

# Software Testing

Shreyata Sugandhi

Corporate Trainer & Consultant

[shreyata@goldentouchsolutions.tech](mailto:shreyata@goldentouchsolutions.tech)

<https://www.linkedin.com/in/shreyatasugandhi/>

# Types of Application

There are mainly 5 types of Applications

- Web based application
  - GUI based Web Application
  - Web Services
- Desktop applications
- Operating Systems
- Mobile apps
  - Android
  - iPhone
  - Symbian
  - Linux
  - Kai
- Database applications
  - SQL databases (Oracle, MySql, MS SQL)
  - NoSQL databases (mongo, aerospike)

# What is Software Testing?

- Software testing is a process used to identify the correctness, completeness, and quality of developed computer software.
- In simple words, “software testing is an activity to check whether the actual results match the expected results and to ensure that the software system is defect free”.
- Validating and Verifying that a software program/application/product
  - meets the business and technical requirements that guide its design and development;
  - works as expected; and
  - can be implemented with the same characteristics.

# Why is it important?

- Software Testing is important to ensure the quality of the software
- To create a bug free software
- Testing is important because software bugs could be expensive or even dangerous
- To provide stakeholders with information about the quality of the product or service under test
- To provides an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation

# Principles of Software Testing

Principle No	Description
Principle 1	Testing shows presence of defects
Principle 2	Exhaustive testing is impossible
Principle 3	Early Testing
Principle 4	Defect Clustering
Principle 5	Pesticide Paradox
Principle 6	Testing is context dependent
Principle 7	Absence of errors - fallacy

# Who does Testing?

- Software Tester
- Software Developer
- Project Lead/Manager
- End User

# When to Start Testing?

- Testing is done in different forms at every phase of SDLC:
  - Requirement gathering phase - analysis and verification of requirements are also considered as testing.
  - Designing phase - reviewing design with the intent to improve the design is also considered as testing.
  - Development phase - Unit Testing
  - Testing Phase – Functional and non-functional testing
  - Deployment phase – verification of deployment
  - After the release – User Acceptance testing (Alpha & Beata)

# When to Stop Testing?

- It is difficult to determine when to stop testing, as testing is a never-ending process and no one can claim that a software is 100% tested.
- But testing can be stopped on -
  - Testing Deadlines
  - Completion of test case execution
  - Completion of functional and code coverage to a certain point
  - Bug rate falls below a certain level and no high-priority bugs are identified
  - Management decision



# Bugs can be

- Defect
- Fault
- Problem
- Error
- Incident
- Anomaly
- Variance
- Failure
- Inconsistency
- Product Anomaly
- Product Incidence
- Feature

# Software Testing - Myths

- **Myth:** Anyone can do manual testing

**Fact:** Testing requires many skill sets

- **Myth:** Testing ensures 100% defect free product

**Fact:** Testing attempts to find as many defects as possible. Identifying all possible defects is impossible.

- **Myth:** Automated testing is more powerful than manual testing

**Fact:** 100% test automation cannot be done. Manual Testing is also essential.

- **Myth:** Testing is easy

**Fact:** Testing can be extremely challenging .Testing an application for possible use cases with minimum test cases requires high analytical skills.

# Software Testing - Myths

- **Myth:** Testing is Too Expensive

**Fact:** Early testing saves both time and cost in many aspects, however reducing the cost without testing may result in improper design of a software application rendering the product useless.

- **Myth :** Testing is Time-Consuming

**Fact:** During the SDLC phases, testing is never a time-consuming process. However diagnosing and fixing the errors identified during proper testing is a time-consuming but productive activity.

- **Myth:** Only Fully Developed Products are Tested

**Fact:** Testing is being done in each and every phase of SDLC – reviews, analysis, unit testing, functional & non functional testing, deployment checks, Alpha & Beata Testing

# Verification & Validation

S.N.	Verification	Validation
1	Verification addresses the concern: "Are you building it right?"	Validation addresses the concern: "Are you building the Product right?"
2	Ensures that the software system meets all the functionality.	Ensures that the functionalities meet the intended behavior.
3	Verification takes place first and includes the checking for documentation, code, etc.	Validation occurs after verification and mainly involves the checking of the overall product.
4	Done by developers.	Done by testers.
5	It has static activities, as it includes collecting reviews, walkthroughs, and inspections to verify a software.	It has dynamic activities, as it includes executing the software against the requirements.
6	It is an objective process and no subjective decision should be needed to verify a software.	It is a subjective process and involves subjective decisions on how well a software works.

# QA, QC & Testing

Quality Assurance	Quality Control	Testing
QA includes activities that ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements.	It includes activities that ensure the verification of a developed software with respect to documented (or not in some cases) requirements.	It includes activities that ensure the identification of bugs/error/defects in a software.
Focuses on processes and procedures rather than conducting actual testing on the system.	Focuses on actual testing by executing the software with an aim to identify bug/defect through implementation of procedures and process.	Focuses on actual testing.
Process-oriented activities.	Product-oriented activities.	Product-oriented activities.
Preventive activities.	It is a corrective process.	It is a preventive process.
It is a subset of Software Test Life Cycle (STLC).	QC can be considered as the subset of Quality Assurance.	Testing is the subset of Quality Control.

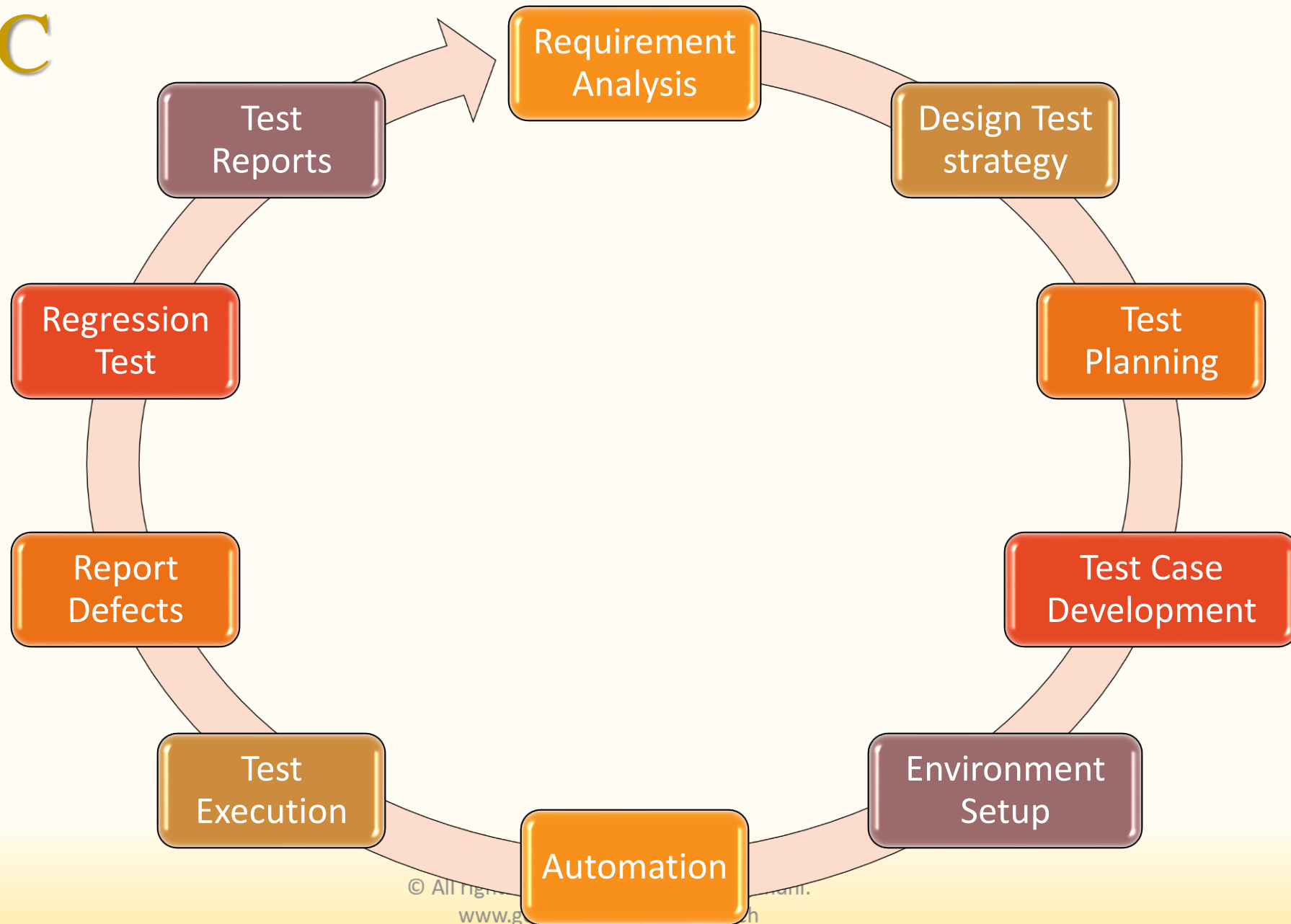
# Testing and Debugging

- **Testing:**
  - It involves identifying bug/error/defect in a software without correcting it.
  - Normally, QA involved in bugs identification.
  - Testing is performed in the testing phase.
- **Debugging :**
  - It involves identifying, isolating, and fixing the problems/bugs.
  - Developers when encountering an error in the code do debugging.
  - Debugging is a part of White Box Testing or Unit Testing or analysis while fixing the reported bugs.

