndown chart?

Ans: A description in the form of the graph which shows implemented and not – implemented product backlog is called the burndown chart.

80. What is the defect burn down chart?

Ans: The number of defects identified and removed is represented by the defect burn down chart.

81. What is the sprint burndown chart?

Ans: A graph used to describe no. of implemented/non-implemented sprint in the Scrum cycle.

82. What is the Release burndown chart?

Ans: The graph used to depict the pending release which was earlier planned is called Release burn down the chart.

83. What is the sprint planning meeting?

Ans: A sprint planning meeting is joined by all entities like scrum master, product owner and whole scrum team where they discuss the priority features of the team and product backlog items.

84. What is a Sprint Retrospective meeting?

Ans: This is mostly the last part of the sprint or may be done after the sprint review meeting. Scrum master and the whole team participate in it. They discuss ‘ what was good during the sprint’,’ what was bad’,’ what needs to be improved’. It generally lasts for 2-3 hrs.

85. Tell me something about Kanban?

Ans: Kanban is a tool that helps the team to overlook the work i.e., its progress. Progress, as well as the status of your current development story, is perfectly described using Kanban and more accurately it is done by the ‘Kanban board’.

Kanban board allows you to write the whole scenario of your project at a single place so that you can get a perfect picture of the bottleneck, a task done, workflow progress or basically the complete status of your project.

86. Name three other Agile frameworks?

Ans: Test Driven Development, Feature Driven Development, and Kanban.

87. When to use agile model?

Ans: We can use agile model when the project is large, undefined, complex with unclear requirements. Agile methods are best suited for the project that requires frequent inspection by client so teams and stakeholders can assess and re-prioritize as needed to deliver the most value.

88. When not to use Agile?

Ans: The Project with a fixed scope and have everything pre-defined with little to no uncertainty don't require Agile methods.

Jira Tool Interview Questions and Answers

89. What is Jira?

Ans: JIRA is a software tool that was developed by the software company, Atlassian. JIRA is an issue and project tracking software to plan, track and manage your projects. JIRA is mainly used by agile development teams to customize your workflows, team collaboration, and release software with confidence. It is the perfect solution for organizing tasks and managing agile teams.

JIRA is an efficient tool and has the capability to track any kind of defects/issues.

90.Explain the workflow of JIRA?

Ans: JIRA Workflow is the series of stages or steps a bug/issue follows during its lifecycle from creation to closing or completion.

The important phases in the workflow are:

- Created/Open
- Work in Progress (WIP)
- Completed/Closed

91. What is an issue in JIRA?

Ans: An issue can be:

- A Software bug
- The Project Task
- The Form for Leave-request
- A Help-desk Ticket

92. What are some of the benefits of using Jira?

Ans: The following are some of the benefits of using Jira:

- It is easily customizable and extensible.
- It runs almost anywhere as it is platform-independent. It is recognized by quite a few well-known companies.
- We can get the latest update on the progress of projects via Jira.
- It has an upfront and fair licensing policy.

93. What are the agile methodologies supported by Jira?

Ans: Jira Software supports the following agile methodologies: Scrum and Kanban.

Scrum: Scrum is an agile methodology in which the development team works in an iterative manner to complete a given project. Every sprint or iteration has some set scope and timeline for the project. For software development projects, Scrum is most suitable.

Kanban: Kanban is an agile methodology that focuses on just-in-time delivery. It accomplishes this by visualizing the workflow and the tasks in progress. Kanban is mostly suitable for operation teams.

94. What are the issue types in a Scrum project in Jira?

Ans: An individual unit of work in Jira is referred to as an issue. This issue could be a unit such

as a story or an epic. Every issue has a field called issue type, which represents the type and use of the issue. For instance, in a Scrum project, one would have the following types of issues by default:

Story: A story in Jira represents a single feature that is to be implemented. It is generally used in order to get the requirements from the end user's perspective. For this purpose, stories are often written in non-technical language. These are used for focusing on the end results of the feature.

