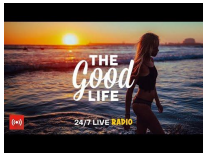
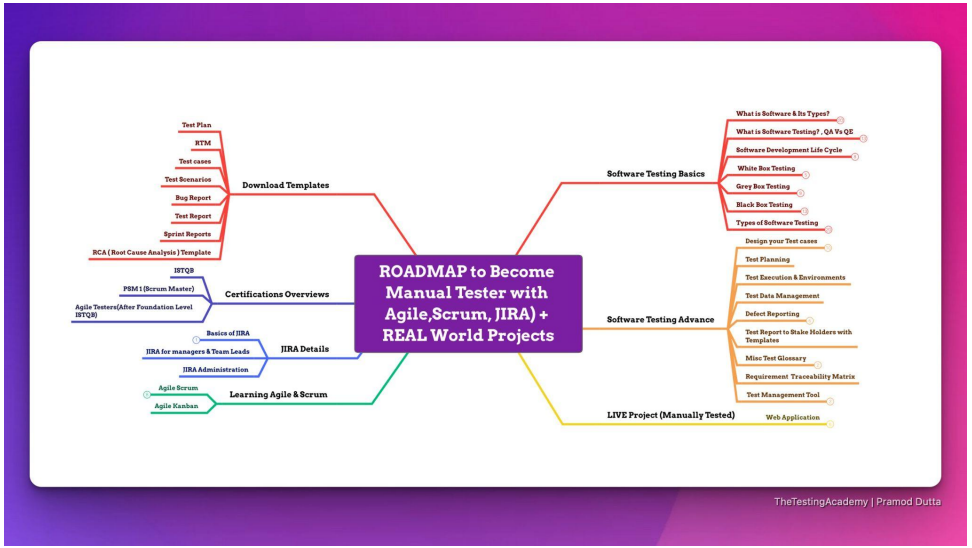
# Meeting Etiquette

- Please be on Mute all the time.
- Turn off Video, So that we can save bandwidth.
- There is separate Q&A Section in End Please use that, Add questions in Google Form.
- Break at 5 min.
- Mute your microphone
- To help keep background noise to a minimum, make sure you mute your microphone when you are not speaking.
- Be mindful of background noise
- Avoid multitasking
- All links and Slides will be shared.



# { Software Tester } BluePrint.

Pramod Dutta




## Exact BluePrint.

## You Need to Become Software Tester.

# Agenda

- Introduction to Software Testing?
- Details on SDLC and Different Models.
- STLC Life Cycle & 7 Principles of Software Testing.
- RTM & Different Types of Software Testing.
- Test Design Techniques
- Bugs, Severity vs Priority

# Rules

Focus on One Thing.

5% : 95% Rule

70% is Perfect
100% is Failure

New Action

# Commitment!
# Block at least
# 3-4 hour Per Week.

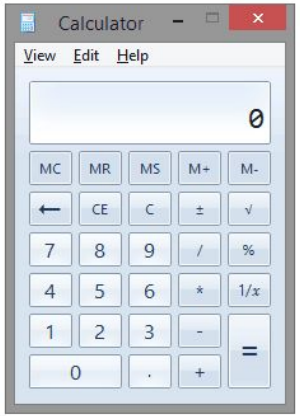
Laptop - 8 GB Ram,
512 GB HDD (SDD)

Mobile - Yes
Excel, PDF
Resume Building, LinkedIn,
ChatGPT(How to get Job)

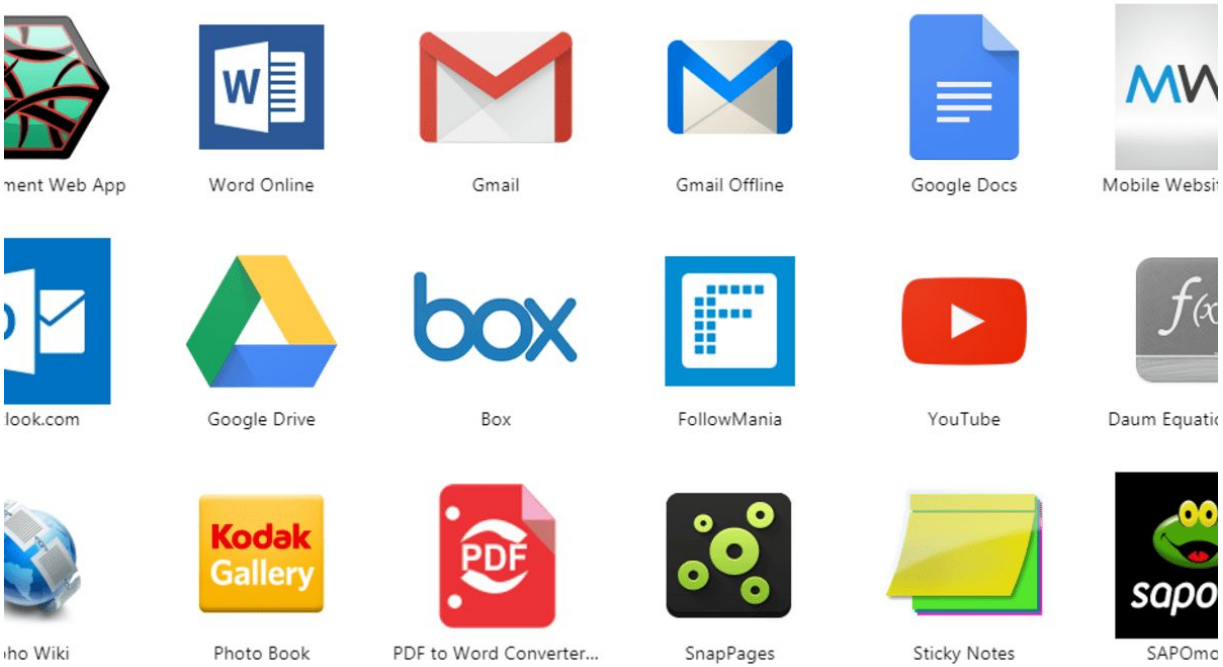
# What is Software?

Software is basically a set of instructions or commands that tells a computer what to do

Windows Calculator



# What is Software?



# Types of Softwares

- System software

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

- Programming software

Ex: compilers, debuggers, interpreters, etc.

- Application software

Ex: Web Applications, Mobile Apps, Desktop Applications etc

# Types of Software

**System Software**

System software basically controls a computer's internal functioning and also controls hardware devices such as monitors, printers, and storage devices, etc

Operating System
Language Processor
Device Driver
Written in a low-level language

**Driver software. ...**
**Middleware. ...**
**Programming software.**

**Application Software**

Application software is designed to perform a specific task for end-users

It is a product or a program that is designed only to fulfill end-users' requirements

General Purpose Software
MS-Word, MS-Excel, PowerPoint, etc.
Customized Software
airline reservation system
Utility Software
antivirus, disk fragmenter, memory tester, disk repair, disk cleaners
Written in a high-level language

# What is Software Testing

Software testing is the process of evaluating and verifying that a software product or application does what it is supposed to do.

Expect Result ( Requirement)
= Actual Result ( By doing it / QA)

Software Testing is a part of software development process.

Main Objective of testing is to release quality product to the client.

Activity to detect and identify the defects in the Software. (Not Fix)

# What are software testing objectives and purpose?

# What are software testing objectives and purpose?

To prevent defects.

**Finding defects**

Gaining confidence in and providing information about the level of quality

ensure that it satisfies the BRS that is Business Requirement

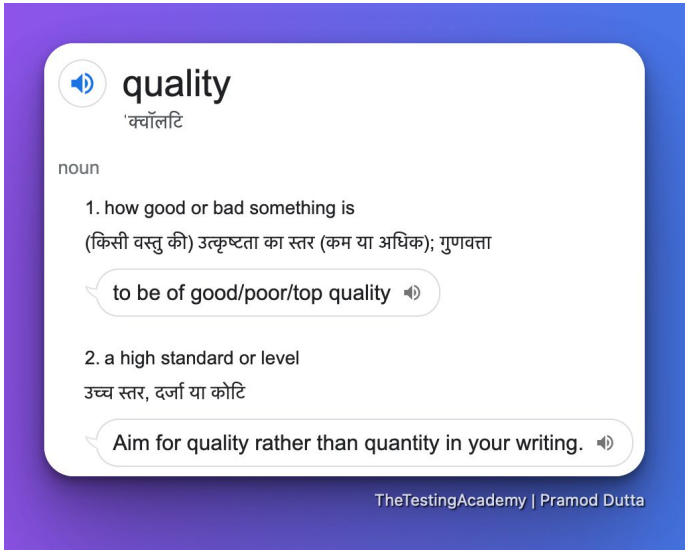
**end result meets t**he business and user requirements.

**gain the confidence of the customers by providing them a quality product.**


Prevent Defects
1
Evaluate Work Product
2
Verify Requirements
3
Validate Test Objects
4
5
Build Confidence
6
Reduce Risk
7
Share Information To Stakeholders
8
Find Failure & Defects

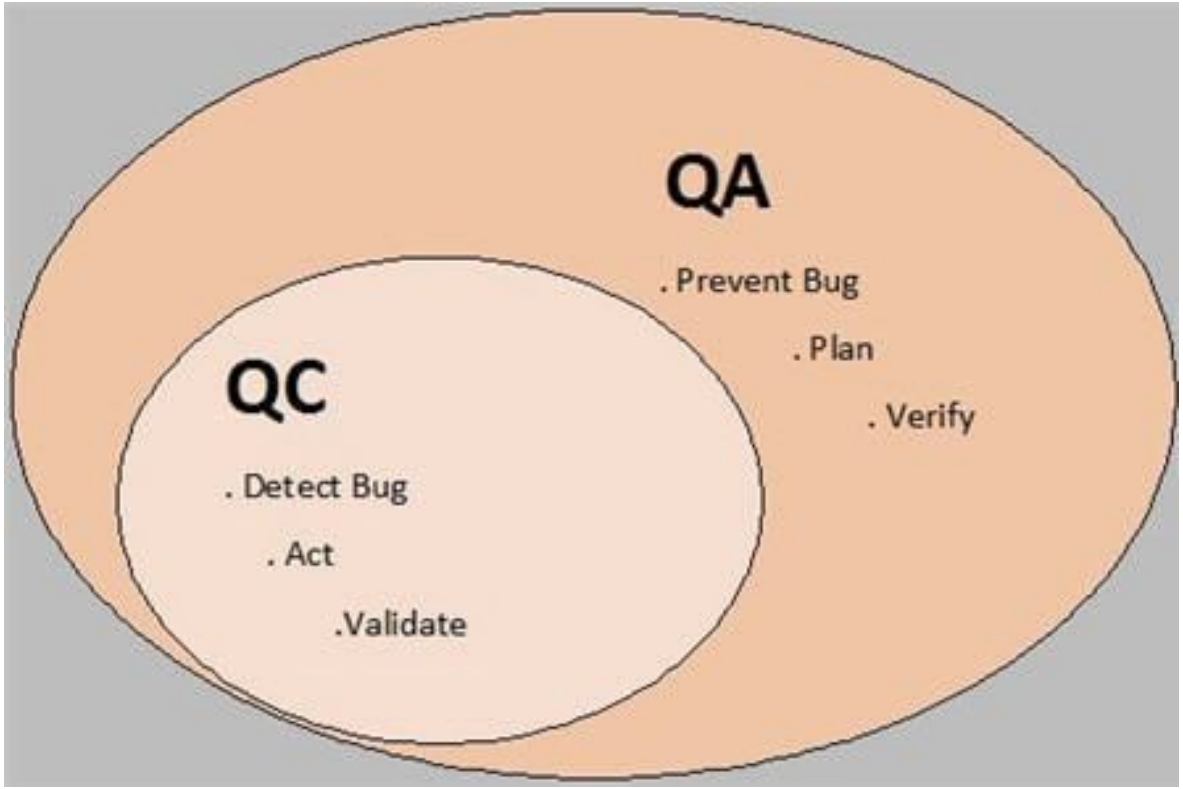
# Quality

**Quality is defined as justification of all the requirements of a customer in a product.**



**Quality software is Means**

- Bug-free
- Delivered on time.
- Within budget.
- Meets requirements and/or expectations.
- Maintainable

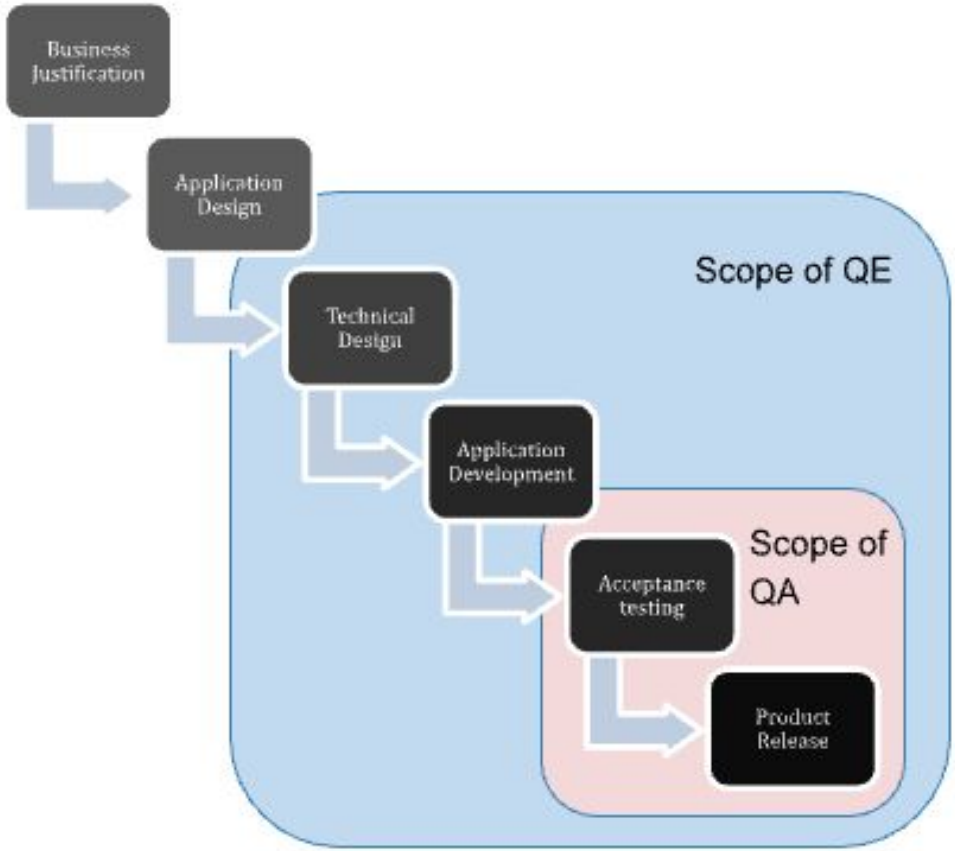

QA
. Prevent Bug
. Plan
. Verify
QC
. Detect Bug
. Act
.Validate

| Quality Assurance (QA) | Quality Engineer (QE) | Software Development Engineer in Test (SDET) |
|---|---|---|
| QA ought to be familiar with the bug tracking, ticketing, and testing processes. | QE must be familiar with operations as infrastructure, servers, platforms, etc. | SDET must be able to do advanced automated tasks. |
| Software engineering technical expertise, such as SQL overload or basic programming, is required by QA. | Security testing, performance testing, and integrating checks in a CI/CD methodology are all skills that QE should have. | SDET ought to be able to perform white box testing. |
| Manual and automated testing in Selenium, Cucumber, SoapUI, JMeter, and other tools should be familiar to QA. | QE ought to be able to test automation at several levels, including API, UI, and protocol. | SDET must be well-versed in development. |
| QA must be able to ask the proper questions, listen carefully to replies, thoroughly explain issues, and perform well under pressure. | Selenium, Cucumber, SoapUI, JMeter, and other tools should be familiar to QE. | SDET should be able to create orchestration platforms. |
| | Quality should be a concern for QE. | |

# Quality Assurance vs Testing

# Comparison between QA, QC and Testing

| Quality Assurance | Quality Control | Testing |
| --- | --- | --- |
| Subset of SDLC | Subset of QA | Subset of QC |
| Process oriented | Product oriented | Product orinted |
| Ensure that processes and procedures are in place to achieve quality | Activities to ensure the product quality | Validate the product against specifications |
| Focus on process to achieve required quality | Focus on product to check for the required quality | Focus on actual testing of the product |
| Prevent defects | Find and fix defects | Find and fix defects |
| Whole team approach | Reactive process | Reactive process |
| Proactive process | Testing team | Testing team |


Business Justification
Application Design
Technical Design
Application Development
Acceptance testing
Product Release
Scope of QE
Scope of QA

# Software Testing

To check whether the **Actual** software product matches **Expected** requirements and to ensure that software product is Defect free.

| Id | Input | Expected Result | Actual Result | Status |
|---|---|---|---|---|
| 1 | Read ATM Card | Accept card and ask for pin # | Accepted the card and asked for a pin. | **Passed** |
| 2 | Read Invalid Card | "No ATM Card" exception is thrown and card is returned to the user. | Accepted the card and asked for a pin. | **Failed** |
| 3 | Invalid PIN Entered | "Stolen Card" exception is thrown and card is destroyed. | Accepted the card and asked for a pin. | **Failed** |

# Why Software Testing is Important?

## 4. Bitcoin Hack, Mt. Gox, 2011

Mt. Gox was the biggest bitcoin exchange in the world in the 2010s, until they were hit by a software error that ultimately proved fatal.