# Manual and Automated Testing

	Manual Testing	Automation Testing
Pros	<ul style="list-style-type: none"><li>• Easy way to improve quality</li><li>• Focused on customer's workflow primarily</li><li>• Only testing skills are required</li><li>• Non-programmable, hence easy</li><li>• Very useful for non-stable application</li><li>• Can be done in any phase of the software</li></ul>	<ul style="list-style-type: none"><li>• Faster test cycles</li><li>• Can find more defects in a shorter time frame</li><li>• Saves cost after the 3<sup>rd</sup> execution/run</li><li>• Excellent way to meet testing needs of Agile development</li><li>• Less investment in human resources</li><li>• More reliable</li></ul>
Cons	<ul style="list-style-type: none"><li>• May not identify all the defects</li><li>• Time consuming</li><li>• More expensive as needs more skilled human recourses all the time</li><li>• Exhaustive testing is not possible</li><li>• Low product quality as higher defect count.</li></ul>	<ul style="list-style-type: none"><li>• Programming knowledge is must</li><li>• Requires manual tests to be written first and then it can be converted.</li><li>• Requires automation testing platform</li><li>• No cost effective for less than 3 cycles</li><li>• Not useful for non-stable app</li></ul>

# STLC





# Testing Methodology

- **Black Box** - Black Box testing refers to the technique of testing a system *with no knowledge* of the internals of the system, no access to source code and system architecture. A Black Box tester typically tests the functionalities provided by the web services.
  - Advantages - Efficient Testing, Unbiased Testing, Non intrusive, Easy to execute
  - Disadvantages - Localized Testing, Inefficient Test Authoring, Blind Coverage
- **White Box** - White Box testing refers to the technique of testing a system with knowledge of the internals of the system, access to source code and system architecture. A White Box tester typically analyzes source code, derives test cases from knowledge about the source code, and finally targets specific code paths to achieve a certain level of code coverage.
  - Advantages - Increased Effectiveness, Full Code Pathway Capable, Early Defect Identification, Reveal Hidden Code Flaws
  - Disadvantages - Difficult To Scale, Difficult to Maintain, Cultural Stress, Highly intrusive

# Testing Methodology

- **Grey Box** - Gray Box testing refers to the technique of testing a system with limited knowledge of the internals of the system, access to detailed design documents with information beyond requirement documents. Gray Box tests are generated based on information such as state-based models or architecture diagrams of the target system.
  - Advantages - Offers Combined Benefits, Non Intrusive, Intelligent Test Authoring, Unbiased Testing
  - Disadvantages - Partial Code Coverage, Defect Identification,

# Testing Levels

- **Governance Testing** - ensure each new and existing service conforms to the standards, policies and objectives of an organization for the entire life of that service
- **Service-component-level Testing / Unit Testing** - to test the code not only successfully compiles, but the basic functionality of the components and functions within a service are working as specified
- **Process Testing** - Process/Orchestration testing ensures services are operating collectively as specified
- **Integration Testing** - determine if interface behavior and information sharing between the services, are working as specified

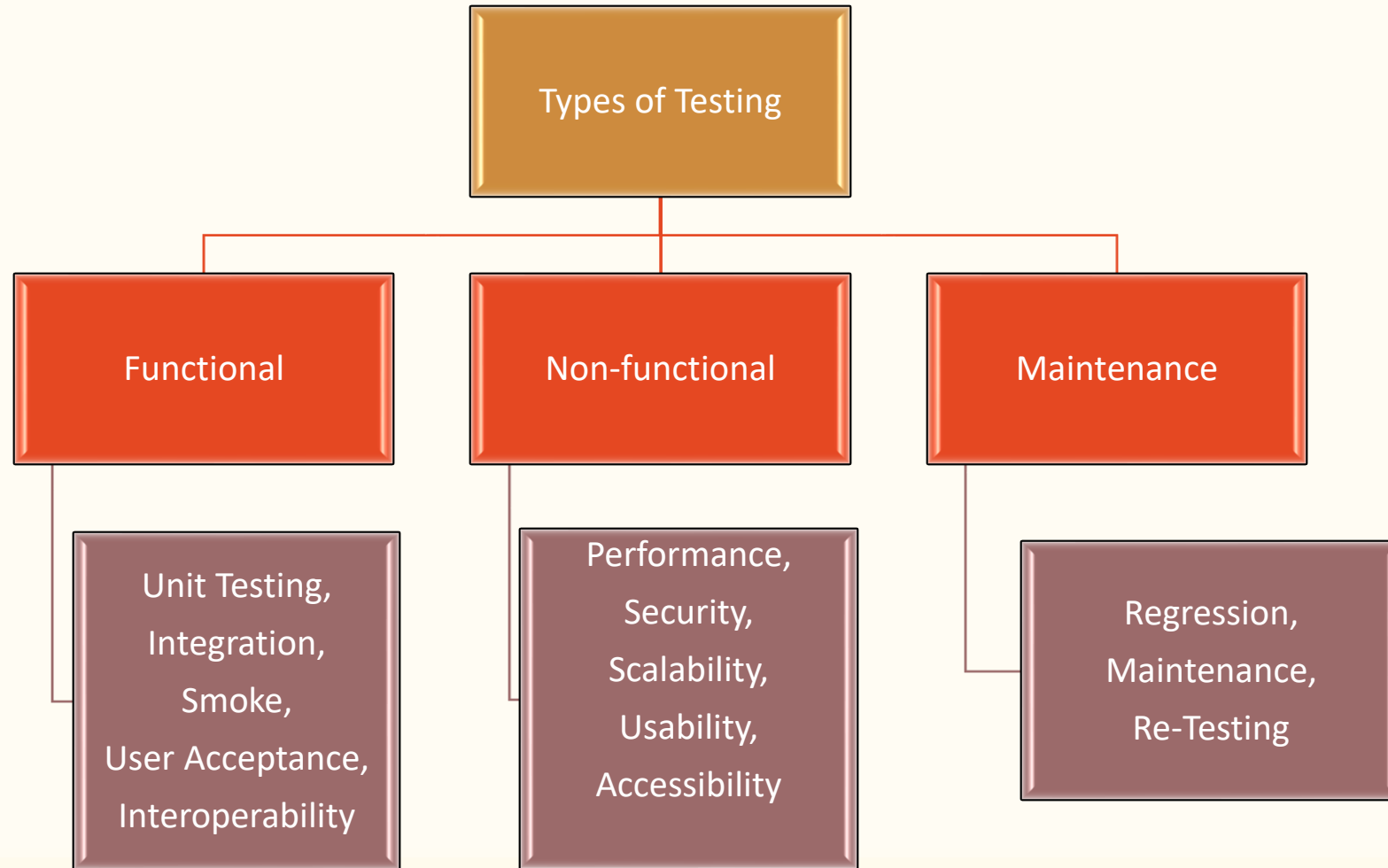
# Testing Levels

- **System Testing** – defined business requirements and has met the defined business acceptance criteria
- **Security Testing** – to ensure protection from unauthorized software
- **Service level Testing** - Service reuse will demand each service is delivered from this level/phase of testing with a comprehensive statement of quality and even a guarantee. This includes Functional Testing, Performance Testing and Security Testing.