Epic: An epic means a big user story. This represents that the story has not been broken down into smaller and finer requirements. In Jira, epics are mostly used to define the "theme" for various stories that will be part of it, along with the modules or the major components in a big development project.

Bug: A bug in Jira represents a problem or a defect that needs to be fixed in a given product.

Task: This represents a generic task that is neither a bug nor a story, but it needs to be completed.

95. What are some of the popular add-ons for Jira?

Ans: Following are some of the popular add-ons for JIRA:

- Zephyr for JIRA – Test Management
- JIRA Toolkit Plugin
- JIRA Charting Plugin
- Portfolio for JIRA
- Suite's utilities for JIRA
- ScriptRunner for JIRA
- Atlassian REST API Browser
- Tempo Timesheets for JIRA
- JIRA Misc Workflow Extensions

96. What is an Event in Jira?

Ans: In Jira, an event gives information about the status, the default template and the

notification scheme and workflow transition post function associations for the event. Basically, the events are classified into these two categories:

- A System event (Jira defined events)
- Custom event (User-defined events)

97. What are the report types that are generated in Jira?

Ans: In Jira, there are a number of reports generated that are used to showcase different project statistics throughout the entire life cycle of the project. There are general reports available for analyzing issues in Jira. Along with this, there are also some reports for Scrum and Kanban projects.

The general reports available for analyzing the issues includes:	**For Scrum projects, the following types of reports can be generated:**
Time Tracking Report	Sprint Report
Version Workload Report	Version Report
Workload Pie chart Report	Control chart
Created vs Resolved issue Report	Velocity chart
Average Age Report	Cumulative Flow diagram
User Workload Report	Release Burndown
Resolution Time Report	Burndown chart
Pie Chart Report	Epic Report
Recently created Issue Report	

98. Explain the three colour indicators to show time spent in Jira and their significance.

Ans: In Jira, for a particular issue, three colour, s namely Blue, Green, and Orange are used to show how much time is spent on that particular issue. You can go to the section of 'Time Tracking' to view this information. The significance of each colour is as follows:

Blue: The blue colour is used to show the 'Original Estimate'. This is the estimate of time to be spent in resolving an issue. You can see this field shown as 'Estimated'.

Orange: The orange colour is for the time left to resolve an issue. You can see this field shown

as ‘Remaining’.

Green: The green colour shows the actual time that has been spent in solving given the issue till now. You can see this field shown as ‘logged’.

99. For an Agile project, how are user stories created in Jira?

Ans: For an Agile project, the user stories are created in the following way in Jira:

Issue type -> Epic and Issue type –> Story linked to it. To do so, on the ‘Create Issue’ page, go to “Configure Fields”. Then select the “Epic Link” field to be included in the issue creation screen.

Alternatively, you can also have a product backlog by creating one main User story and having multiple sub-tasks under it.

100. Why are the issues in Jira labelled?

Ans: The issues in Jira are labelled in order to categorize an issue within a particular section. This can help to easily search with the help of labels. Label for a given issue can be initially set at the time of creating the issue. It can also be edited within the issue.

101. What does the Jira Schema consist of?

Ans: The Jira Schema consists of the following:

Notifications: This indicates what email someone receives when an issue is changed.

Workflows: This indicates which workflow is used for each issue type.

Permissions: This indicates what changes can be made to an issue by someone.

Issue types: This indicates what issue types like Bug, New Feature etc. can be used in a given Jira project.

Screens: This indicates where the fields are displayed in an issue’s screen.

Field configurations: This is used to define which Field Configuration is used for each issue type.

102. What are some of the common Jira add-ons?

Ans: Some of the common and useful Jira add-ons are as follows:

1. Slack
2. Github
3. Tempo Timesheets
4. PagerDuty
5. Jenkins-CI
6. Usersnap

103. What do you mean by Cloning an Issue in JIRA?

Ans: Cloning an issue basically means copying an issue. This allows us to quickly make a copy of an issue inside a similar project. The clone issue is a mirror image of the original issue. It contains similar data that is put in the original issue like the Summary, Components, Affects Versions, and so on. A clone issue is a different element from the original issue. The clone issue can similarly be connected to the original issue.