The glitch led to the exchange creating transactions that could never be fully redeemed, costing up to $1.5 million in lost bitcoins.

But Mt. Gox's woes didn't end there. In 2014, they lost more than 850,000 bitcoins (valued at roughly half a billion USD at the time) in a hacking incident. Around 200,000 bitcoins were recovered, but the financial loss was still overwhelming and the exchange ended up declaring bankruptcy.

https://raygun.com/blog/costly-software-errors-history/

# Why Software Testing is Important?

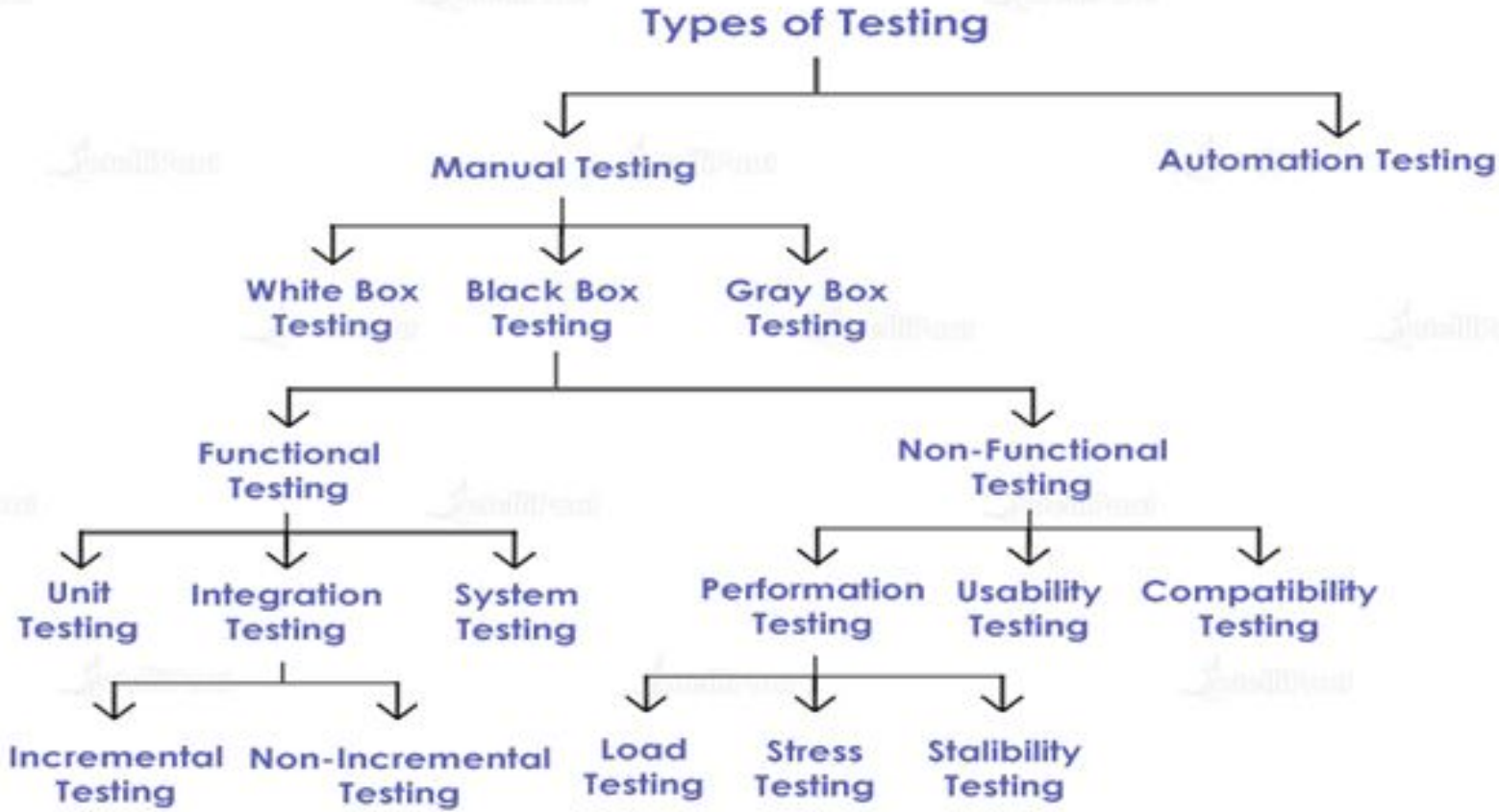
Vulnerability in Windows 10. This bug enables users to escape from security sandboxes through a flaw in the win32k system

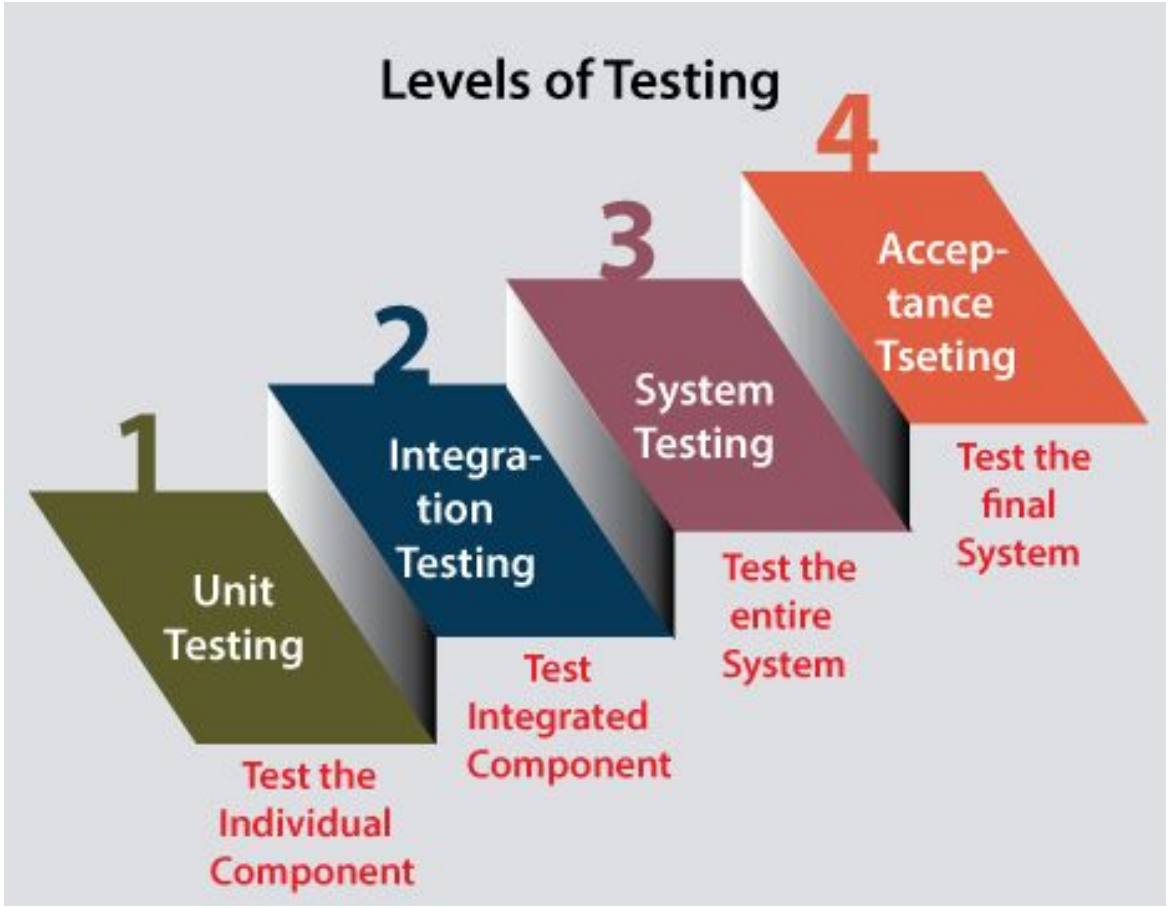
China Airlines Airbus A300 crashed due to a software bug on April 26, 1994, killing 264 innocents live

https://raygun.com/blog/costly-software-errors-history/

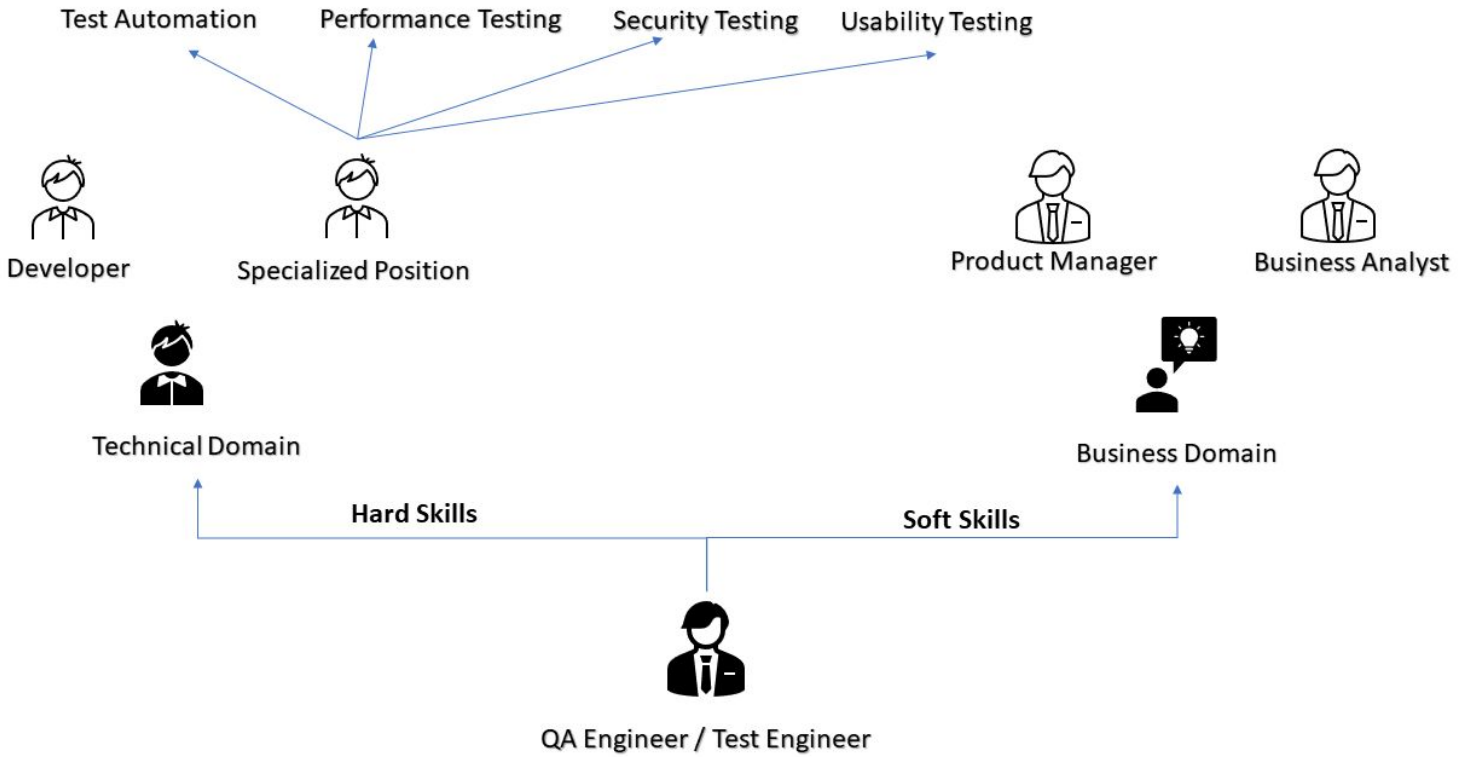
# Benefits of Software Testing

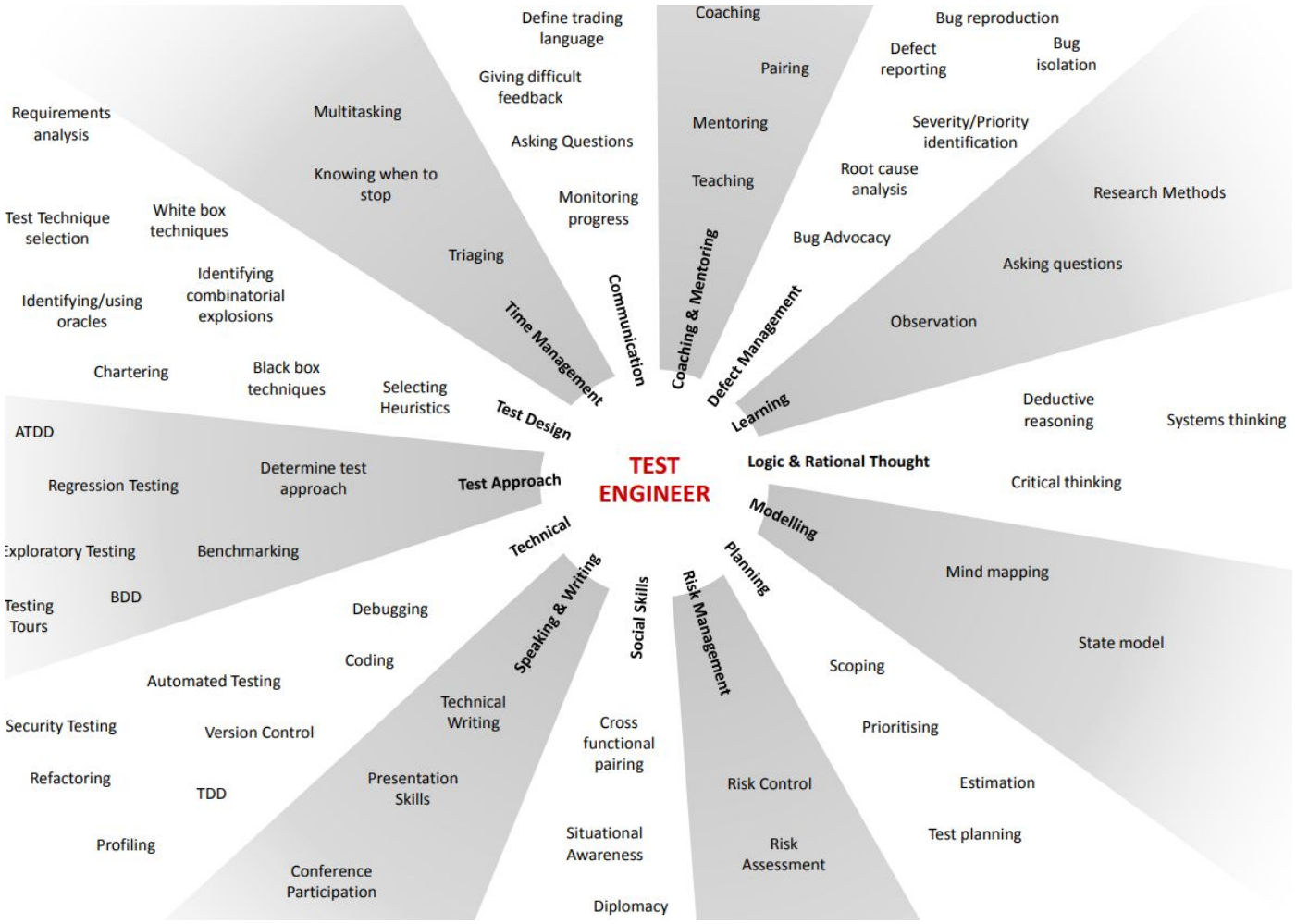
- Cost-Effective
- Product quality
- **Customer Satisfaction**


Types of Testing
Manual Testing
Automation Testing
White Box Testing
Black Box Testing
Gray Box Testing
Functional Testing
Non-Functional Testing
Unit Testing
Integration Testing
System Testing
Performation Testing
Usability Testing
Compatibility Testing
Incremental Testing
Non-Incremental Testing
Load Testing
Stress Testing
Stalibility Testing


Levels of Testing
1
Unit Testing
Test the Individual Component
2
Integra-tion Testing
Test Integrated Component
3
System Testing
Test the entire System
4
Accep-tance Tseting
Test the final System


Trainee Tester
Software Tester
QA Analyst
Software Test Engineer
QA Team Coordinator
Test Manager
Senior Test Manager


Test Automation
Performance Testing
Security Testing
Usability Testing
Developer
Specialized Position
Product Manager
Business Analyst
Technical Domain
Business Domain
Hard Skills
Soft Skills
QA Engineer / Test Engineer