# Testing Levels

- **User Acceptance Testing** - User acceptance is a type of testing performed by the Client to certify the system with respect to the requirements that was agreed upon. This testing happens in the final phase of testing before moving the software application to Market or Production environment.
  - Alpha - Alpha testing is carried out in a lab environment and usually the testers are internal employees of the organization.
  - Beta - Beta Testing of a product is performed by "real users" of the software application in a "real environment" and can be considered as a form of external User Acceptance Testing.

# Types of Testing



# Functional Testing

- The prime objective of Functional testing is checking the functionalities of the software system. It mainly concentrates on
  - **Mainline functions:** Testing the main functions of an application
  - **Basic Usability:** It involves basic usability testing of the system. It checks whether an user can freely navigate through the screens without any difficulties.
  - **Accessibility:** Checks the accessibility of the system for the user
  - **Error Conditions:** Usage of testing techniques to check for error conditions. It checks whether suitable error messages are displayed.

# Functional Vs Non Functional

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 may or may not 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 Vs Non Functional

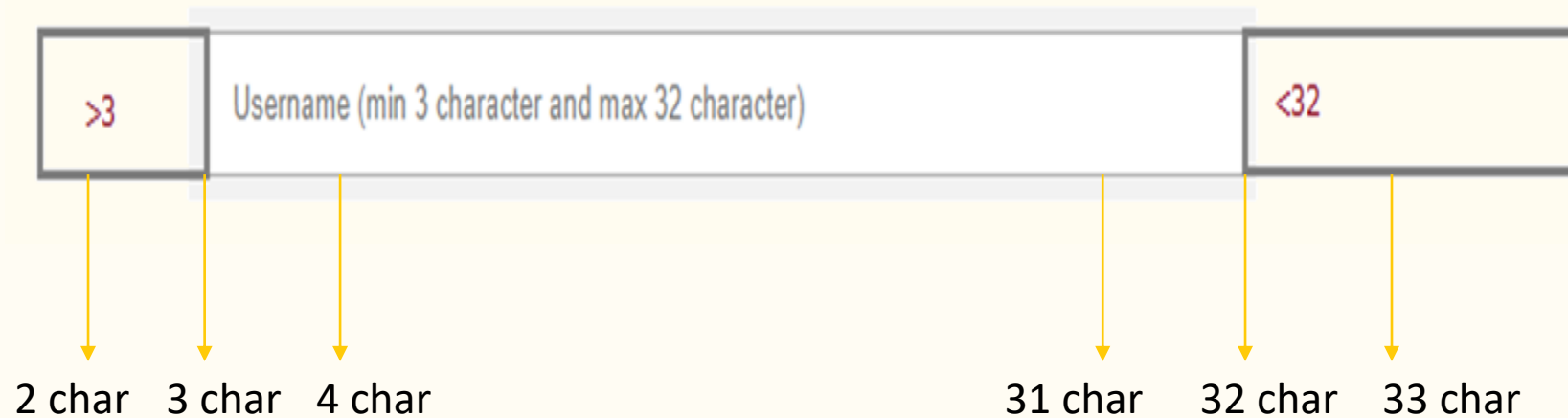
Functional Testing	Non-Functional Testing
Business requirements are the inputs to functional testing	Performance parameters like speed , scalability are inputs to non-functional testing.
Functional testing describes what the product does	Nonfunctional testing describes how good the product works
Easy to do manual testing	Tough to do manual testing
<ul style="list-style-type: none"><li>•Types of Functional testing are Unit Testing</li><li>•Smoke Testing</li><li>•Sanity Testing</li><li>•Integration Testing</li><li>•White box testing</li><li>•Black Box testing</li><li>•User Acceptance testing</li><li>•Regression Testing</li></ul>	<ul style="list-style-type: none"><li>•Types of Non functional testing are Performance Testing</li><li>•Load Testing</li><li>•Volume Testing</li><li>•Stress Testing</li><li>•Security Testing</li><li>•Installation Testing</li><li>•Penetration Testing</li><li>•Compatibility Testing</li><li>•Migration Testing</li></ul>

# Testing Techniques

- Software testing Techniques help you design better cases.
- Since exhaustive testing is not possible; Testing Techniques help reduce the number of test cases to be executed while increasing test coverage.
- They help identify test conditions that are otherwise difficult to recognize.

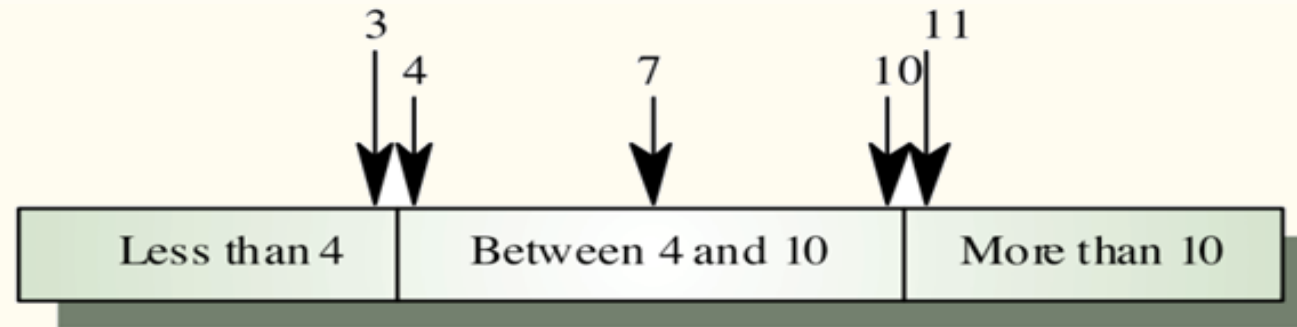
# Testing Techniques

- Boundary Value Analysis –

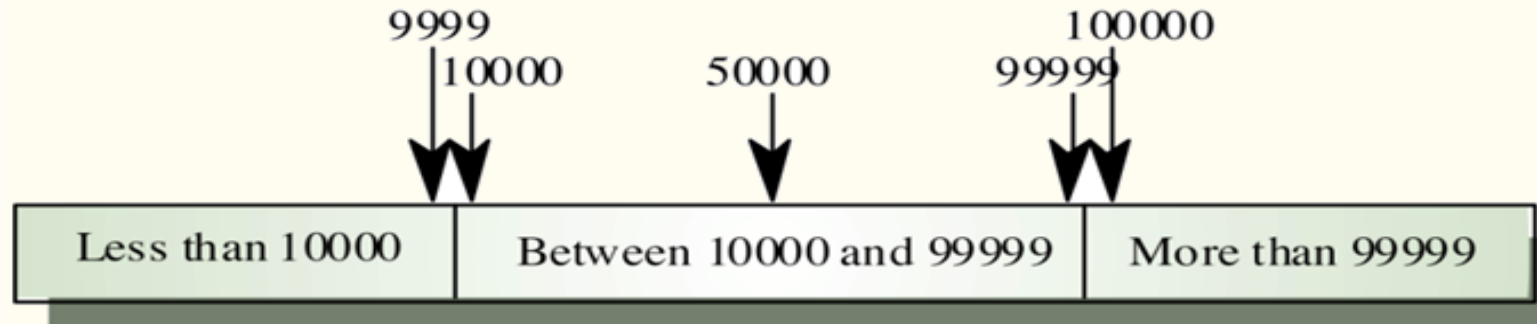


# Testing Techniques

- Equivalence Class Partitioning



Number of input values



Input values

# Testing Techniques

- Decision Table based testing OR Cause-Effect table

Enquiry Form

We are glad that you preferred to contact us. Please fill our short form and one of our friendly team members will contact you back.