The tasks on the original issue have no impact on the clone issue and also the other way around. If a link is made between the original issue and the clone issue, that serves as the main connection.

Cloning an issue retains the following:

- Project
- Summary
- Issue Links
- Description
- Assignee
- Environment
- Reporter
- Attachments
- Components
- Affects Versions
- Fix For Versions
- Priority

- Issue Type
- Security

104. Which issues cannot be cloned?

Ans: The following are the issues that are cannot be cloned:

- Time tracking
- Issue history
- Comments and
- Links to Confluence pages

105.Can we back up data in Jira Cloud?

Ans: Yes, Jira provides backup functionality of data. But it can save backup files only once and next time when you backup, the existing data will be overwritten.

106. What is scheduling an event in Jira?

Ans: Scheduling an event is activated in order to trigger action with respect to the issue. And to perform this scheduling, one should request "Schedule issue permission" from the administrator. This provides a "due date" to an issue to be scheduled for.

107. What are the important differences between JIRA and Bugzilla?

Ans: Let me explain a few important differences between JIRA and Bugzilla

JIRA	**Bugzilla**
JIRA is a commercial software tool	Bugzilla is an open-source software tool
JIRA is a user-friendly software tool	Bugzilla is not a user-friendly tool
Drag and drop issue options are available in JIRA	Drag and drop issue options are not available in Bugzilla
JIRA has a configuration link types which contains user-defined semantics	Bugzilla has only one configuration link type

JIRA has a pluggable debugging links and user can access them outside of the tool	Supports blocks/user depends on the custom option available
JIRA software tool allows customized configuration based on the project type and type fields.	Bugzilla owns show/hide the customized common field or specific values.

108. Is it possible to access the JIRA cloud site via a mobile device?

Ans: The answer is yes, the user can access the JIRA cloud site via a mobile device, for that you have to use the appropriate URL of the JIRA mobile cloud site in your mobile website browser.

109. Can JIRA be used for test case management?

Ans: As we know, the JIRA architecture was not designed to work well with the server as a test case management tool. So, it will be configured to support the test case management system in two different possible ways. After that users can either change the native JIRA components to meet the test case management system and also you can use add-ons option available in the parent company (Atlassian marketplace).

110. How to create a burndown and burn up charts in JIRA?

Burn up chart creation steps:	**Burndown chart creation steps:**
Go to the JIRA official web page.	Go to the JIRA official web page.
Now click on the project that you are working on.	Now click on the project that you are working on.
Click on the reports -> appears at the left side	Click on the reports ->appears on the left side of the panel.
Now it's time to select the Burn-up chart option.	Now select the burn down from the reports drop-down option.

111. For any agile project, how user stories in JIRA are created?

Ans: The following steps will explain how to create user stories in JIRA;

Issue type – epic and issue types ->then the agile stories linked to it. To perform this, follow the navigation "create issue "page -> then go to the "configure fields" -> select the "epic link".

Or the user can have a product backlog feature -> by creating the main user story -> create sub-tasks under it.

112. Define JQL.

Ans: JQL means Jira Query Language, which offers advanced and flexible ways to discover various issues in JIRA. The JQL includes field, operator, keyword, and value. It is useful for developers, managers, testers, business users, etc.

113. Name the Agile methodologies that support JIRA.

Ans: Kanban and Scrum are two popular Agile methodologies that support JIRA.

114.What is Zephyr for Jira?

Ans: Zephyr for JIRA is an add-on application that provides highly sophisticated test management capabilities right inside JIRA.

115. Is Jira agile or Scrum?

Ans: Jira is a popular agile project management tool that supports agile methodologies like Kanban and Scrum. It allows the project teams to adapt the agile practices easily. However, while configuring Jira software, a major confusion among the project teams is whether to leverage the Scrum board or the Kanban board.

******THE END********