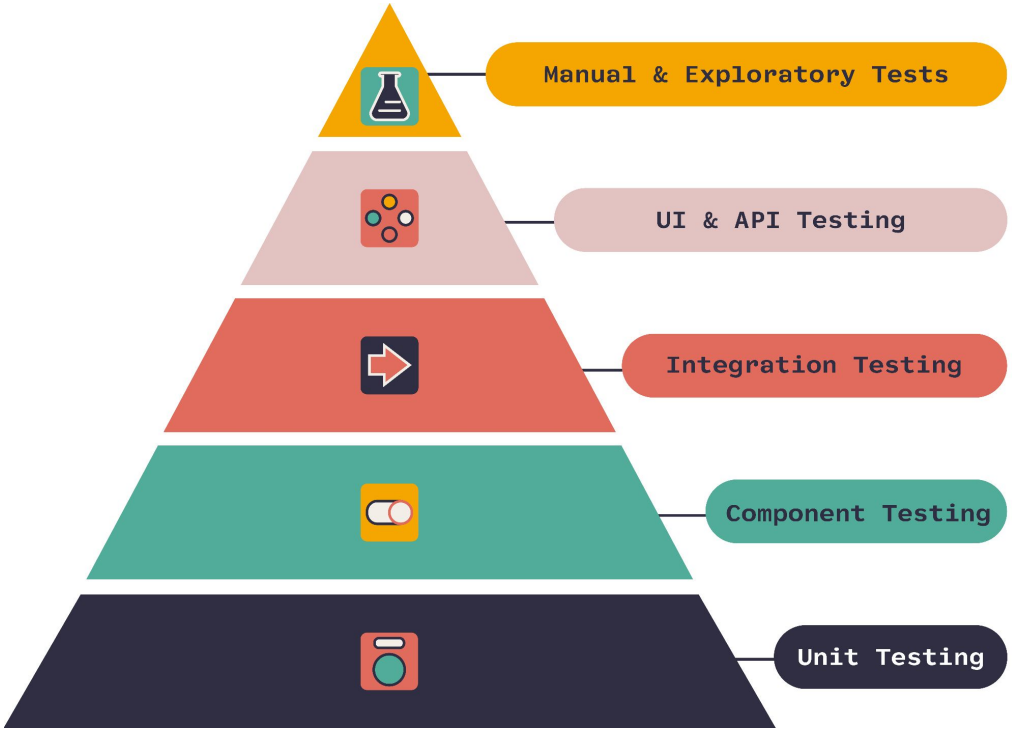

TEST ENGINEER
Time Management
Communication
Coaching & Mentoring
Defect Management
Learning
Test Design
Test Approach
Technical
Logic & Rational Thought
Modelling
Planning
Risk Management
Social Skills
Speaking & Writing
Define trading language
Giving difficult feedback
Asking Questions
Monitoring progress
Triaging
Multitasking
Knowing when to stop
Requirements analysis
White box techniques
Test Technique selection
Identifying combinatorial explosions
Identifying/using oracles
Chartering
Black box techniques
Selecting Heuristics
Coaching
Pairing
Mentoring
Teaching
Bug Advocacy
Bug reproduction
Defect reporting
Bug isolation
Severity/Priority identification
Root cause analysis
Research Methods
Asking questions
Observation
Deductive reasoning
Systems thinking
Critical thinking
ATDD
Regression Testing
Determine test approach
Exploratory Testing
Benchmarking
Testing Tours
BDD
Debugging
Coding
Automated Testing
Security Testing
Version Control
Refactoring
TDD
Profiling
Technical Writing
Presentation Skills
Conference Participation
Cross functional pairing
Situational Awareness
Diplomacy
Risk Control
Risk Assessment
Scoping
Prioritising
Estimation
Test planning
Mind mapping
State model

# Types of Software Testing

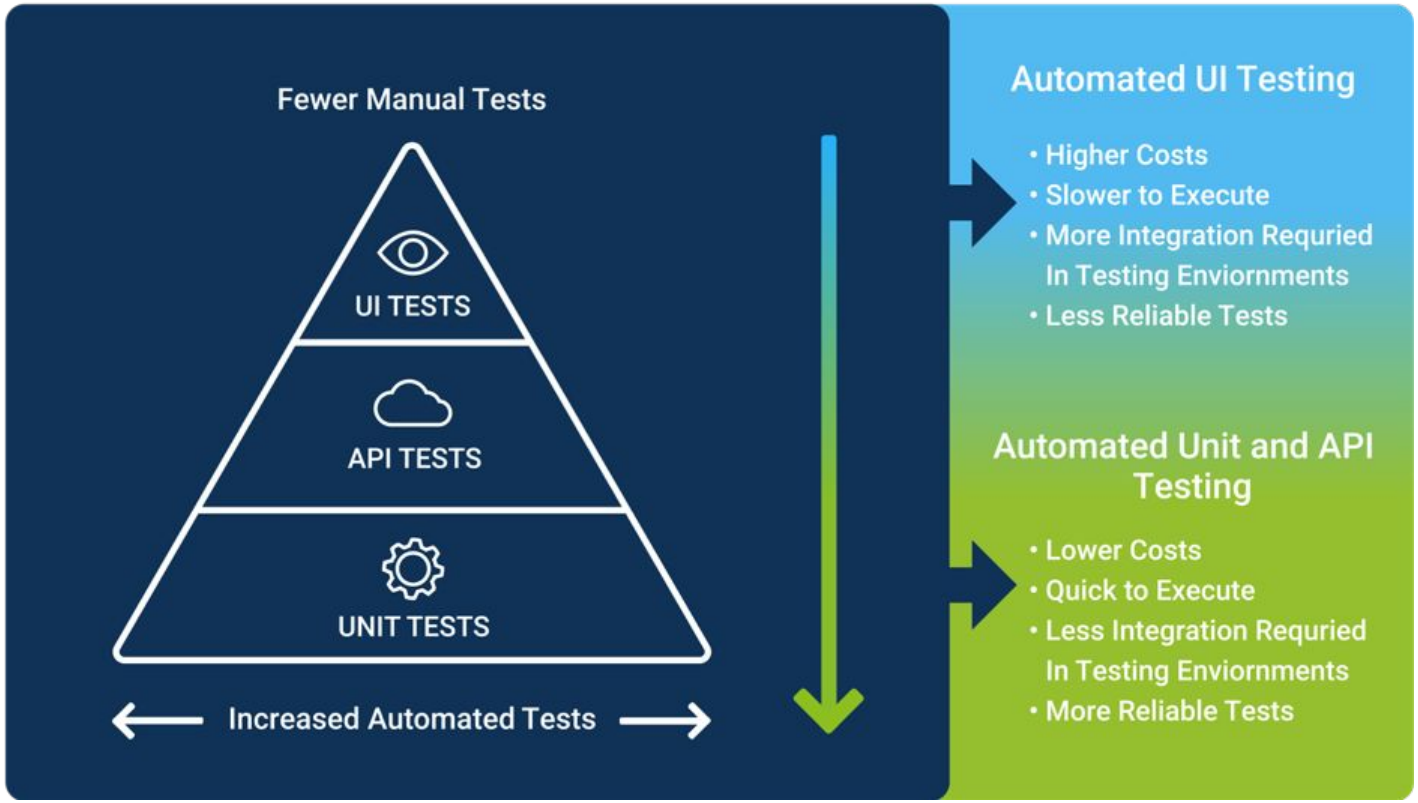
**Functional Testing**

Non-Functional Testing

# Testing Pyramid



# The Automation Pyramid



# Unit Testing

**Type of software testing** where **individual units** or components of a software are tested

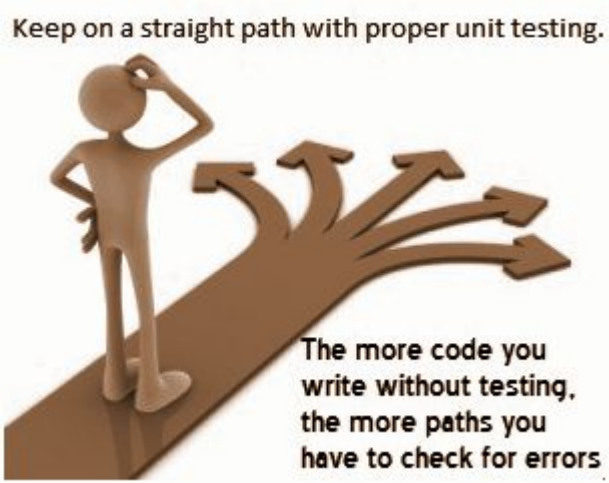
Unit tests help to fix bugs early in the development cycle and save costs.

**A developer writes a section of code in the application just to test the function. They would later comment out and finally remove the test code when the application is deployed.**

A coder generally uses a **UnitTest Framework** to develop automated test cases

# Code coverage techniques used in Unit Testing

- **Statement Coverage**
- Decision Coverage
- Branch Coverage
- Condition Coverage
- Finite State Machine Coverage



# Unit Testing

## Demo of Web App unit

```
public static boolean isSumPositive(int a, int b){
    // Statement Coverage Unit Example
    if(a+b > 0){
        return true;
    }else{
        return false;
    }
}
```

isSumPositive function

```
@Test
public void shouldAnswerWithPass()
{
    Assert.assertTrue(AddModule.isSumPositive( a: 2, b: 3));
}

@Test
public void shouldAnswerWithFail()
{
    Assert.assertTrue(AddModule.isSumPositive( a: 2, b: -3));
    //Assert.assertFalse(AddModule.isSumPostive(2,-3));
}
```

# Statement Coverage!

Statement coverage is a type of software testing that aims **to ensure that every statement in a program has been executed at least once during testing.** This means that every line of code is tested, but not necessarily every possible path through the code.

To achieve **100% statement coverage for this code**,

write a test case where a is greater than b, and another test case where b is greater than or equal to a.

This would ensure that both branches of the if statement are executed at least once, as well as the assignment and return statements.

```
int max(int a, int b) {
  int result;
  if (a > b) {
    result = a;
  } else {
    result = b;
  }
  return result;
}
```

# Decision Coverage

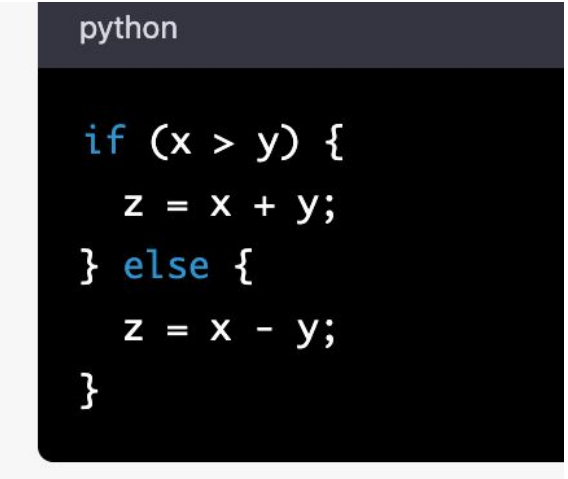
Ensuring that all possible outcomes of a decision or boolean expression have been evaluated

This means that each possible true or false outcome must be tested at least once.

To achieve 100% decision coverage,

we would need to test both the true and false outcomes of the boolean expression x > y. So,

we might write two test cases: one where x is greater than y, and one where x is not greater than y

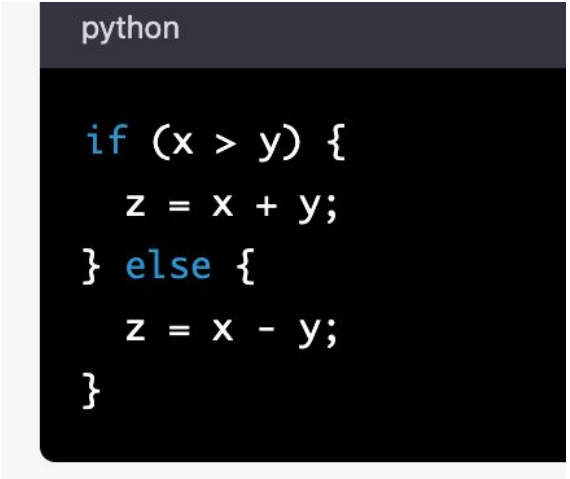


# Branch Coverage

Each possible path through a piece of code must be executed at least once.

we would need to test both the true and false outcomes of the boolean expression x > y,

as well as both branches of the if statement.



# Condition Coverage

Condition coverage is similar to decision coverage, but it focuses specifically on ensuring that each possible combination of boolean sub-expressions within a decision have been evaluated

we would need to test all possible combinations of the sub-expressions x > y and y > z.

This means we would need to test the true-true, true-false, false-true, and false-false combinations.

python

```python
if (x > y && y > z) {
  z = x + y;
} else {
  z = x - y;
}
```

# Finite State Machine Coverage

Finite state machine coverage is a testing technique that focuses on ensuring that all possible states and transitions within a finite state machine have been tested.

consider a vending machine that dispenses snacks:

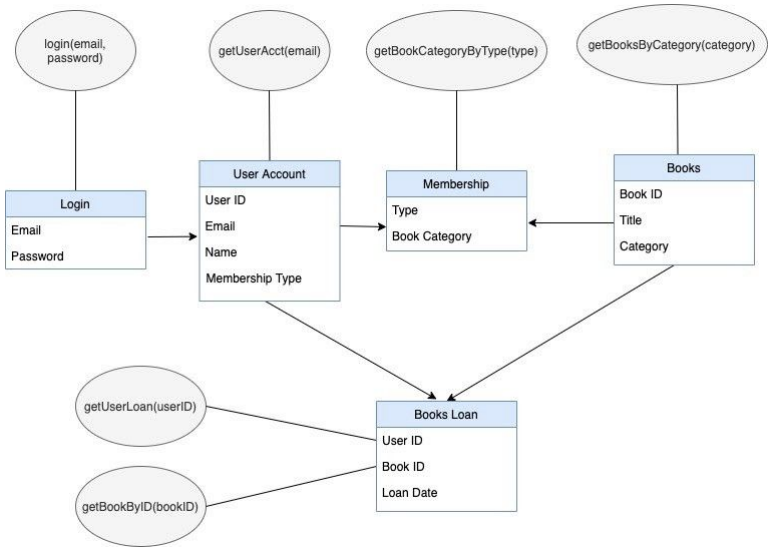
State 1: Idle
  -> Money inserted: transition to state 2
State 2: Selecting product
  -> Product selected: transition to state 3
  -> Money refunded: transition to state 1
State 3: Dispensing product
  -> Product dispensed: transition to state 1

To achieve 100% finite state machine coverage, we would need to test every possible combination of starting state and transitions within the machine.

# Integration Testing

Type of software testing where 2 or more unit combined.

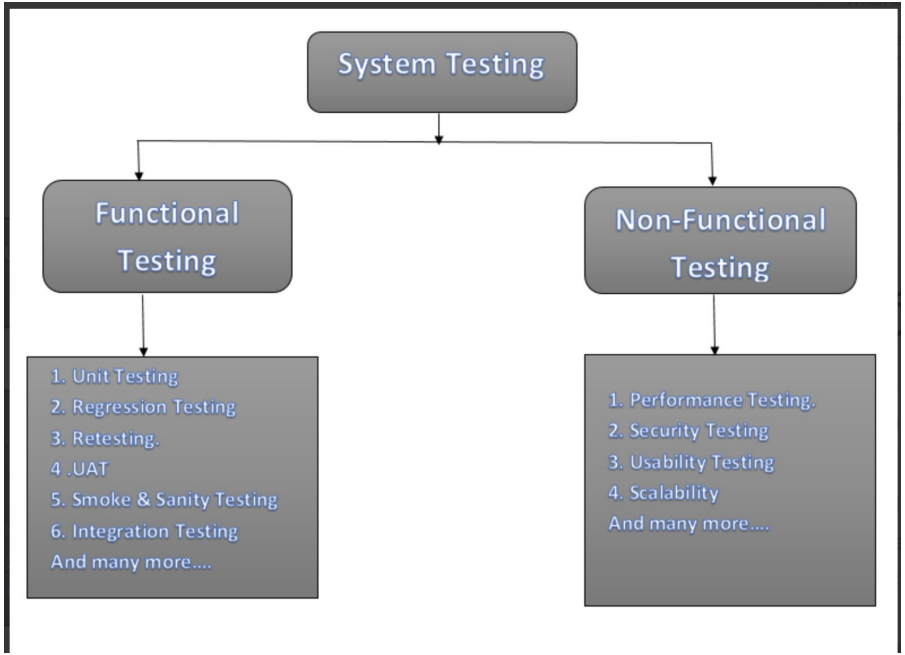
Software modules are integrated logically and tested as a group.




**Testing flow of data/information between the modules**.

# System Testing

System testing is a testing level in which tests are performed to know if a complete build aligns with functional and nonfunctional requirements made for it



# Integration Testing

The purpose of integration testing in the **context of an ecommerce site** would be to **ensure that individual modules or components work together as expected**.

For example, we might test the integration between **the product catalog module and the shopping cart module to ensure that items are correctly added to the cart and that inventory is updated.**

We might also test the integration between **the payment processing module and the order fulfillment module to ensure that payments are correctly processed and that orders are shipped out in a timely manner.**

The test environment would be a testing environment with simulated interactions, and the test data would be specific to each module or component being tested.

# System Testing

The purpose of system testing in the context of an ecommerce site would be to **verify that the entire system works as expected, from the user's perspective.**

This would involve **testing a range of user scenarios, such as browsing products, adding items to the cart, checking out, and making payments.**

The **test environment would be production-like, and the test data would be representative of the types of products, users, and transactions that the site is expected to handle**.

# Acceptance testing

Acceptance testing is usually the final stage of testing before the software is deployed, and **it is typically conducted by the end-users, stakeholders, or other representatives of the business who are responsible for ensuring that the software meets their needs.**

**User Acceptance Testing (UAT):** In this type of testing, **end-users or other stakeholders test the software in a real-world scenario to ensure that it meets their specific business requirements.** The purpose of UAT is to verify that the software is usable and meets the needs of the users.