Your Name (required)

Your Email (required)

Your Number (required)

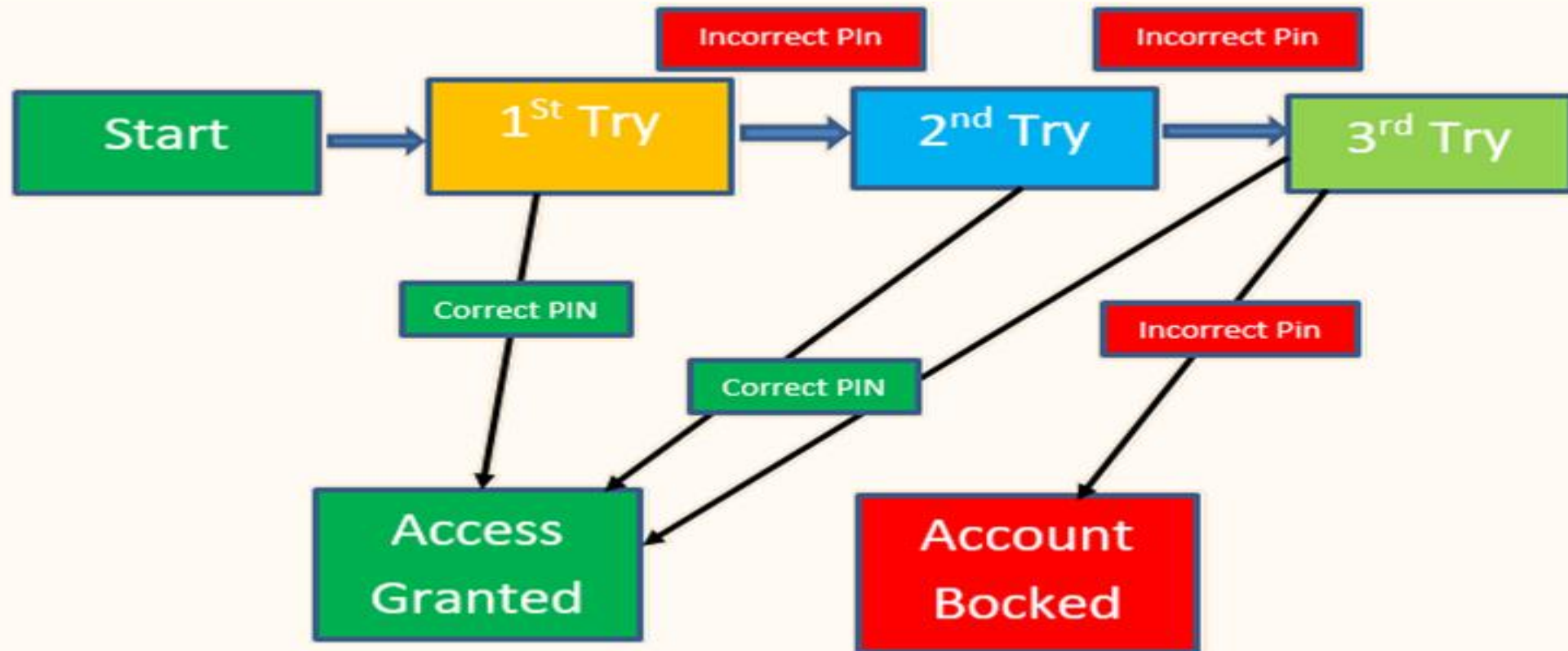
Your Message

SEND

	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
Input					
Name	F	T	T	T	T
Email	F	F	T	T	T
Number	F	F	F	T	T
Message	F	F	F	F	T
Send	F	F	F	F	T

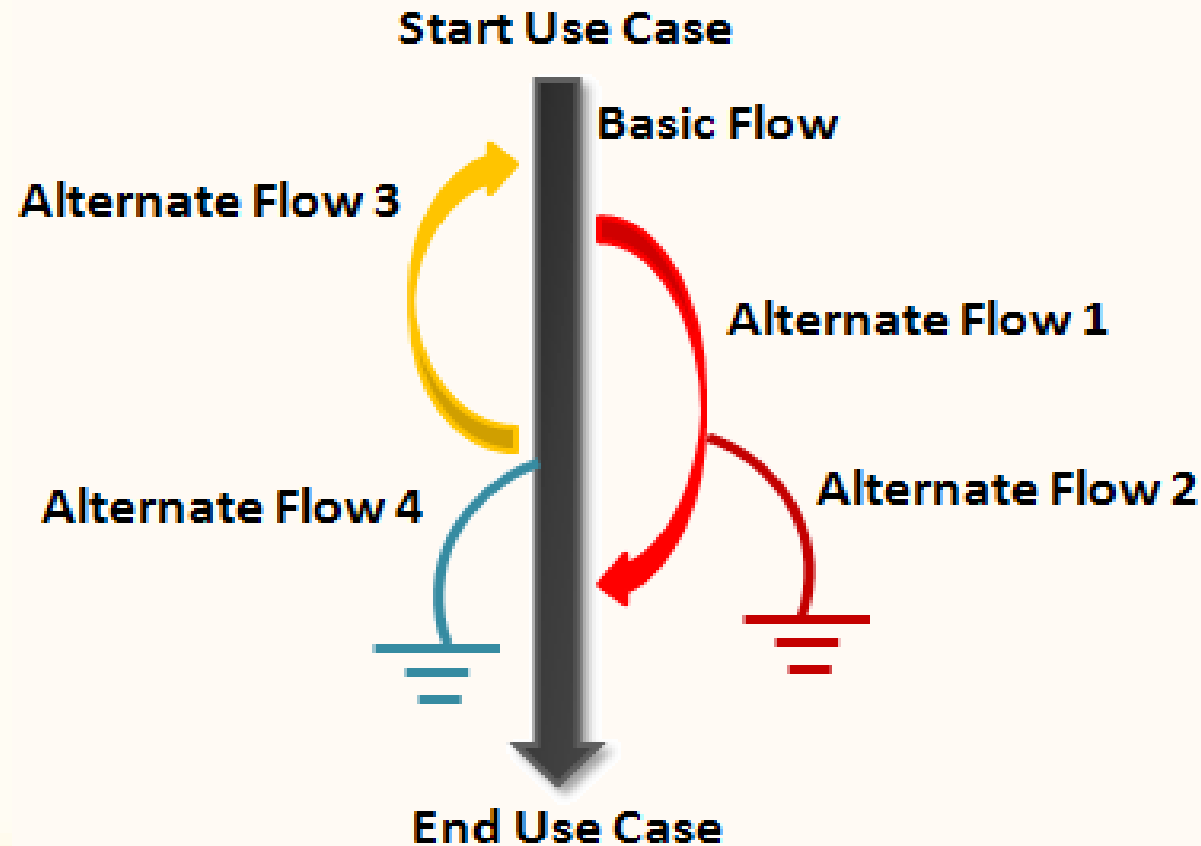
# Testing Techniques

- State Transition



# Testing Techniques

- Use case Testing



# Testing Techniques

- Exploratory Testing –
  - **First of all, the exploratory testing is not a random or ad-hoc test.**
  - One of the biggest misconceptions about this test technique is that the exploratory testing is perceived as a random, non-testable, non-observable test technique.
  - **The exploratory testing is a test approach based on learning and exploring the product at the same time by using the experience, domain knowledge, analytical and intellectual knowledge of a test engineer in agile processes.**





# Testing Techniques

- Experienced Based Testing -
  - This testing technique **based on the knowledge, skills, and experience of the person** who will make the test.
  - In this testing technique; test planning, test strategy, test inputs, and test scenarios are determined by the experience of the person performing the test.
  - When we have very short test execution time or there is a lack of sufficient documentation on the project, etc. make sense to use this test technique.
- Checklist Based Testing –
  - We normally use Test Checklist when we don't have enough time on Test Preparation Phase to prepare Test Case.
  - It is a smaller version of Test Case where you only define which Cases/Scenarios those are need to be tested without putting plenty of information into them.

# Testing Techniques

- Example –
  - All links in the system (Web / Mobile) should work correctly.
  - There should not be a grammar error in the system writings.
  - Font sizes, fonts, should be as expected.
  - There should not be any pictures that can not be loaded/broken in the system.
  - Pictures, text, etc. The alignment between the other components must be as expected.
  - All buttons must work properly and each of them directs the user to the corresponding operation.
  - Each page should have the main page logo and should be redirected to the main page when clicked.
  - Warning, information messages should be displayed in the correct format.
  - If the page is responsive, it should be checked at all resolutions.
  - All components on the site (dropdown, checkbox, radio button, etc.) should work correctly.
  - Special conditions (numeric, alphanumeric, etc.) in the input fields must be checked.
  - Operations can not be performed by leaving the required fields blank.
  - Any operation of the site must not last more than 3 to 15 seconds.
  - ... etc.

# Testing Techniques

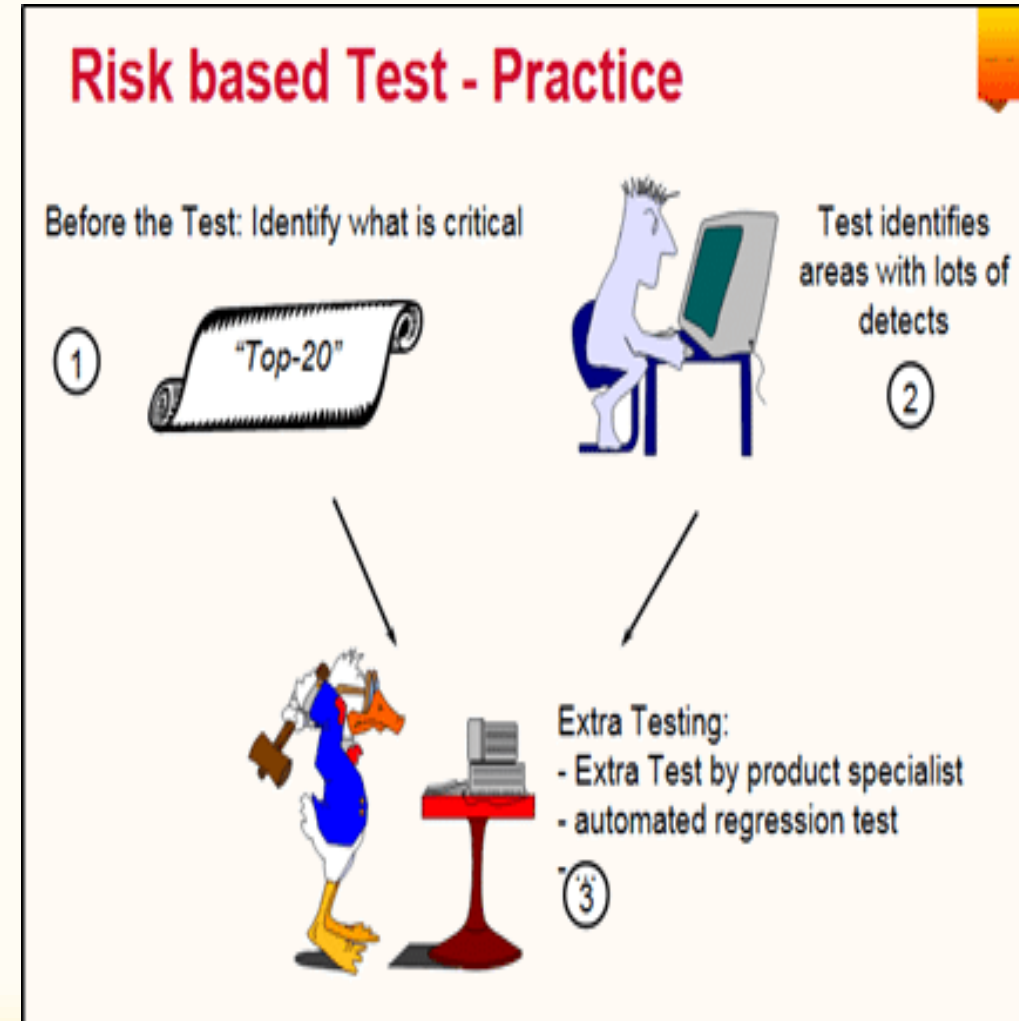
- Ad-hoc testing –
  - Testing is done based on the skills, intuition, and experience. There are no strong test cases for this type of testing.
- Error guessing / Experienced Based Testing –
  - Error guessing is a software testing technique which is based on guessing the error which can prevail in the code. It is an experience-based technique where the test analyst uses his/her or experience to guess the problematic part of the testing application.
  - Guidelines for Error Guessing:
    - The test should use the previous experience of testing similar applications
    - Understanding of the system under test
    - Knowledge of typical implementation errors
    - Remember previously troubled areas
    - Evaluate Historical data & Test results

# Testing Techniques

- User Journey Test -
  - The user's **interactions with the system** are covered as much as possible.
  - These tests are usually “**end-to-end**” tests, so they may take more time to run than other tests, but the coverage percentage of these tests are higher than the other ones.
  - “User journey” tests can be created taking into account the most critical “use-case” scenarios & “**happy-path**”, **should be run first**.
  - The broad coverage of the “user journey” tests beneficial in early detection of critical faults

# Testing Techniques

- Risk-Based Testing –
  - Objectives of risk-based test techniques is to find the most critical and most important errors as early as possible with lowest cost
  - In risk-based tests, we prioritise and test the most error-prone functionalities
- **Magnitude of Risk = Likelihood \* Impact**
- The most basic steps of the risk-based test are –
  - First, risks are identified and a prioritized risk list is prepared.
  - Make a test plan according to the prioritized risk list and tests are executed for each risk.
  - As a result of the tests, some risks eliminate and some of them arise. New risks are tested by considering their test effort.