**Business Acceptance Testing (BAT):** In this type of testing, representatives from the business side of the organization test the software to ensure that it meets the overall business goals and objectives. The purpose of BAT is to verify that the software aligns with the organization's strategic direction and that it will provide the intended value.

# System Testing vs Integration Testing

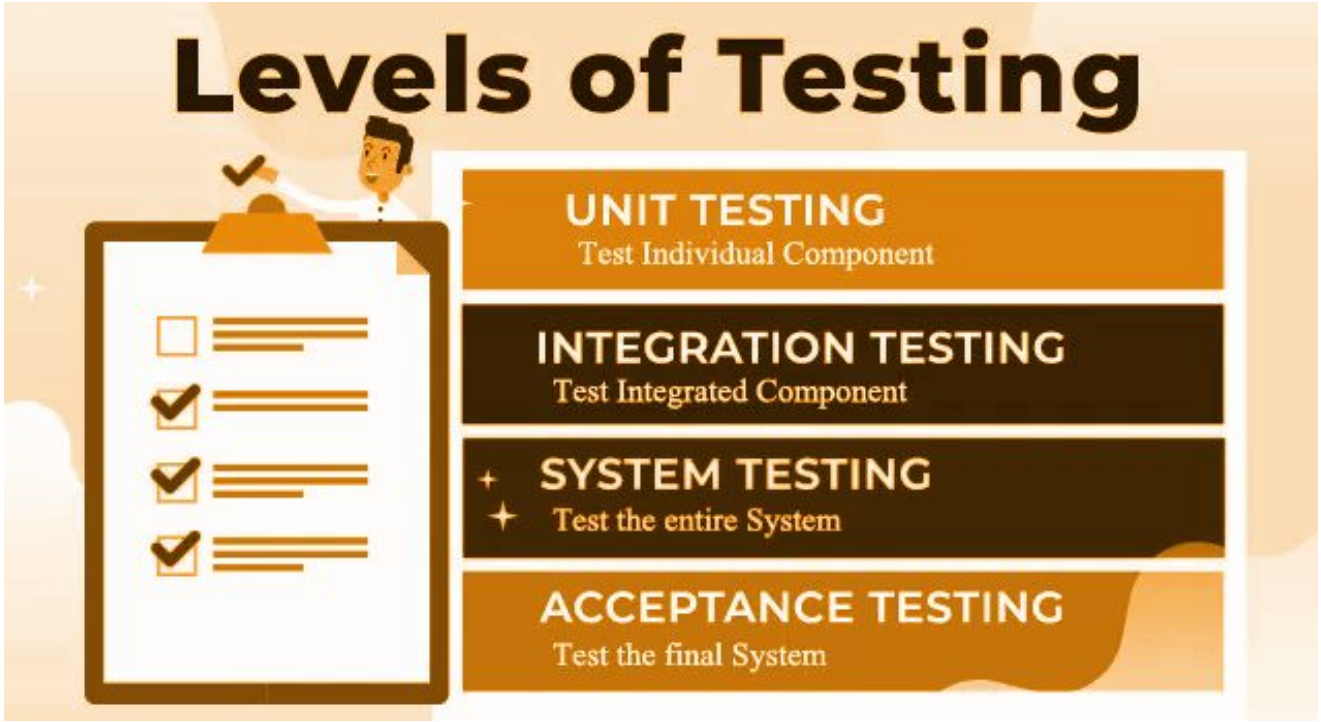
| | System Tests | Integration Tests |
|---|---|---|
| Intention | To guarantee that the total build fulfills the business specifications. | To guarantee that joined units can act together without problems. |
| Type | Nonfunctional and functional type of test. It falls in the acceptance testing class. | Functional type of test. It's not in the acceptance testing class. |
| Technique | Black box testing | White and black box testing or gray box testing |
| Level | Three (3) | Two (2) |
| Value | Helps to identify system errors. | Helps to identify interface errors. |
| Teams involved | Developers and Testers | QA |

# System Testing vs Integration Testing

| Criteria | System Testing | Integration Testing |
|---|---|---|
| Purpose | To ensure that the entire system or application works as expected | To ensure that individual modules or components work together as expected |
| Scope | End-to-end testing of the entire system | Testing of individual modules or components and their interactions |
| Test Environment | Production-like environment | Testing environment with simulated interactions |
| Test Data | Realistic data representative of production environment | Test data specific to each module or component |
| Test Strategy | Black-box testing approach, focusing on user scenarios | White-box testing approach, focusing on internal workings of modules or components |
| Examples | Testing the entire ecommerce site from the user's perspective, including browsing, adding to cart, checkout, and payment | Testing the integration between the product catalog module and the shopping cart module, or between the payment processing module and the order fulfillment module |

# System Testing vs E2E Testing

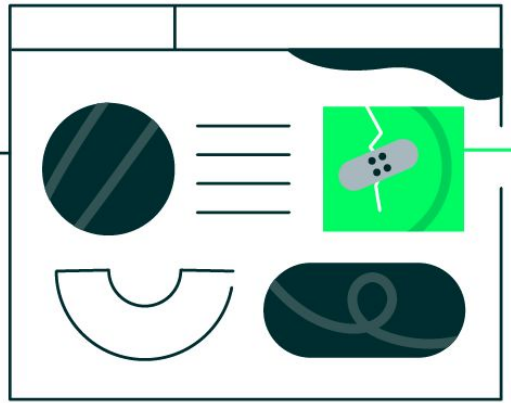
| System Testing | End-to-end Testing |
| --- | --- |
| It is carried out once integration testing is performed. | It is performed after the system testing. |


Levels of Testing
UNIT TESTING
Test Individual Component
INTEGRATION TESTING
Test Integrated Component
SYSTEM TESTING
Test the entire System
ACCEPTANCE TESTING
Test the final System

# Regression testing vs. retesting

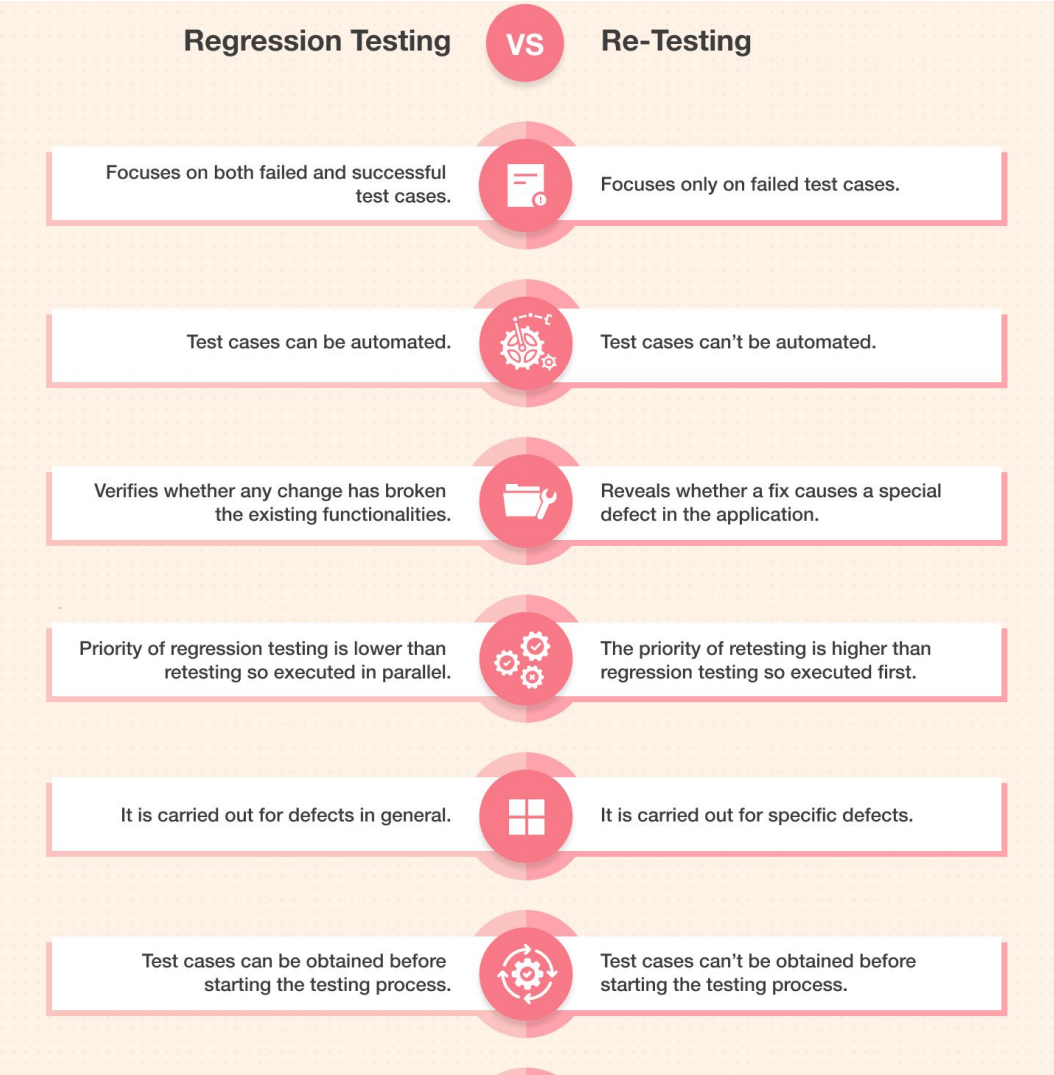
**Regression testing**

Testing code which was working, but now, due to updates, might not be



**Retesting**

Testing code which was not working, but which you believe to have been fixed



# Manual Tester / Software Tester

## Roles and Responsibilities

# Manual Tester / Software Tester

## Roles and Responsibilities

- **Gather requirements from team**
- Prepare Test Plan, Test Scenarios, Test cases - https://sdet.live/3DbA
- Verifying the Software Web/ App by Hand
- Execute the Test cases and Report Bugs
- Send Test report to stakeholders
- More concentrated on the UI/UX issues
- In charge of paperwork(aka documentation online)

# Manual Tester Roles and Responsibilities

Test environment setup
Participation in meetings
Analysis of customer requests
Software bug tracking
Analysis and execution of test cases
Maintaining contact with test managers

# What you do can extra? As Manual tester

- Help in preparing the requirements to PM
- Share a Video or Images of the Manual flows to automation team, so that they can create better automation.
- Help the team pm, devs to sync and come with timelines of release.
- Learn coding and help in automation of the flows manual tested.
- Identify the automation flows and pain manual areas automate them using scripts

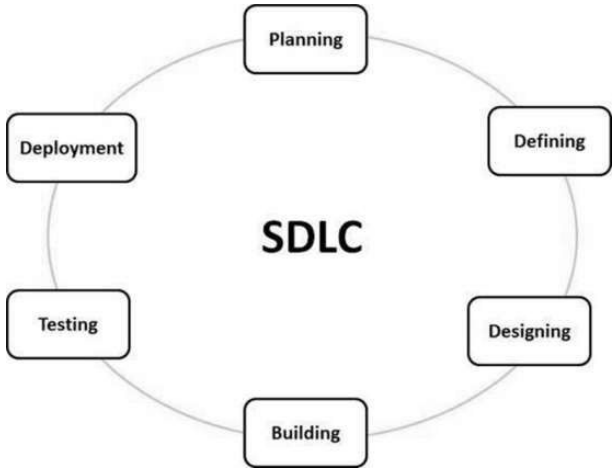
# What you do can extra? As Manual tester

https://forms.gle/KcCe2bhZSFsS5GzLA

# Software Development Life Cycle

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares.

ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.



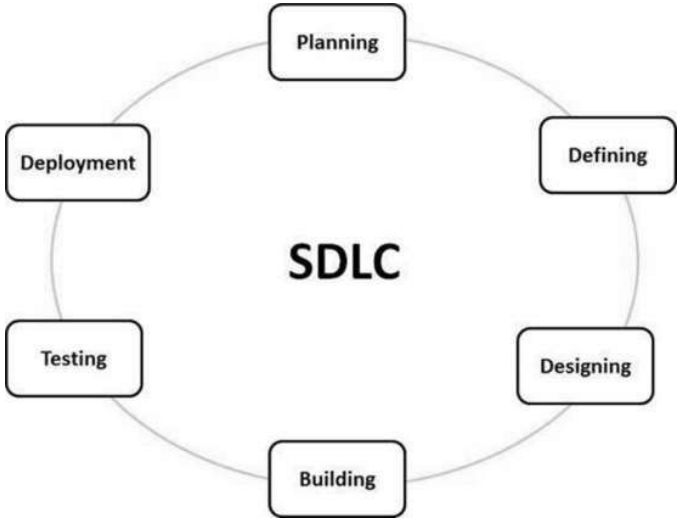
# 1. Planning and Requirement Analysis

- It is performed by the senior members of the team with inputs from the customer.
- Sales department, market surveys and domain experts in the industry.

Documents - Statement of Work, Project Plan

Outcome - Various Technical approaches that can be followed to implement the project successfully with minimum risks.

https://bugz.atlassian.net/l/cp/EeXpfJ0W



# SOW-TEMPLATE-Project Manager-ND

https://docs.google.com/document/d/1oP7Fw3RTvCdYuc1sH_1MeRYzOUhmtapr/edit?usp=sharing&ouid=104755920778477387077&rtpof=true&sd=true

# 2. Defining Requirements

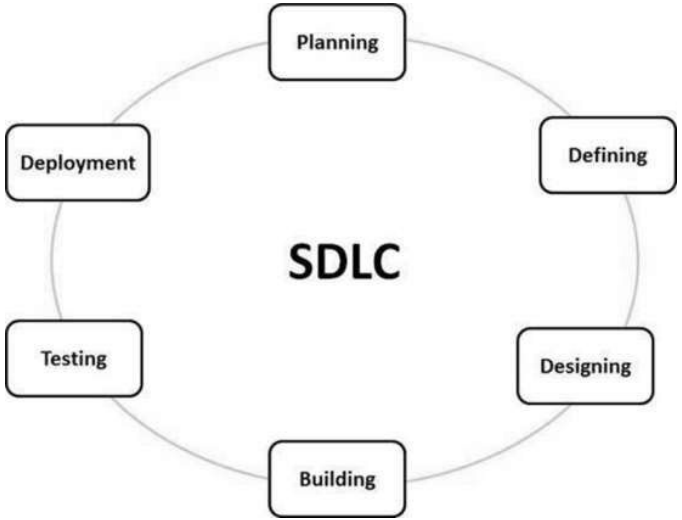
- Define and document the product requirements and get them approved from the customer or the market analysts

Documents - SRS (Software Requirement Specification) document

https://sdet.live/samplesrs

Consists of all the product requirements to be designed and developed during the project life cycle.

https://www.geeksforgeeks.org/software-requirement-specification-srs-format/



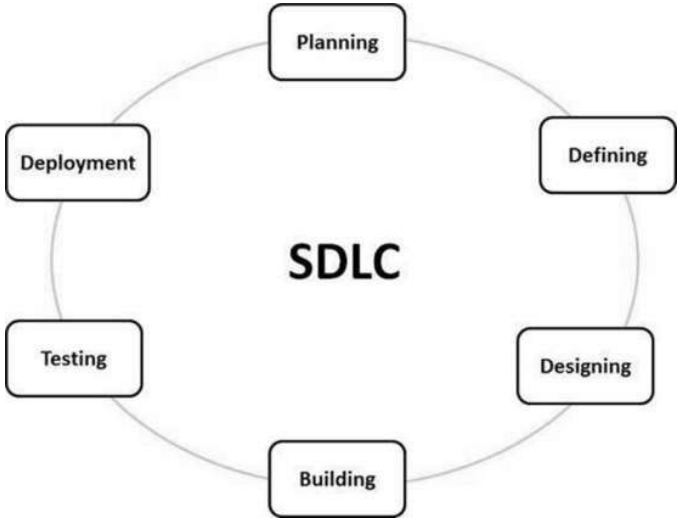
# 3. Designing

- Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification

Documents - DDS

https://sdet.live/samplesrs

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation



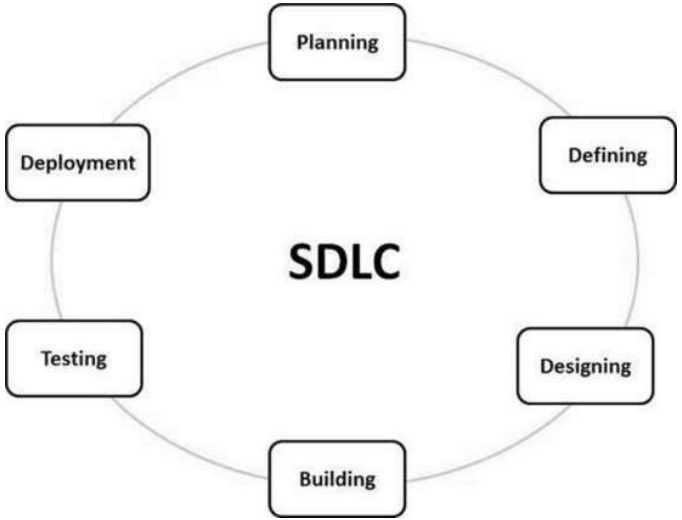
# 4. Building

- The programming code is generated as per DDS during this stage

Documents - FRDs

https://sdet.live/samplesrs

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers



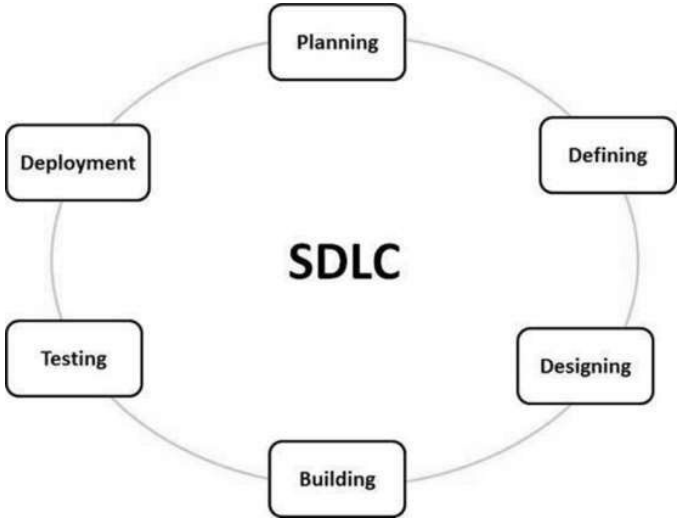
# 5. Testing

- This stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Documents - multiple Docs

https://sdet.live/notes

Full STLC Life Cycle

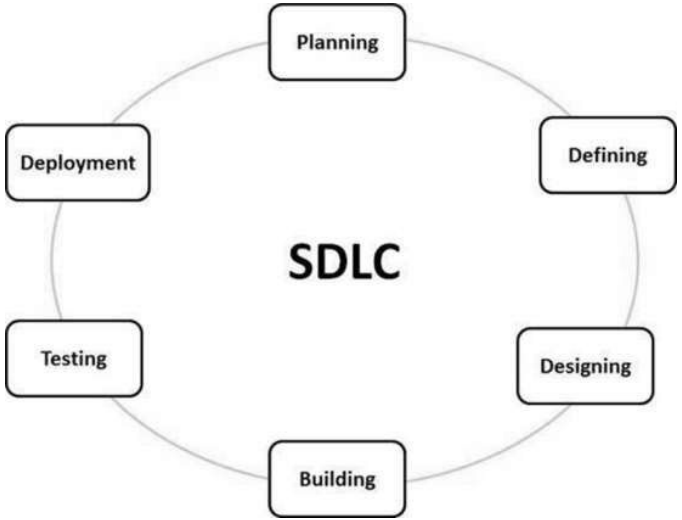


# 6. Deployment

- Once the product is tested and ready to be deployed it is released formally in the appropriate market.

Documents - NA

The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).



# BRD

**Business Requirement Document i.e. BRD** is created during the initial phase of the project. This document contains the high-level business requirements that could be easy to follow by business stakeholders, managers, board of directors, etc.

a business requirement document is a functional business requirement that is written in a well-structured manner without any technical jargon.

- Bank customers should be able to register themselves.
- Registered bank customers should be able to log in.
- Customers should be able to do online transactions.
- Customers should be able to open fixed deposit online.
- Customers should be able to recharge their mobile phones, etc.

# FRD

**Functional specification document** is required with detailed requirements in technical terms that will be referred by the technical team for further development of the system.

- Username should not include numeric value.
- The password should be 8 or more char long.
- Only 5 transactions should be allowed in a day, etc

# SRS

**Software Requirement Specification** i.e. SRS Document is one of the important documents for the development team. It is a **complete description of the behavior of a system to be developed.**

SRS document contains all the functional and non-functional requirements along with the use cases that the software must meet.

Software specification requirement document elaborates on the **business requirements mentioned in BRD** to accommodate the functional and non-functional requirements along with user intersections with the system i.e. use cases.

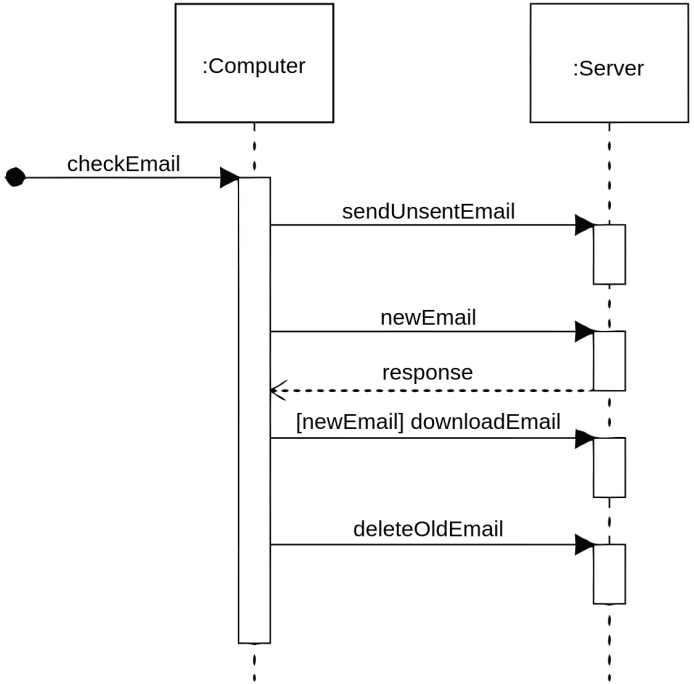
document is prepared by system analysts or business analysts and referred by the project management, development, and implementation teams.

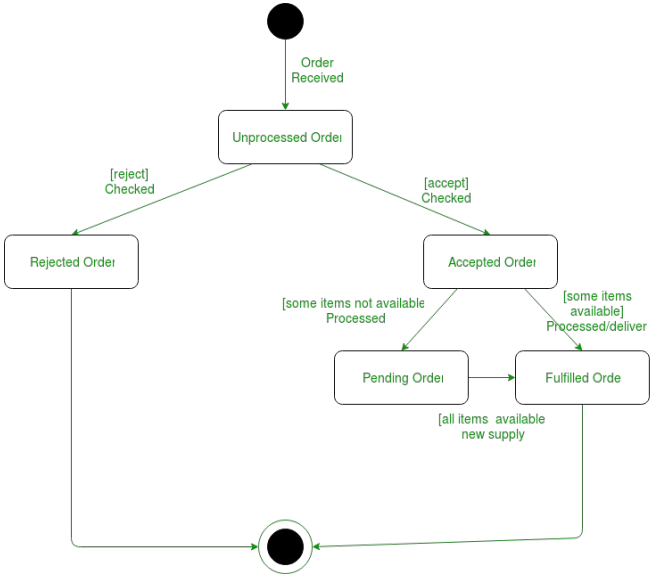
UML **(Unified Modeling Language)** is a visual modeling language that is widely used in software engineering to represent **software designs.**

Standard language used for modeling software systems, and it includes various diagrams such as class diagrams, **use case diagrams, sequence diagrams, and state diagrams.**

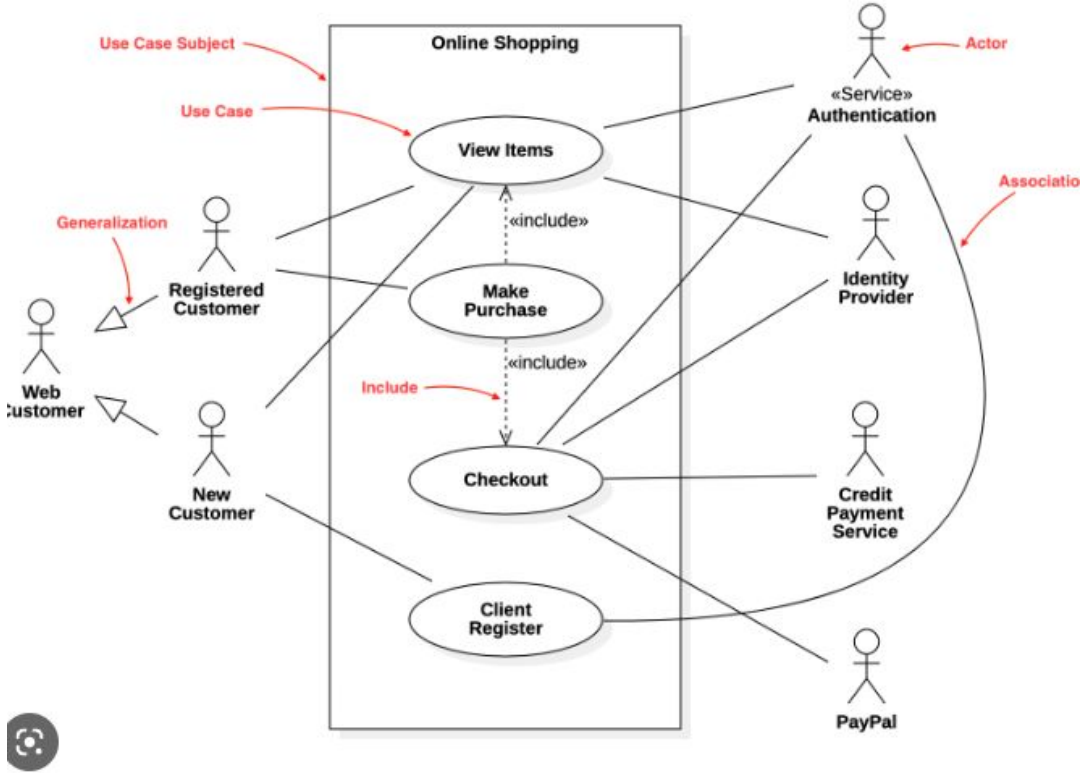
Use case diagrams are a type of UML diagram that is often used in software testing

Use case diagrams help testers understand the system's requirements and how it will be used by end-users.



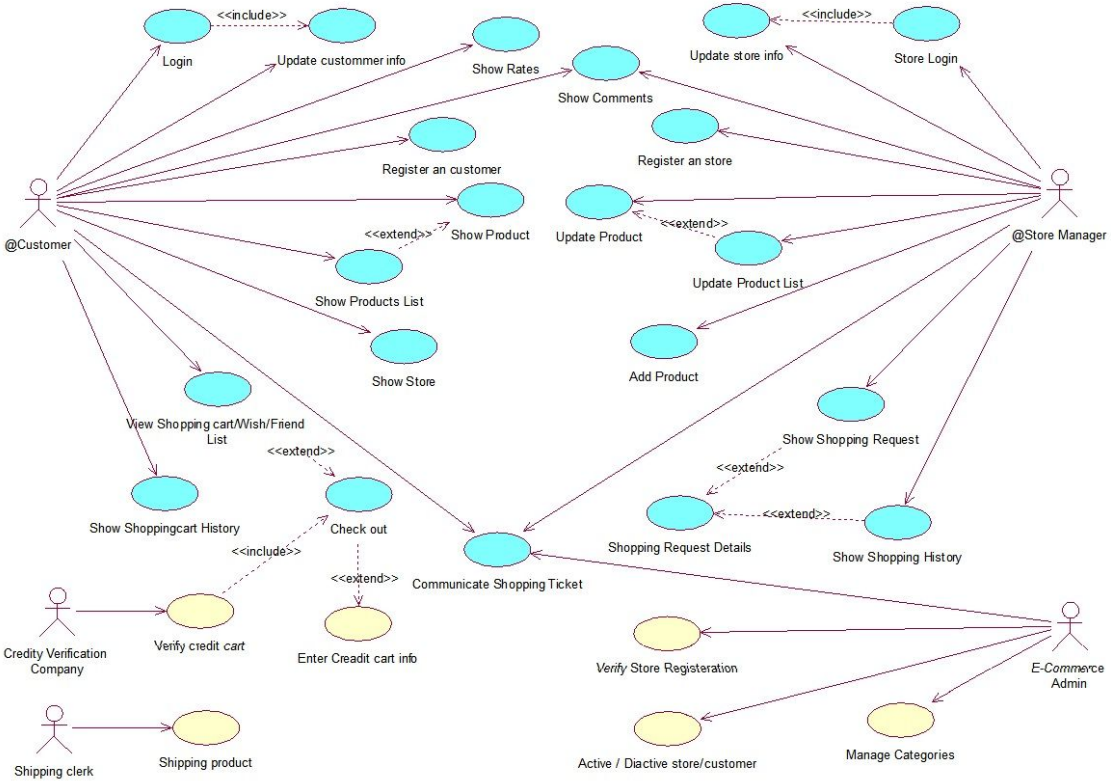




**sequence diagrams, and state diagrams.**


Use Case Subject
Online Shopping
Actor
Use Case
«Service»
Authentication
View Items
Associatio
Generalization
«include»
Registered
Customer
Identity
Provider
Make
Purchase
«include»
Include
Web
Customer
New
Customer
Checkout
Credit
Payment
Service
Client
Register
PayPal

# UML contains different diagrams and Use case is one of it

UML also contains Structural diagrams as well as - such as Class Diagram




https://stackoverflow.com/questions/9107448/uml-class-diagram-for-an-e-commerce-website

| Based on | BRD or BRS | SRS | FRD |
|---|---|---|---|
| What it includes? | BRD or a BRS includes the high-level business requirement of a system to be developed in layman language. | SRS document includes the functional and non-functional requirements and Use Cases. | FRD document consists of detailed requirements in technical terms and technical diagrams like UML, Data Flow, etc. |
| What is Answers? | BRD answers the **WHY** part i.e. Why the requirements are being prepared? | SRS answers the **WHAT** i.e. What requirements to be fulfilled. | FRS focuses on the **HOW** part i.e. How the requirements will be implemented. |
| When will it be prepared? | A BRD document is created during the analysis phase of the project. | SRS document is prepared during the planning phase of the project. | FRD or FRS document is also created during the planning phase of the project. |
| Who will be responsible for creating? | A BRD will be created by the business analysts. | Business Analyst and System Analyst work together to prepare an SRS document. | Since the functional requirement document is detailed and technical, it is created by Business Analyst, System Analysts, and Implementation team together. |
| Who will be using? | Business Requirements Document is developed for the business users, stakeholders, etc. | SRS document is prepared for the subject matter experts and technical leads. | FRD document will be used by the development team and quality assurance i.e. testing team. |

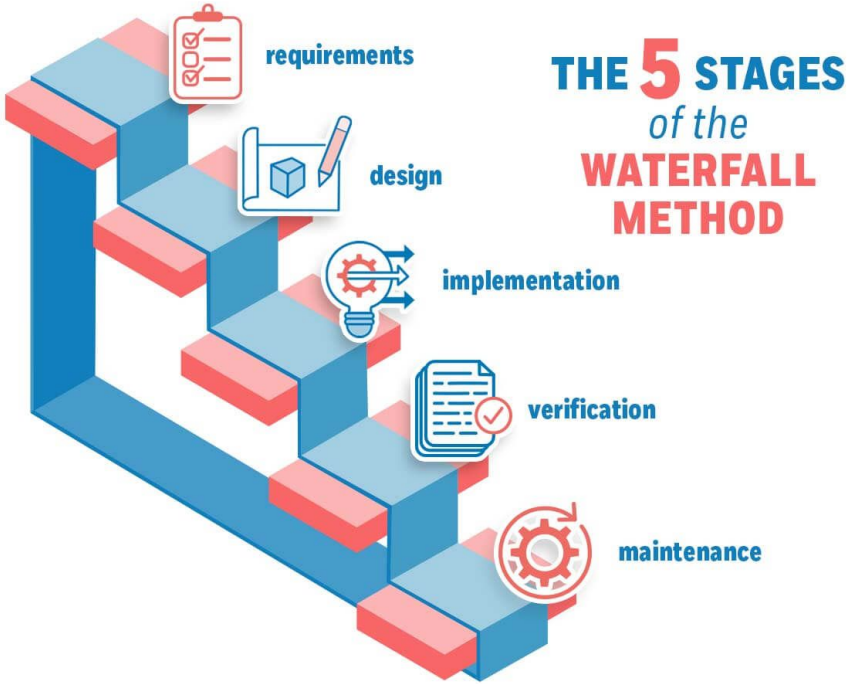
| Aspect | High-Level Design | Low-Level Design |
|---|---|---|
| Abstraction | High | Low |
| Granularity | Coarse | Fine |
| Focus | System architecture, modules, and interfaces | Algorithms, data structures, and code implementation |
| Detail | Conceptual | Technical |
| Goal | Overall system design, functionality, and requirements | Detailed implementation of specific features or components |
| Example | Use case diagram, architecture diagram, flow chart, block diagram | Pseudo code, class diagram, sequence diagram |

# SDLC Models

- Waterfall Model
- Spiral Model
- V-Model
- Agile Model

Other related methodologies are Agile Model, RAD Model, Rapid Application Development and Prototyping Models.

# Waterfall Model



# Waterfall Model

## ADVANTAGES
OF WATERFALL

- Simple method and easy to use
- Phases are clear
- Suitable for smaller projects
- Easy to manage

## DISADVANTAGES
OF WATERFALL

- Does not allow much revision
- Not suitable for complex projects
- Risk and uncertainty are high
- Does not include a feedback path

## Advantages of Waterfall Model

- The product will be of high quality.
- There are less possibilities of detecting problems because requirement modifications are prohibited.
- Since the testers are employed later, the initial cost is lower.
- Preferred for little projects with frozen criteria.