# Positive & Negative Testing

Enter Only Numbers

99999

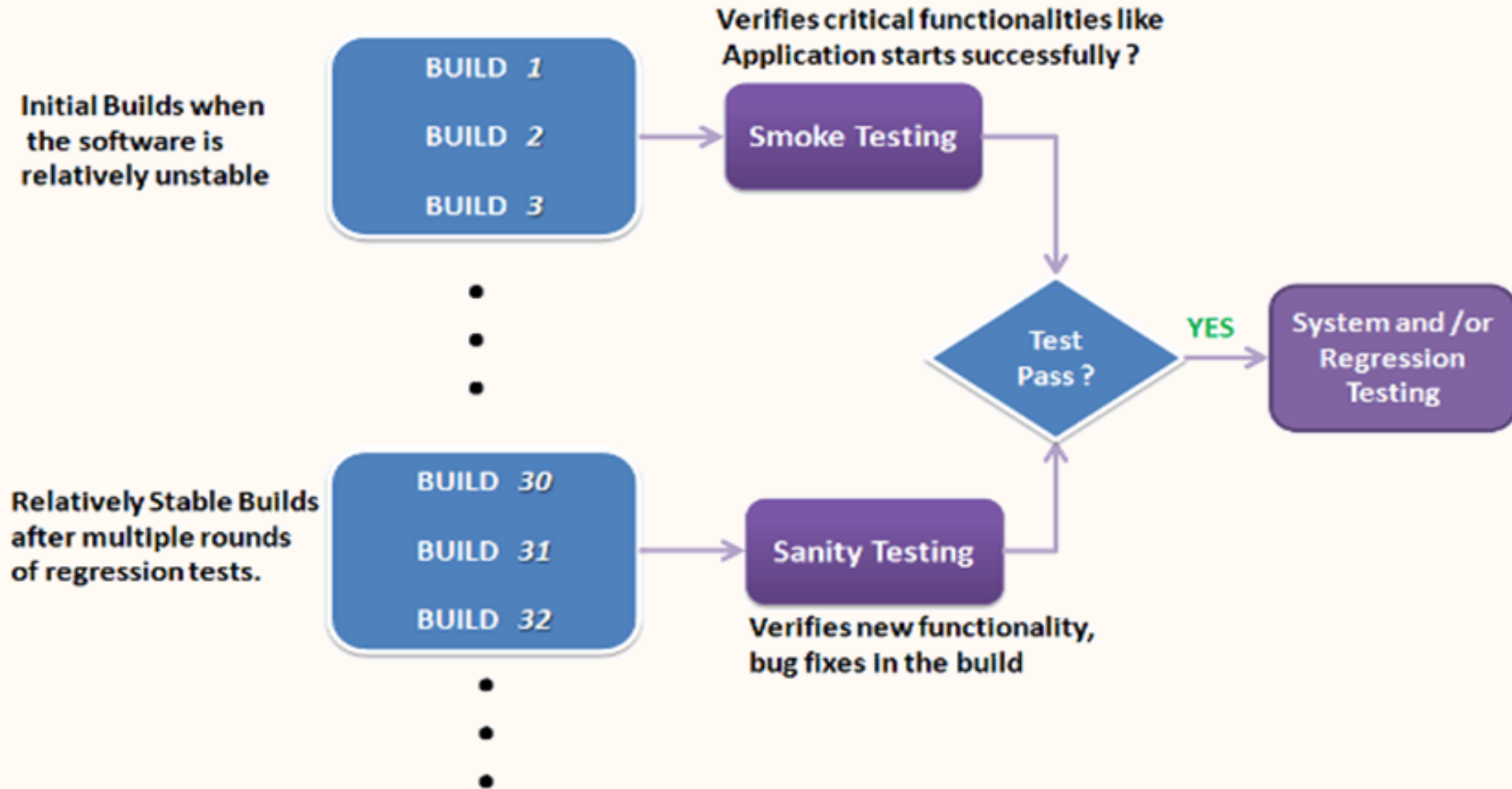
**Positive Testing**

Enter Only Numbers

abcdef

**Negative Testing**

# Sanity & Smoke Testing



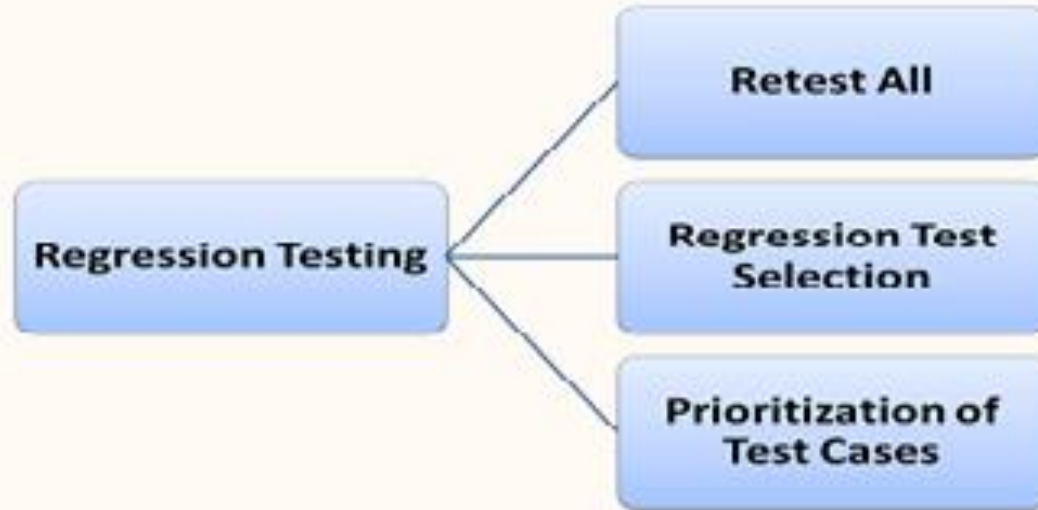
# Smoke & Sanity Testing

Smoke Testing	Sanity Testing
Smoke Testing is performed to verify that the critical functionalities of the program is working fine	Sanity Testing is done to check the new functionality / bugs have been fixed
Objective - to verify the "stability" of the system in order to proceed with more rigorous testing	The objective of the testing is to verify the "rationality" of the system in order to proceed with more rigorous testing
This testing is performed by the developers or testers	Sanity testing is usually performed by testers
Smoke testing is usually documented or scripted	Sanity testing is usually not documented and is unscripted
Smoke testing is a subset of Regression testing	Sanity testing is a subset of Acceptance testing
Smoke testing exercises the entire system from end to end	Sanity testing exercises only the particular component of the entire system
Smoke testing is like General Health Check Up	Sanity Testing is like specialized health check up



# Types of Testing

- **Regression Testing** - Regression tests are to find defects that got introduced to defect fix(es) or introduction of new feature(s).



- **Retesting** - Retesting is carried out by software testers as a part of defect fix verification.
- **Incremental integration testing** - continuous testing of an application as new functionality is added. Application functionality and modules should be independent enough to test separately. done by programmers or by testers.

# Types of Testing

- **Ad-hoc testing** - Very informal and unstructured and can be performed by any stakeholder with no reference to any test case or test design documents.
- **Accessibility Testing** - In accessibility testing, the aim of the testing is to determine if the contents of the website can be easily accessed by disable people.
- **API Testing** - API testing is a type of testing that is similar to unit testing. Each of the Software APIs are tested as per API specification.
- **Reliability Testing** – Validate that software operates under stated conditions for a specified time period
- **Component Testing** - Component testing involves testing a group of units as code together as a whole rather than testing individual functions, methods.

# Types of Testing

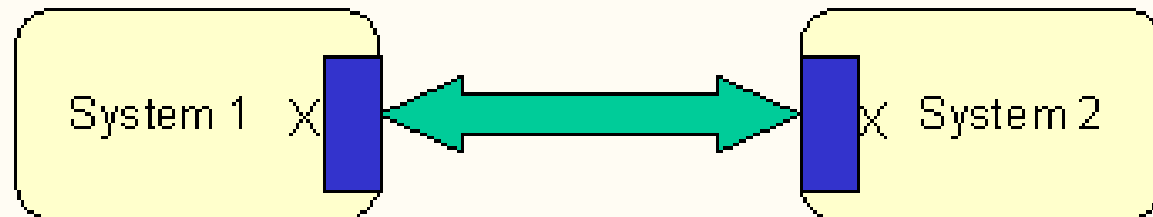
- **Compatibility testing** - Compatibility testing checks if the software can be run on different hardware, operating system, bandwidth, databases, web servers, application servers, hardware peripherals, emulators, different configuration, processor, different browsers and different versions of the browsers etc.
  - There are two types of Compatibility Testing
  - Backward compatibility - determine if changes to an interface will affect existing users of the interface
  - Forward compatibility - a system that is designed to fit with planned future versions of itself.

# Performance Testing

- **Definition** - A non-functional test to validate capacity and reliability.
- **Need** - To ensure a good user experience (fast and error free) under any user load.

# Interoperability Testing


- **Interoperation** is the ability of two or more systems or components to exchange information and to use the information that has been exchanged.
- Interoperation is a peer to peer sort of thing where the two systems interoperate between themselves sharing information (which may be state and so can change processes in one another).
- To test end-to-end functionality between (at least) two communicating systems is as required by the standard(s) on which those systems are based is called **Interoperability Testing**.



# Security Testing

- Security is set of measures to protect an application against unforeseen actions that cause it to stop functioning or being exploited. Unforeseen actions can be either intentional or unintentional.
- Security Testing ensures, that system and applications in an organization, are free from any loopholes and weaknesses that may cause a big loss.

# Security Testing



Vulnerability Scanning

Security Scanning

Penetration testing

Risk Assessment

Security Auditing

Posture Assessment

Ethical hacking

# Test Case Development

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
TU01	Check Customer Login with valid Data	1.Go to site <a href="http://chocolateshop.com">http://chocolateshop.com</a> 2.Enter UserId 3.Enter Password 4.Click Submit	Userid = shreyata Password = shreyata	User should Login into application	As Expected	Pass