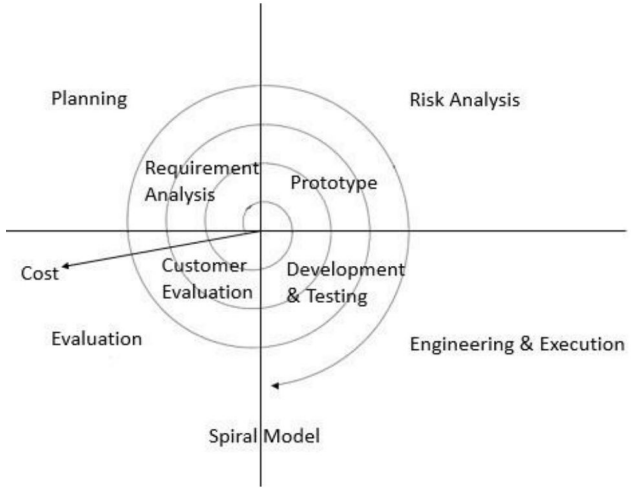
## Disadvantages of Waterfall Model

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

# Spiral Model

Spiral Model is iterative model.

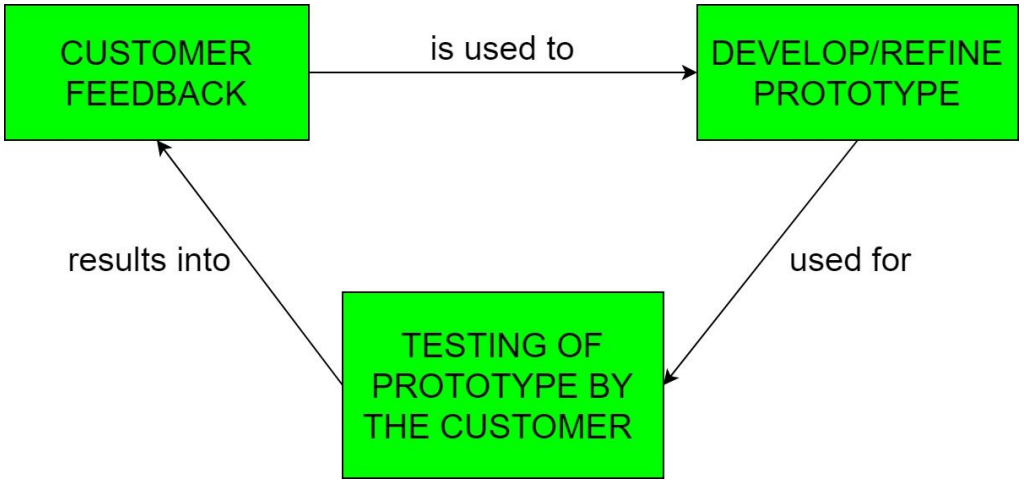
- Spiral Model overcome drawbacks of Waterfall model.
- We follow spiral model whenever there is dependency on the modules.
- In every cycle new software will be released to customer.
- Software will be released in multiple versions. So it is also called version control model.



The Radius of the spiral at any point represents the expenses(cost) of the project so far, and the angular dimension represents the progress made so far in the current phase.

# Prototyping Model

Prototyping is defined as the process of developing a working replication of a product or system that has to be engineered. It offers a small scale facsimile of the end product and is used for obtaining customer feedback as described below:



In each phase of the Spiral Model, the features of the product dated and analyzed, and the risks at that point in time are identified and are resolved through prototyping.

Thus, this model is much more flexible compared to other SDLC models.

## Risk Handling in Spiral Model

A risk is any adverse situation that might affect the successful completion of a software project. The most important feature of the spiral model is handling these unknown risks after the project has started. Such risk resolutions are easier done by developing a prototype. The spiral model supports coping up with risks by providing the scope to build a prototype at every phase of the software development.

## Advantages of Spiral Model

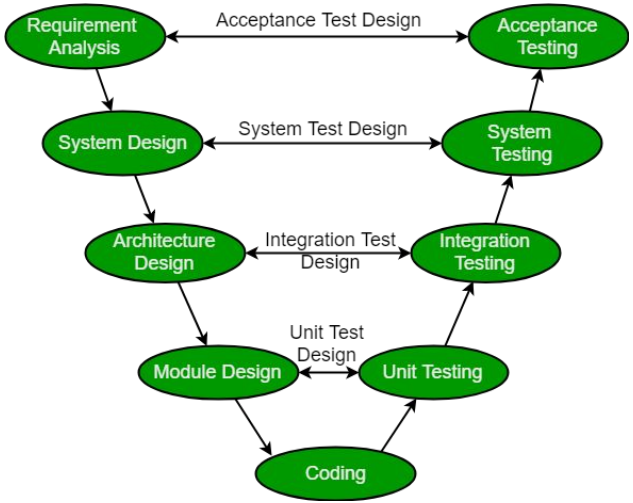
- Testing is done in every cycle, before going to the next cycle.
- Customer will get to use the software for every module.
- Requirement changes are allowed after every cycle before going to the next cycle.

## Disadvantages of Spiral Model

- Requirement changes are NOT allowed in between the cycle.
- Every cycle of spiral model looks like waterfall model.
- There is no testing in requirement & design phase.

# V-Model

The V-model is a type of SDLC model where process executes in a sequential manner in V-shape.



V-Model contains Verification phases on one side of the Validation phases on the other side.

| Verification | Validation |
| --- | --- |
| Verification means **Are we building the software right?** | Validation means **Are we building the right software ?** |
| Verification is the static testing. | Validation is the dynamic testing. |
| It does *not* include the execution of the code. | It includes the execution of the code. |
| Methods used in verification are reviews, walkthroughs, inspections and desk-checking. | Methods used in validation are Black Box Testing, White Box Testing and non-functional testing. |

## VERIFICATION

- 2 sleeves?
- Is it size L?
- Is it blue?
- Are any buttons missing?

## VALIDATION

- Does it fit?
- Is it comfortable to drive in?
- Does the colour match my eyes?
- Can I afford it?
- Is it good quality?
- Will my date like it?

**Verification**: It involves static analysis technique (review) done without executing code. It is the process of evaluation of the product development phase to find whether specified requirements meet.

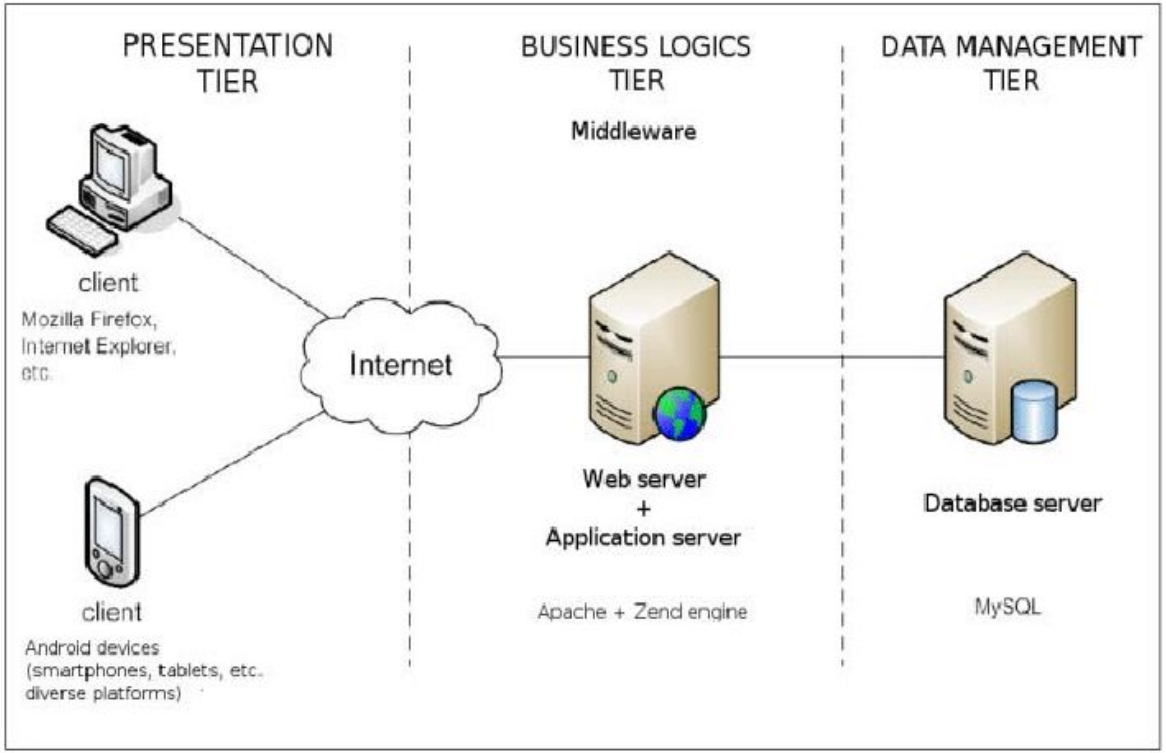
**Validation**: It involves dynamic analysis technique (functional, non-functional), testing done by executing code. Validation is the process to evaluate the software after the completion of the development phase to determine whether software meets the customer expectations and requirements.

**Advantages**

- Testing is involved in each and every phase.

**Disadvantages**

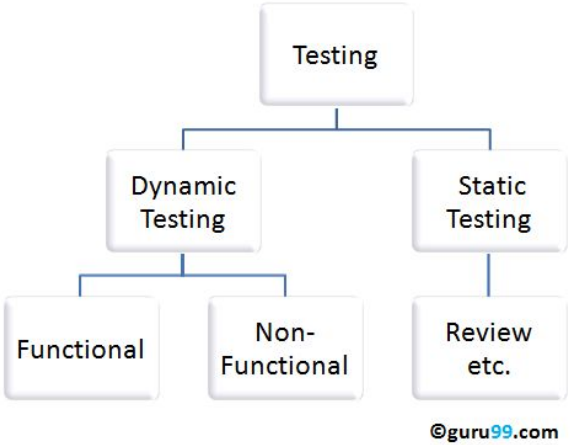
- Documentation is more.
- Initial investment is more.


PRESENTATION TIER
BUSINESS LOGICS TIER
Middleware
DATA MANAGEMENT TIER
client
Mozilla Firefox, Internet Explorer, etc.
Internet
Web server + Application server
Apache + Zend engine
Database server
MySQL
client
Android devices (smartphones, tablets, etc. diverse platforms)

# Static V/S Dynamic Testing

Static testing is an approach **to test project documents in the form of Reviews, Walkthroughs and Inspections.**

Dynamic testing is an approach **to test the actual software by giving inputs and observing results.**



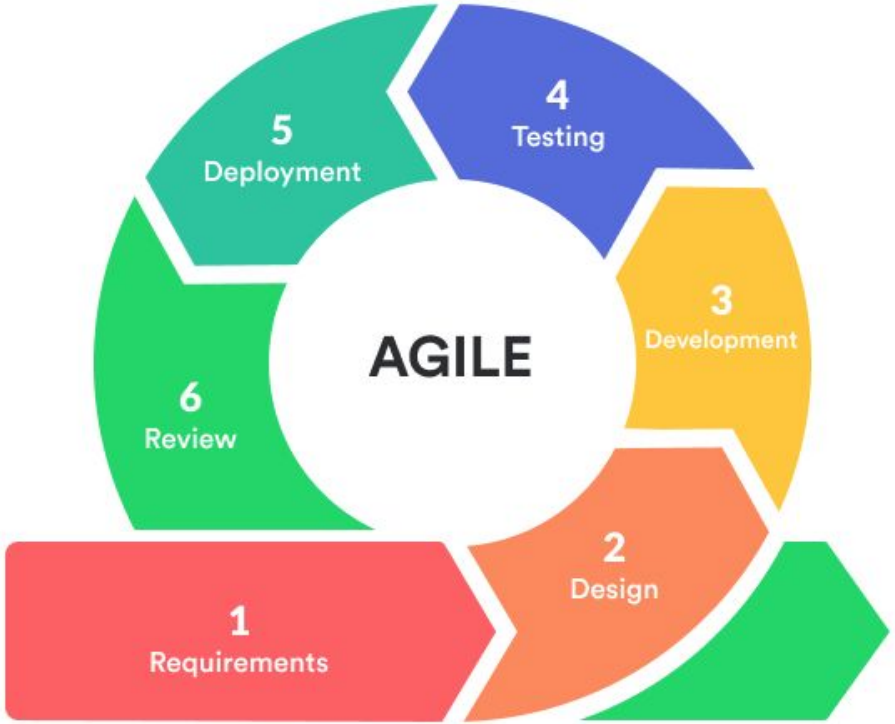
| Static Testing | Dynamic Testing |
| --- | --- |
| 1. Static Testing is white box testing which is done at early stage of development life cycle. It is more cost effective than dynamic testing | 1. Dynamic Testing on the other hand is done at the later stage of development lifecycle. |
| 2. Static Testing Methods include Walkthroughs, code review. | 2. Dynamic testing involves functional and nonfunctional testing |
| 3. It is done before code deployment and without execution of code. | 3. It is done after code deployment with the execution of codes. |
| 4. This type of testing comes under Verification. | 4. This type of testing comes under Validation. |
| 5. Static testing achieves 100% statement coverage in a relatively short time. | 5. Dynamic testing may involve running several test cases, each of which may take longer time. |
| 6. Static testing is about prevention. | 6. Dynamic testing is about cure. |
| 7. In Static Testing techniques a checklist is prepared for testing process. | 7. In Dynamic Testing technique the test cases are executed. |

# AGILE MODEL

The Agile model is a combination of an incremental and iterative approach and is focussed on fitting in well with **flexible requirements.**

Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches

the project is divided into small subparts and is delivered in iterations. The subtasks are divided into time frames to serve working functionality with each build. As a result, the final product has all the required features.


AGILE
1
Requirements
2
Design
3
Development
4
Testing
5
Deployment
6
Review

Instead of betting everything on a "big bang" launch, an agile team delivers work in small, but consumable, increments.

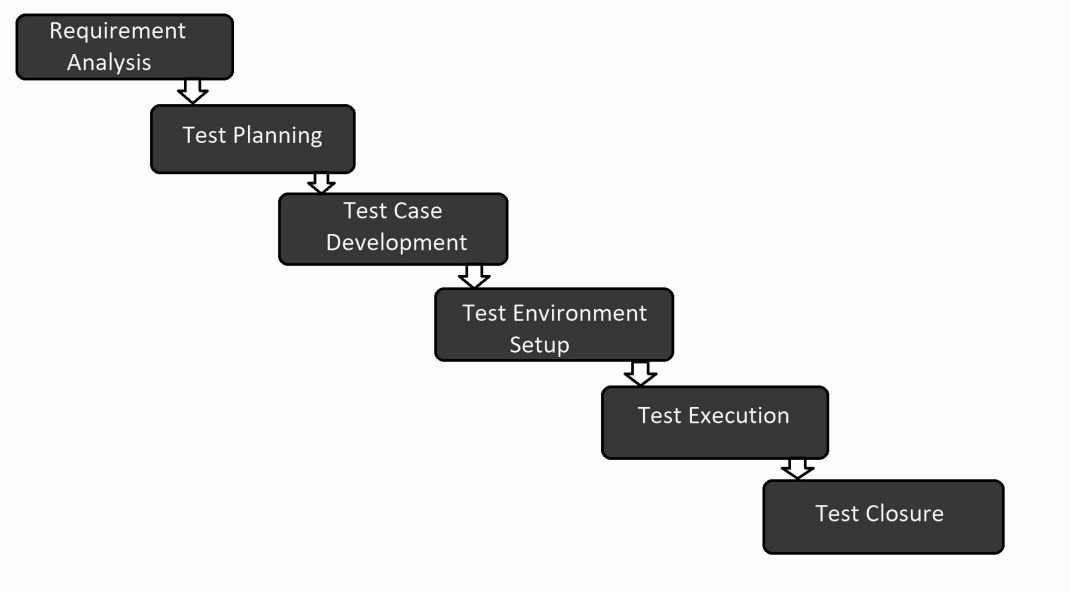
# Benefits:

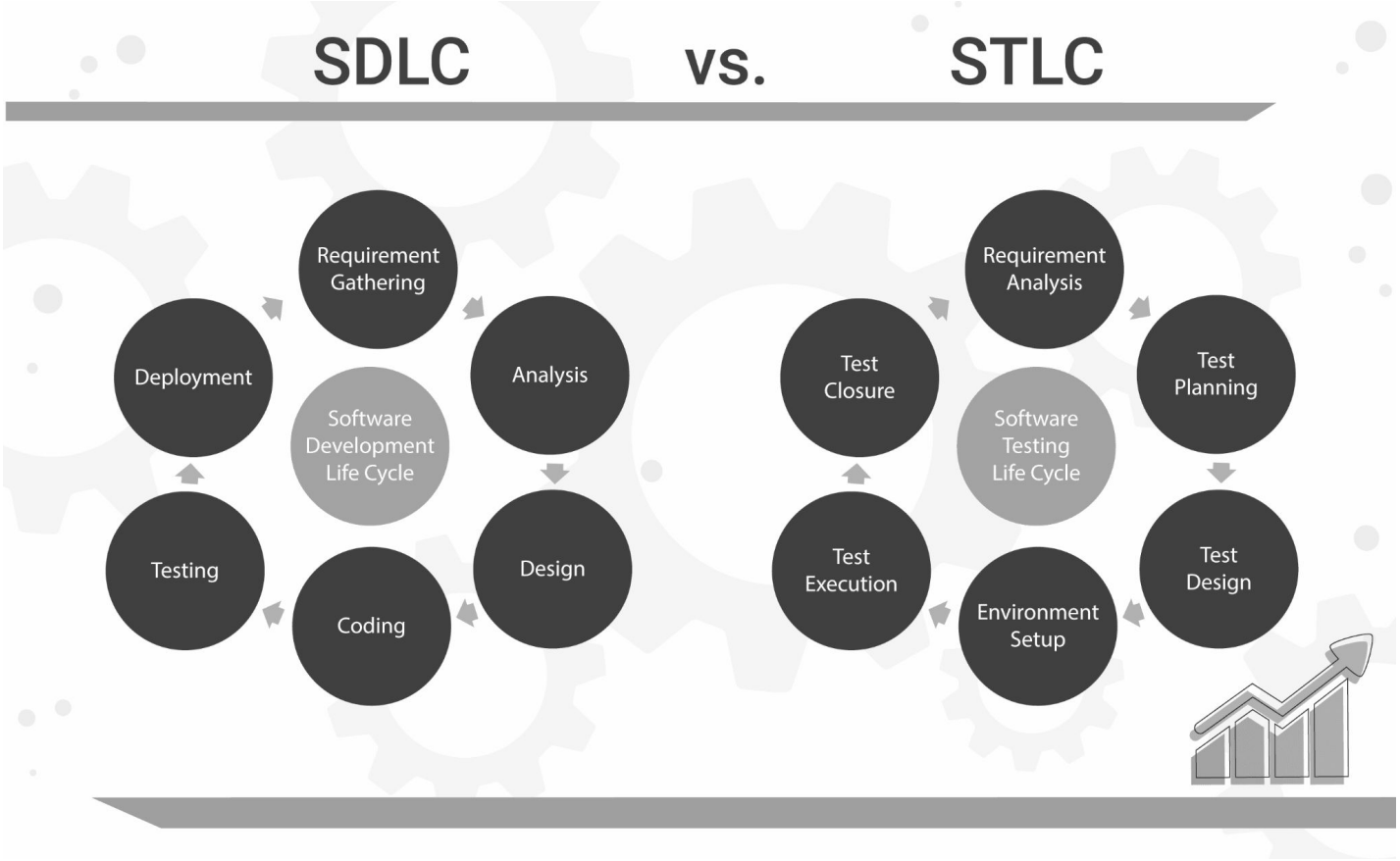
Quick development
Quality and measurable results
Business value can be delivered – demonstrated fast
Requires minimum resources
Highly adaptive to changing requirements


# Better Resume Tech.

# Software Testing Life Cycle (STLC)

**Software Testing Life Cycle (STLC)** is a sequence of different activities performed during the software testing process.




SDLC
vs.
STLC
Requirement Gathering
Analysis
Design
Coding
Testing
Deployment
Software Development Life Cycle
Requirement Analysis
Test Planning
Test Design
Environment Setup
Test Execution
Test Closure
Software Testing Life Cycle

1. **Requirement Analysis:** Quality assurance team understands the requirements like what is to be tested. If anything is missing or not understandable then quality assurance team meets with the stakeholders to better understand the detail knowledge of requirement. - **Documents - SRS, FRD, BRD**
2. **Test Planning:** In this phase manager of the testing team calculates estimated effort and cost for the testing work. This phase gets started once the requirement gathering phase is completed. Documents **Test Plan**

**Test Case Development:** The test case development phase gets started once the test planning phase is completed Test cases. **Documents - Test cases(Excel, GSheet or on Tools)**

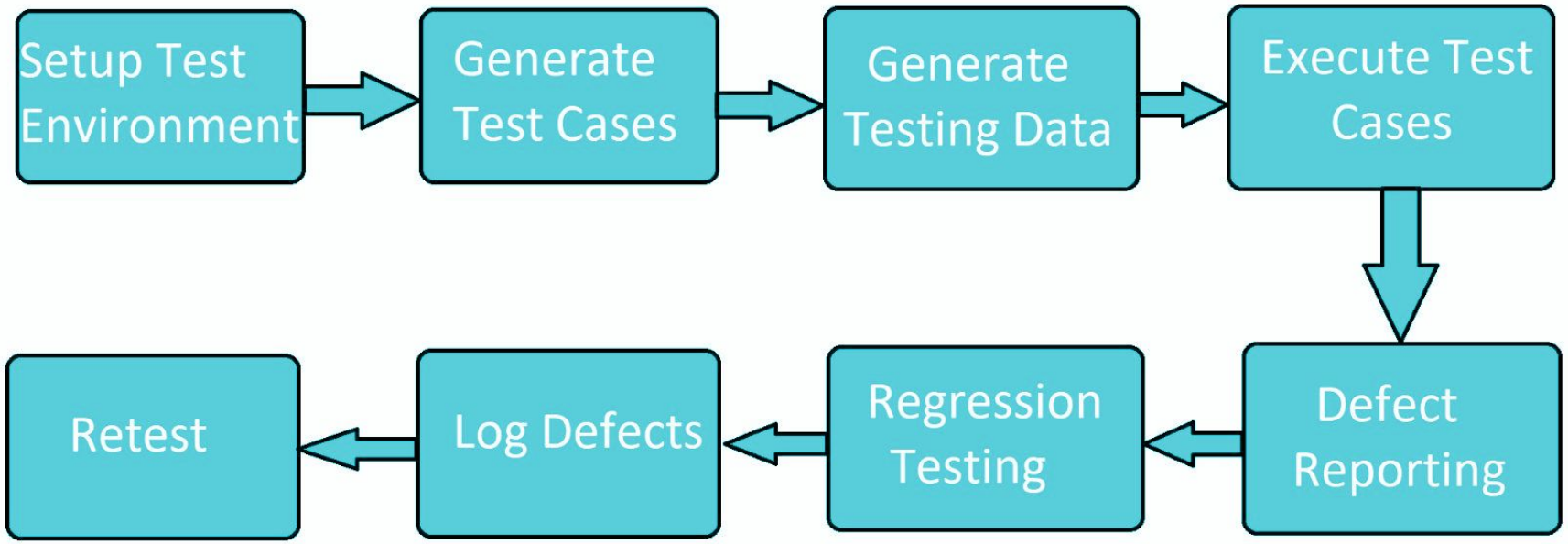
**Test Environment Setup:** Test environment decides the conditions on which software is tested.  NA

**Test Execution: Documents - Test Execution Report Bug Report ,**

In this phase testing team start executing test cases based on prepared test cases in the earlier step.

**Test Closure: Test Report**

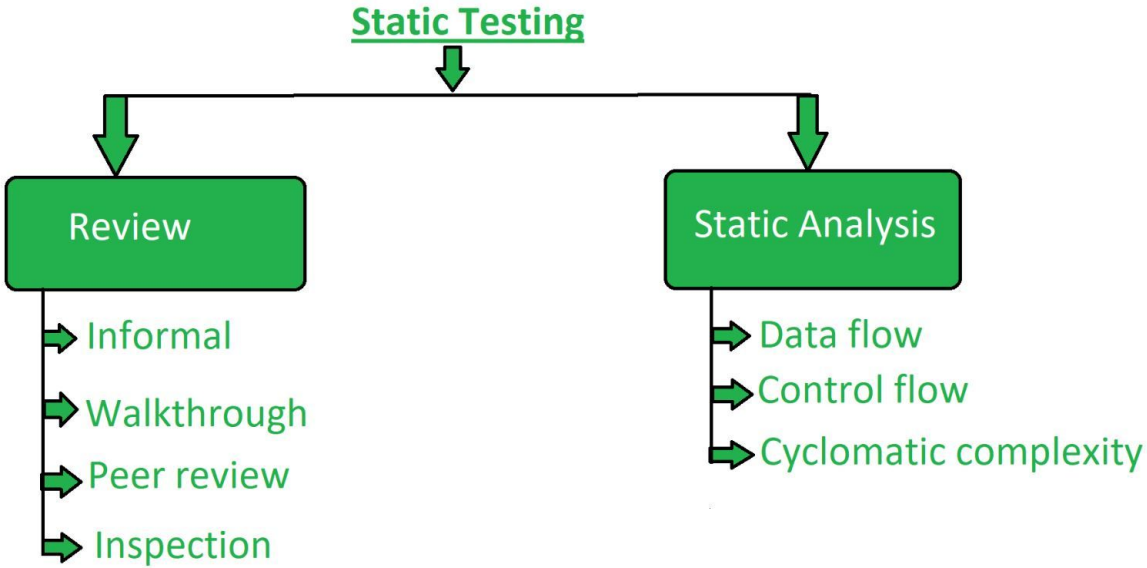
This is the last stage of STLC in which the process of testing is analyzed.


Setup Test Environment
Generate Test Cases
Generate Testing Data
Execute Test Cases
Defect Reporting
Regression Testing
Log Defects
Retest

| Black Box Testing | White Box Testing |
| --- | --- |
| It is also called Specification Based Technique. | It is also called Structural Testing Technique. |
| Internal structure and coding knowledge is not required. | Internal structure and coding knowledge is required. |
| Main concentrate on functionality of system. | Main concentrate on code structure ,branches , loops, conditions etc. |
| Implementation knowledge is not required. | Implementation knowledge is required. |

# Static Testing

**Static Testing** is a type of a Software Testing method which is performed to check the defects in software without actually executing the code of the software application.



# Static Testing done with Static Analysis

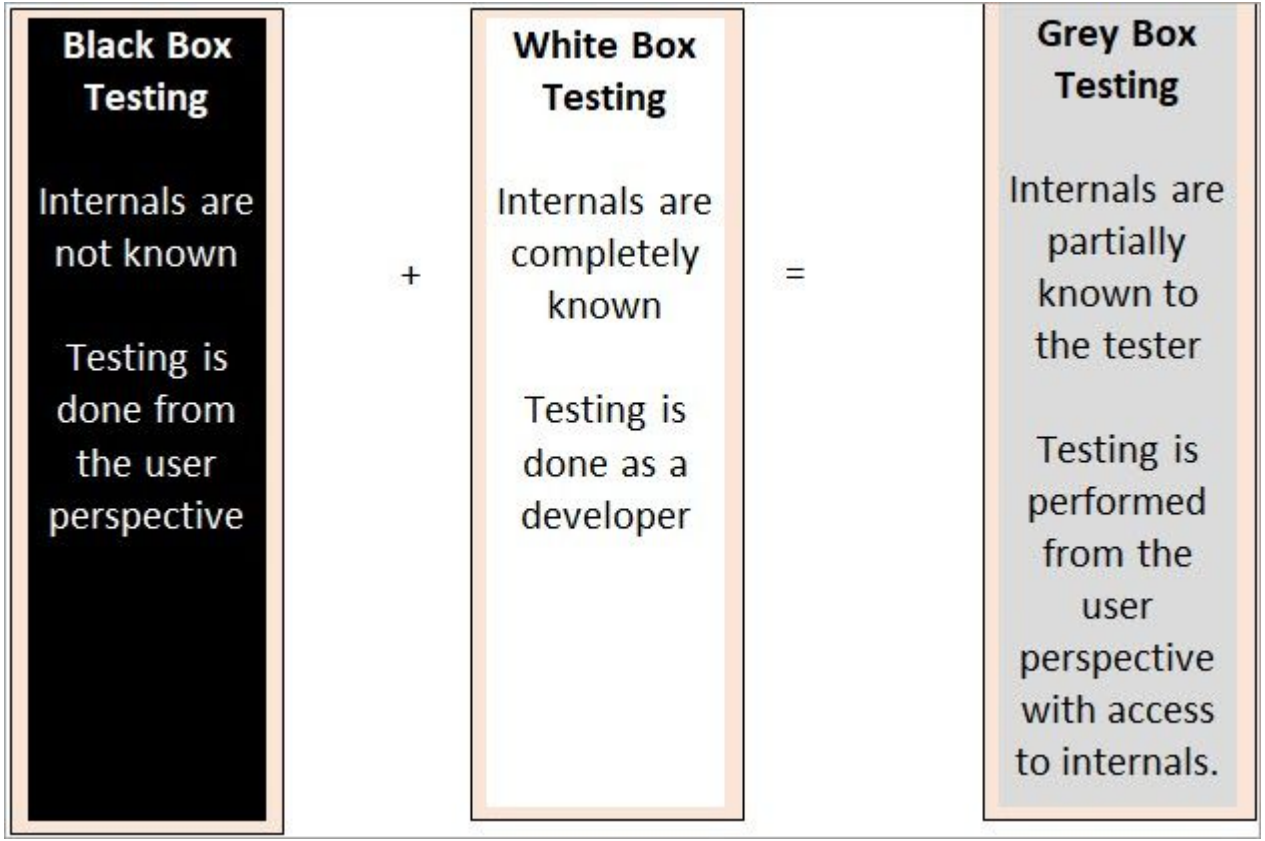
Static Analysis:
Static Analysis includes the evaluation of the code quality that is written by developers. Different tools are used to do the analysis of the code and comparison of the same with the standard.
It also helps in following identification of following defects:

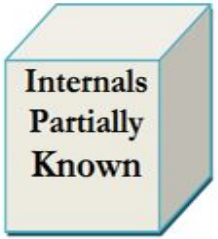
(a) Unused variables
(b) Dead code
(c) Infinite loops
(d) Variable with undefined value
(e) Wrong syntax

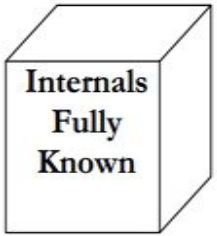
| Static Testing | Dynamic Testing |
| --- | --- |
| It is performed in the early stage of the software development. | It is performed at the later stage of the software development. |
| In static testing whole code is not executed. | In dynamic testing whole code is executed. |
| Static testing prevents the defects. | Dynamic testing finds and fixes the defects. |
| Static testing is performed before code deployment. | Dynamic testing is performed after code deployment. |
| Static testing is less costly. | Dynamic testing is highly costly. |
| Static Testing involves checklist for testing process. | Dynamic Testing involves test cases for testing process. |

# Testing Methodologies

- White box Testing
- Black box Testing
- Grey box Testing

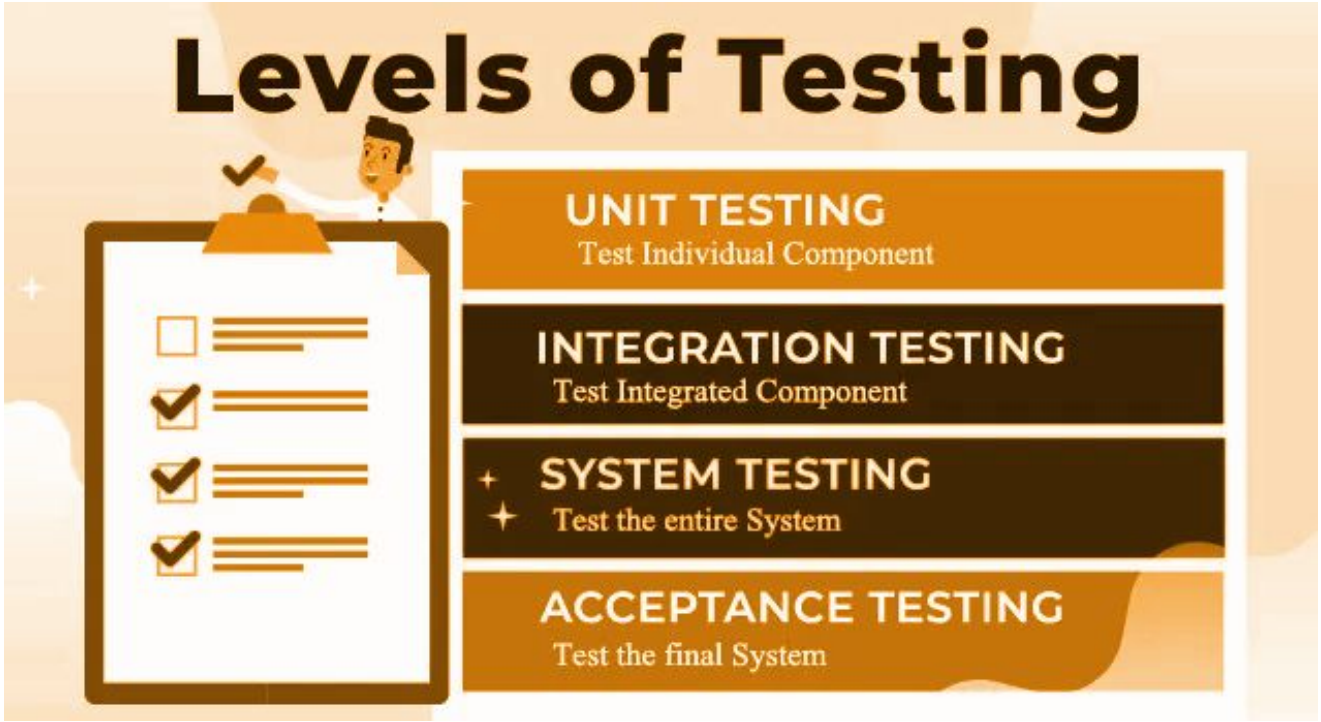

Black Box Testing
Internals are not known
Testing is done from the user perspective
+
White Box Testing
Internals are completely known
Testing is done as a developer
=
Grey Box Testing
Internals are partially known to the tester
Testing is performed from the user perspective with access to internals.