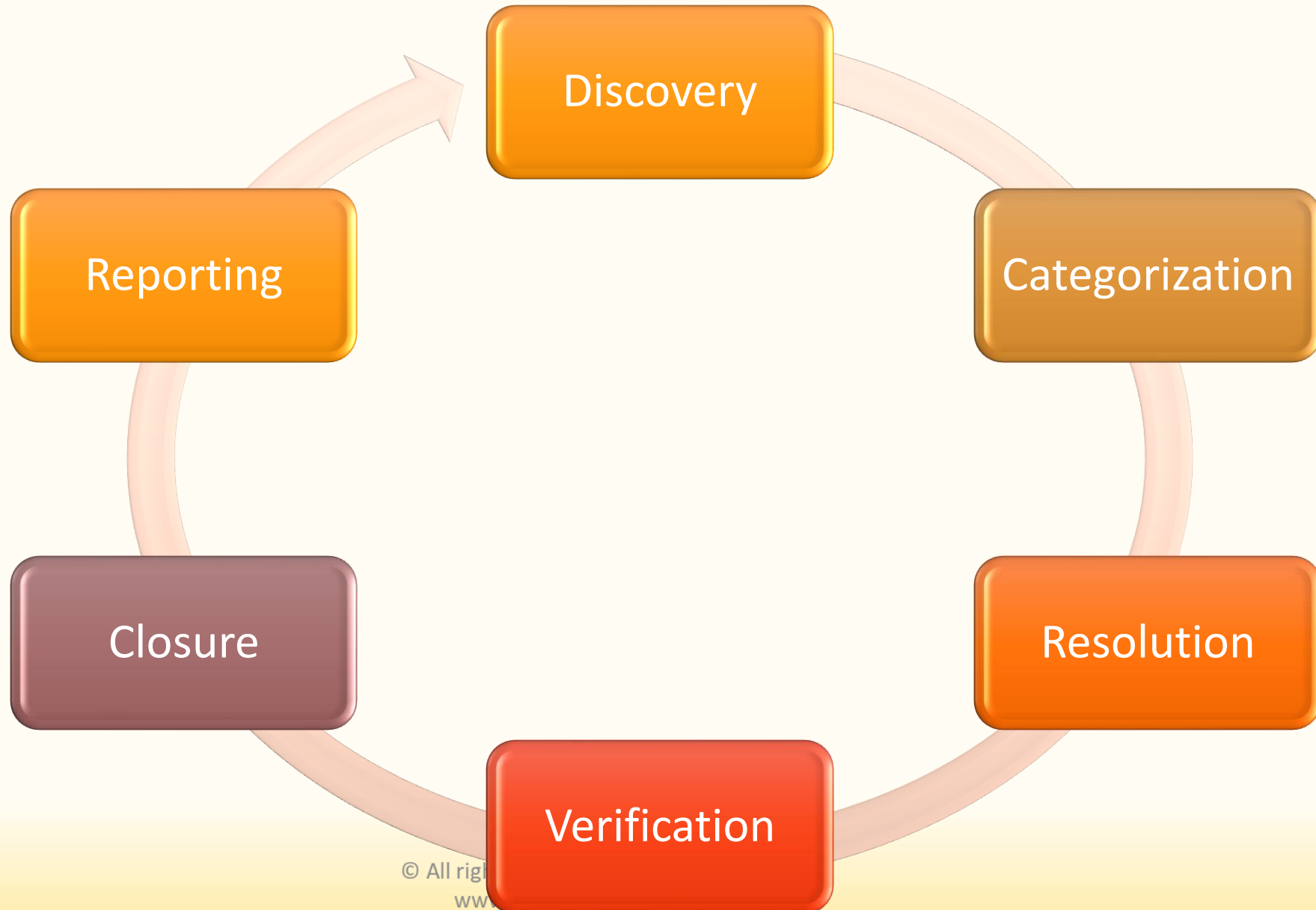
Popular Test Case Management Tools are – HP Quality Center, IBM Rational Quality Manager, Jira – Xray, Test Rail,



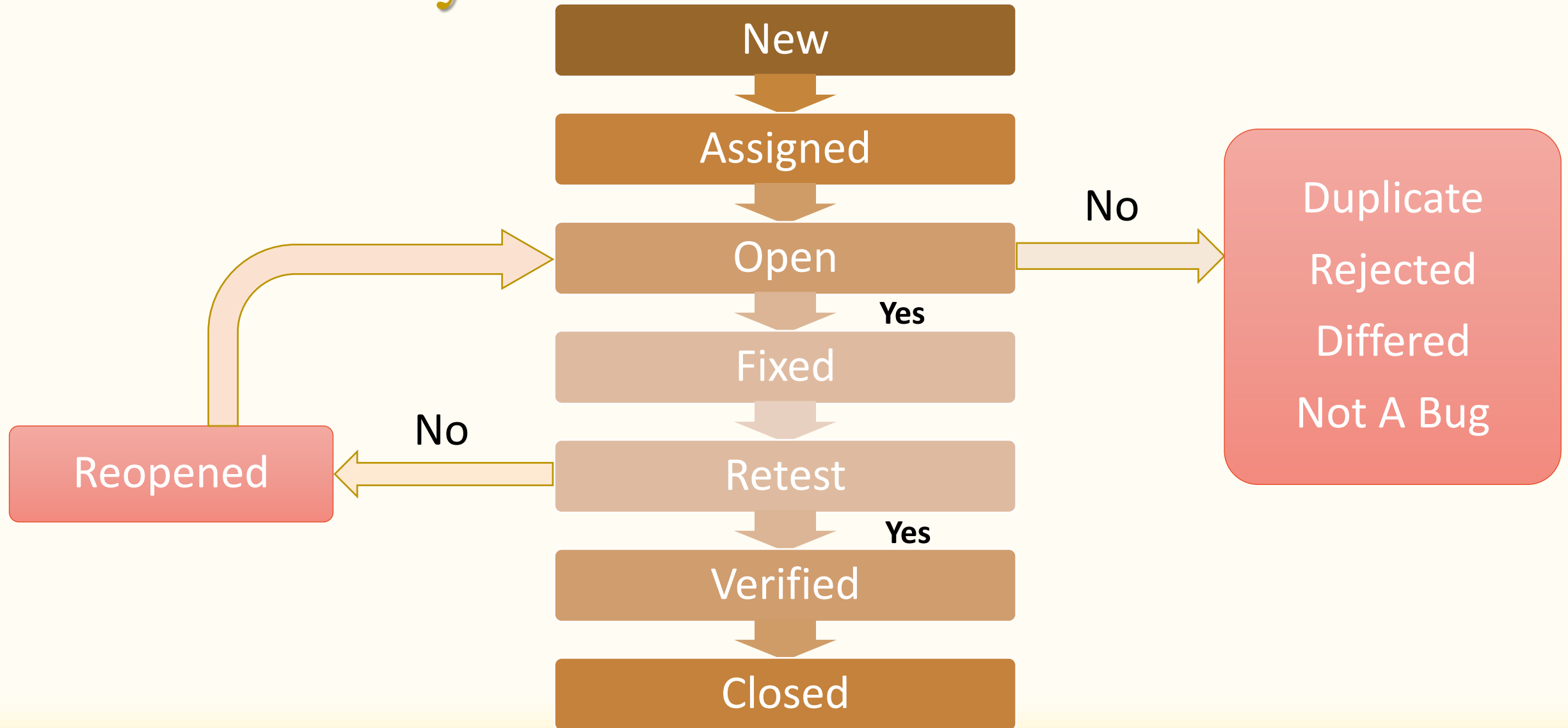
# Test Scenario Vs Test Condition

Test Scenario	Test Condition
<ul style="list-style-type: none"><li>•Test scenario is a possible ways to test an application. <b>Example:</b> For testing you have so many ways like positive testing, negative testing, BVA etc.</li></ul>	<ul style="list-style-type: none"><li>•Test condition is the constraint that you should follow to test an application. <b>Example:</b> When User Name and Password are valid then application will move forward</li></ul>
<ul style="list-style-type: none"><li>•Test scenario can be a single or a group of test cases</li></ul>	<ul style="list-style-type: none"><li>•Test condition can be a piece of functionality or anything you want to verify. In simple terms the goal of a test cases</li></ul>
<ul style="list-style-type: none"><li>•It is important when time is less and most team members understand the details from one line scenario</li></ul>	<ul style="list-style-type: none"><li>•It is an item or event of a system that could be verified by one or more test cases. Eg; transaction, function, structural element etc.</li></ul>
<ul style="list-style-type: none"><li>•Good test coverage can be achieved by dividing application in test scenarios which reduces the complexity</li></ul>	<ul style="list-style-type: none"><li>•Good Test Condition ensure system is bug free</li></ul>
<ul style="list-style-type: none"><li>•Test scenario are rather vague and covers wide range of possibilities</li></ul>	<ul style="list-style-type: none"><li>•Test condition are very specific</li></ul>

# Defect Management Process



# Defect Life Cycle



# Severity

- The degree of impact a defect has on the development or operation of a component application being tested is called Severity
- Higher effect on the system functionality will lead to the assignment of higher severity to the bug

# Severity

## Critical

- This defect indicates complete shut-down of the process, nothing can proceed further

## Major

- It is a highly severe defect and collapse the system. However, certain parts of the system remain functional

## Medium

- It cause some undesirable behavior, but the system is still functional

## Low

- It won't cause any major break-down of the system

# Priority

*Critical*

- Defects that are need to be fixed immediately, because it may cause grate damage to the product

*High*

- The defect impacts the product's main feature

*Medium*

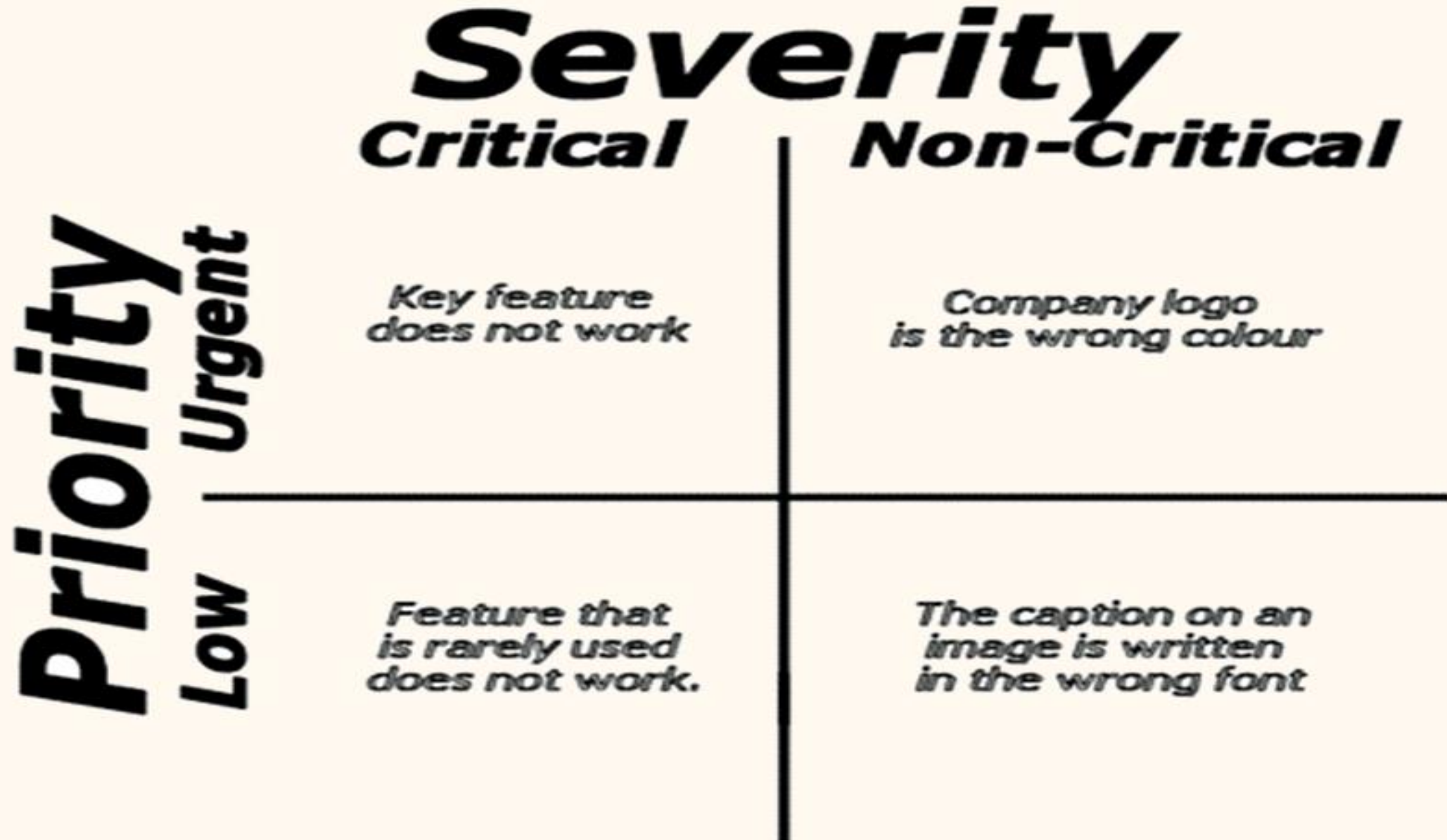
- The defect cause minimal deviation from the product requirement

*Low*

- The defect has very minor affect on the product operation

# Priority

No	Description	Priority	Explanation
1	The website performance is too slow	High	The performance bug can cause huge inconvenience to user.
2	The login function of the website does not work properly	Critical	Login is one of the main function of the banking website if this feature does not work, it is serious bugs
3	The GUI of the website does not display correctly on mobile devices	Medium	The defect affects the user who use Smartphone to view the website.
4	The website could not remember the user login session	High	This is a serious issue since the user will be able to login but not be able to perform any further transactions
5	Some links doesn't work	Low	This is an easy fix for development guys and the user can still access the site without these links





# Examples of Severity & Priority

- **High Priority & High Severity:** An error which occurs on the basic functionality of the application and will not allow the user to use the system. (Eg. 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.)
- **High Priority & Low Severity:** The spelling mistakes that happens on the cover page or heading or title of an application.
- **High Severity & Low Priority:** An error which occurs on the functionality of the application (for which there is no workaround) and will not allow the user to use the system but on click of link which is rarely used by the end user.
- **Low Priority and Low Severity:** Any cosmetic or spelling issues which is within a paragraph or in the report (Not on cover page, heading, title).

# Defect Reports

Defect report contains

- **Defect\_ID** - Unique identification number for the defect.
- **Defect Description** - Detailed description of the defect including information about the module in which defect was found.
- **Version** - Version of the application in which defect was found.
- **Steps** - Detailed steps along with screenshots & logs with which the developer can reproduce the defects.
- **Date Raised** - Date when the defect is raised
- **Reference**- where in you Provide reference to the documents like . requirements, design, architecture or may be even screenshots of the error to help understand the defect

# Defect Reports

- **Detected By** - Name/ID of the tester who raised the defect
- **Status** - Status of the defect
- **Fixed by** - Name/ID of the developer who fixed it
- **Date Closed** - Date when the defect is closed
- **Severity** - which describes the impact of the defect on the application
- **Priority** - which is related to defect fixing urgency. Severity Priority could be Critical/High/Medium/Low based on the impact urgency at which the defect should be fixed respectively

# Test Plan

- Test plan is the project plan for the testing work to be done.
- Test planning, the most important activity to ensure that there is initially a list of tasks and milestones in a baseline plan to track the progress of the project.
- It also defines the size of the test effort.
- The main document often called as master test plan or a project test plan and usually developed during the early phase of the project.

# Test Plan Parameters

Sr. No.	Parameter	Description
1.	Test plan identifier	Unique identifying reference.
2.	Introduction	A brief introduction about the project and to the document.
3.	Test items	A test item is a software item that is the application under test.
4.	Features to be tested	A feature that needs to be tested on the testware.
5.	Features not to be tested	Identify the features and the reasons for not including as part of testing.
6.	Approach	Details about the overall approach to testing.
7.	Item pass/fail criteria	Documented whether a software item has passed or failed its test.
8.	Test deliverables	The deliverables that are delivered as part of the testing process, such as test plans, test specifications and test summary reports.

# Test Plan Parameters

Sr. No.	Parameter	Description
9.	Testing tasks	All tasks for planning and executing the testing.
10.	Environmental needs	Defining the environmental requirements such as hardware, software, OS, network configurations, tools required.
11.	Responsibilities	Lists the roles and responsibilities of the team members.
12.	Staffing and training needs	Captures the actual staffing requirements and any specific skills and training requirements.
13.	Schedule	States the important project delivery dates and key milestones.
14.	Risks and Mitigation	High-level project risks and assumptions and a mitigating plan for each identified risk.
15.	Approvals	Captures all approvers of the document, their titles and the sign off date.

# Test Planning Activities

- To determine the **scope** and the **risks** that need to be tested and that are NOT to be tested.
- Documenting Test **Strategy**.
- Making sure that the testing **activities** have been included.
- Deciding **Entry** and **Exit** criteria.
- Evaluating the test **estimate**.
- Planning **when** and **how** to test and deciding how the **test results** will be evaluated, and defining test exit criterion.
- The Test **artefacts** delivered as part of **test execution**.
- Defining the **management information**, including the metrics required and defect resolution and risk issues.
- Ensuring that the test documentation **generates repeatable test assets**.

# Steps in Test Planning