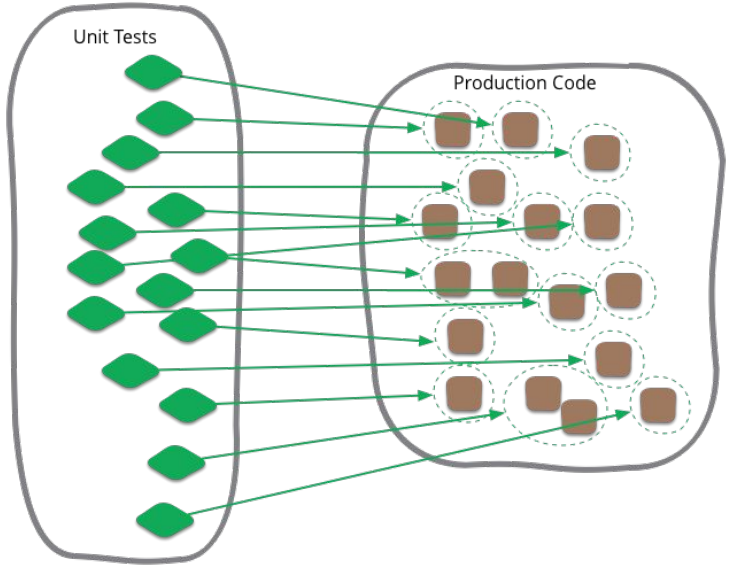

**Comparison between the Three Testing Types**

| | Black Box Testing | Grey Box Testing | White Box Testing |
|---|---|---|---|
| 1. | The Internal Workings of an application are not required to be known | Somewhat knowledge of the internal workings are known | Tester has full knowledge of the Internal workings of the application |
| 2. | Also known as closed box testing, data driven testing and functional testing | Another term for grey box testing is translucent testing as the tester has limited knowledge of the insides of the application | Also known as clear box testing, structural testing or code based testing |
| 3. | Performed by end users and also by testers and developers | Performed by end users and also by testers and developers | Normally done by testers and developers |
| 4. | -Testing is based on external expectations<br>-Internal behavior of the application is unknown | Testing is done on the basis of high level database diagrams and data flow diagrams | Internal workings are fully known and the tester can design test data accordingly |
| 5. | This is the least time consuming and exhaustive | Partly time consuming and exhaustive | The most exhaustive and time consuming type of testing |
| 6. | Not suited to algorithm testing | Not suited to algorithm testing | Suited for algorithm testing |
| 7. | This can only be done by trial and error method | Data domains and Internal boundaries can be tested, if known | Data domains and Internal boundaries can be better tested |


Levels of Testing
UNIT TESTING
Test Individual Component
INTEGRATION TESTING
Test Integrated Component
SYSTEM TESTING
Test the entire System
ACCEPTANCE TESTING
Test the final System

# Unit Testing

**A unit is a single component** or module of a software.
Unit testing conducts on a single program or single module.
Unit Testing is white box testing technique.
**Conducted by the developers.**




Unit testing techniques:

- Basis path testing
- Control structure testing
- Conditional coverage
- Loops Coverage.
- Mutation Testing

# Unit Testing

## Demo of Web App unit

```
public static boolean isSumPositive(int a, int b){
    // Statement Coverage Unit Example
    if(a+b > 0){
        return true;
    }else{
        return false;
    }
```

isSumPositive function

```
@Test
public void shouldAnswerWithPass()
{
    Assert.assertTrue(AddModule.isSumPositive( a: 2, b: 3));
}

@Test
public void shouldAnswerWithFail()
{
    Assert.assertTrue(AddModule.isSumPositive( a: 2, b: -3));
    //Assert.assertFalse(AddModule.isSumPostive(2,-3));
}
```

# Integration Testing

- - Integration testing performed between 2 or more modules.
- - Integration testing focuses on checking **data communication between multiple**
- modules.
- - Integrated Testing is white box testing technique
-

| | Unit Testing | Integration Testing |
|---|---|---|
|  | ✓ | ✗ |
|  | ✓ | ✓ |

# Integration Testing

- Incremental
- Big Bang Approach

# Big Bang Approach

All the components or modules are integrated together at once and then tested as a unit.

- Convenient for small systems.

**We might miss data flow between some of the modules.**

**If you find any defect we can't understand the root cause of defect**

**some interfaces link to be tested could be missed easily.**

- Since the Integration testing can commence only after "all" the modules are designed, **the testing team will have less time for execution in the testing phase.**

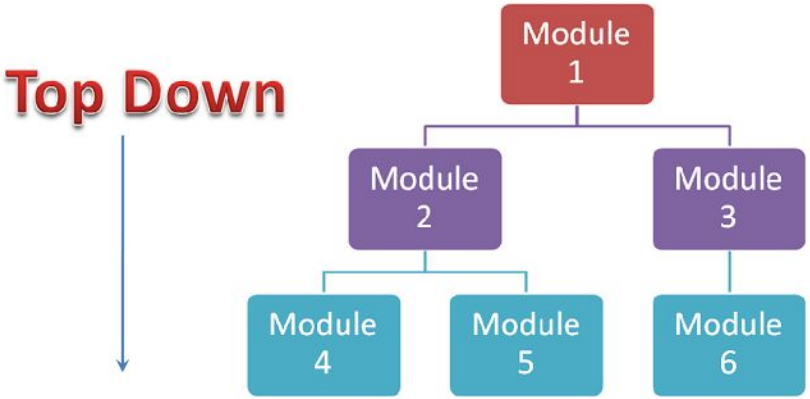
high-risk critical modules are not isolated and tested on priority.

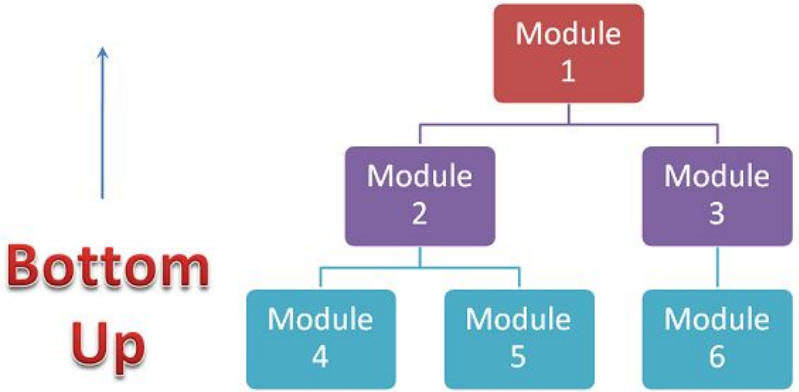
# Incremental Integration Testing

- Testing is done by integrating two or more modules that are logically related to each other and then tested for proper functioning of the application.

- Then the other related modules are integrated incrementally and the process continues until all the logically related modules are integrated and tested successfully.

# Incremental Integration Testing



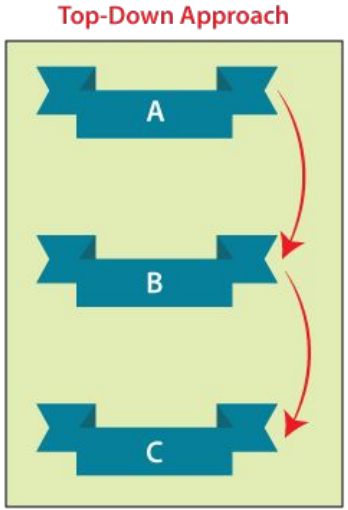
# Incremental Integration Testing

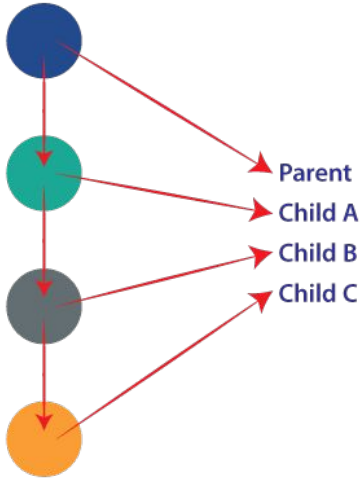
- Testing is done by integrating two or more modules that are logically related to each other and then tested for proper functioning of the application.

- Then the other related modules are integrated incrementally and the process continues until all the logically related modules are integrated and tested successfully.

# Top-down Integration Testing

- We will add the modules incrementally or one by one and test the data flow in similar order as we can see in the below diagram:



# Difference between Stubs and Drivers

The **Stubs and Drivers** are considered as elements which are equivalent to to-do modules that could be replaced if modules are in their developing stage, missing or not developed yet

**Stubs are mainly used in Top-Down integration testing** while the **Drivers are used in Bottom-up integration testing,** thus increasing the efficiency of testing process.

# Difference between Stubs and Drivers

Module-A : Login page website,
Module-B : Home page of the website
Module-C : Profile setting
Module-D : Sign-out page

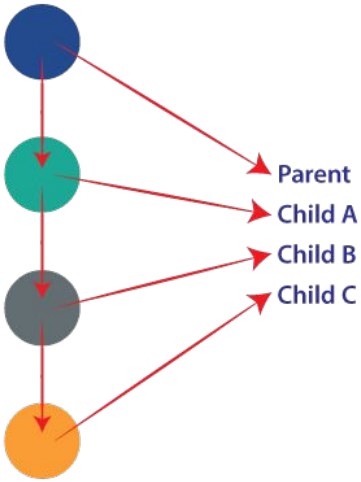
**Assume Module-A is developed.** As soon as it's developed, it undergoes testing, but it requires Module-B, which isn't developed yet. So in this case, we can use the Stubs or Drivers that simulate all features and functionality that might be shown by actual Module-B. So, we can conclude that Stubs and drivers are used to fulfill the necessity of unavailable modules.

# Difference between Stubs and Drivers

| S.No. | Stubs | Drivers |
|---|---|---|
| 1. | Stubs are used in Top-Down Integration Testing. | Drivers are used in Bottom-Up Integration Testing. |
| 2. | Stubs are basically known as a “called programs” and are used in the Top-down integration testing. | While, drivers are the “calling program” and are used in bottom-up integration testing. |
| 3. | Stubs are similar to the modules of the software, that are under development process. | While drivers are used to invoking the component that needs to be tested. |

# Top-down Integration Testing

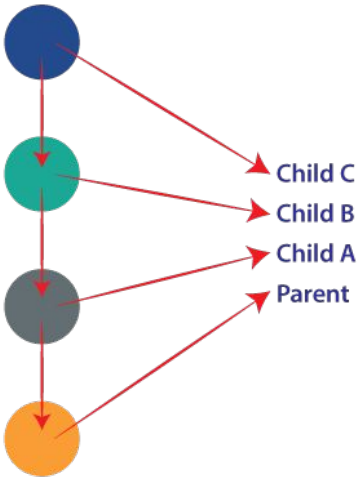
- To detect the significant design flaws and fix them early because required modules are tested first.

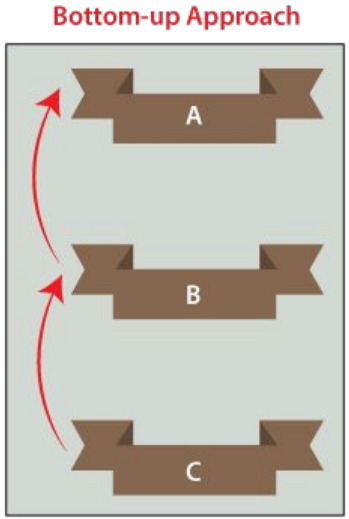



- Critical Modules are tested on priority; major design flaws could be found and fixed first.

# Bottom Up Integration Testing

- Low lower-level modules are tested with higher-level modules until all the modules have been tested successfully.



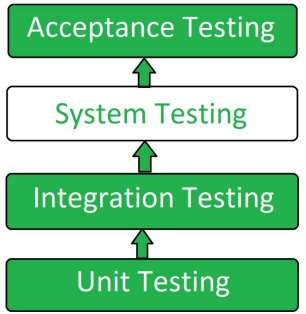

- Critical modules (at the top level of software architecture) which control the flow of application are tested last and may be prone to defects.

| S.NO. | Comparison Basis | Top-Down Integration Testing | Bottom-up Integration Testing |
|---|---|---|---|
| 1. | **Definition** | We will add the modules incrementally or one by one and test the data flow in similar order. | The lower-level modules are tested with higher-level modules until all the modules have been tested successfully. |
| 2. | **Executed on** | The top-down integration testing approach will be executed on the Structure or procedure-oriented programming languages. | The bottom-up integration testing approach will be executed on Object-oriented programming languages. |
| 3. | **Observation** | In the top-down approach, the observation of test output is more complicated. | In the bottom-up approach, the observation of test output is more accessible. |

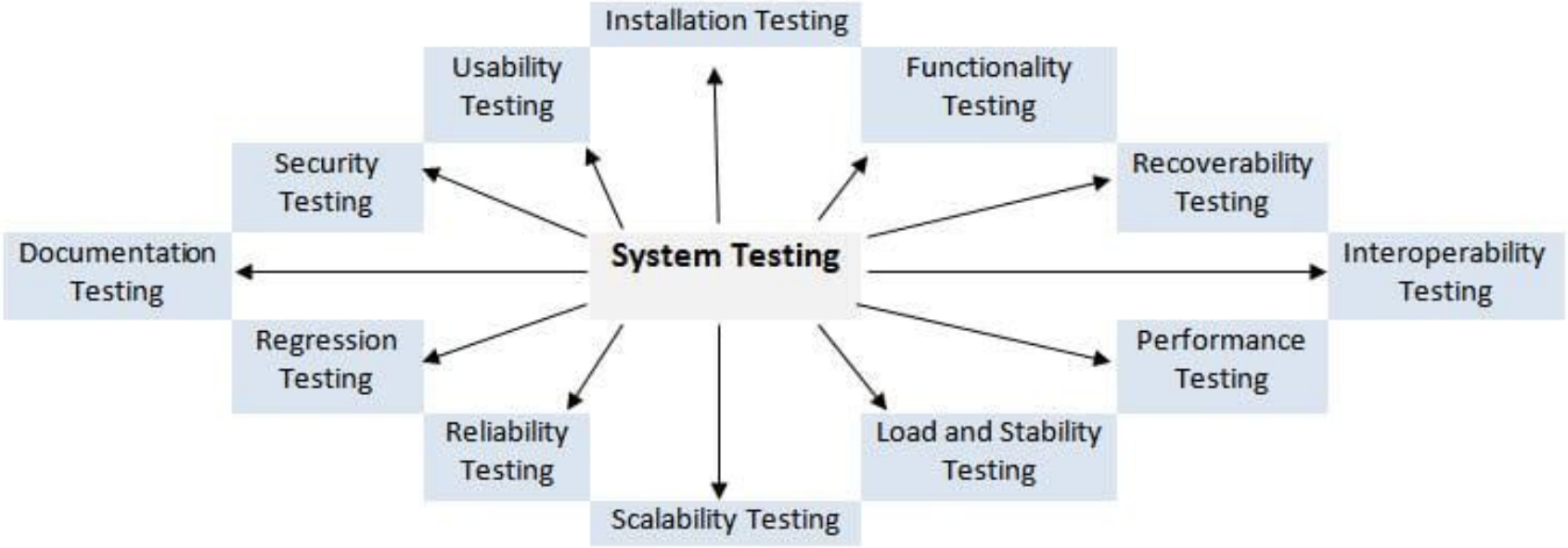
| | Top-Down Integration Testing | Bottom-up Integration Testing |
|---|---|---|
| **Advantage** | Following are some of the significant benefits of using top-down integration testing:<br>◦ In this, the early prototype is possible.<br>◦ Fault Localization is easier. | Below are some of the essential advantages of using bottom-up integration testing:<br>◦ We do not need to wait for the development of all the modules as it saves time.<br>◦ Identification of defects is easy. |
| **Disadvantage** | Some of the most common drawbacks of the top-down approach are as follows:<br>◦ Lower-level modules are tested ineffectively.<br>◦ Due to the high number of stubs, it gets pretty complicated.<br>◦ Critical Modules are tested first so that fewer chances of defects. | The disadvantage of the bottom-up approach is as follows:<br>◦ Compulsory modules are tested last, due to which the defects can occur.<br>◦ There is no possibility of an early prototype. |

# System Testing

- **Testing over all functionality of the application with respective client**
- **requirements.**
- It is a black box testing technique.
- This testing is conducted by testing team.
- After completion of component and integration level testing's we start System testing.
- Before conducting system testing we should know the **customer requirements.**
- System Testing focus on.
  - User Interface Testing (GUI)
  - Functional Testing
  - Non-Functional Testing
  - Usability Testing



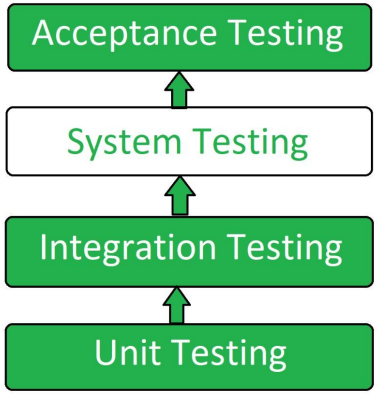
# System Testing Types.



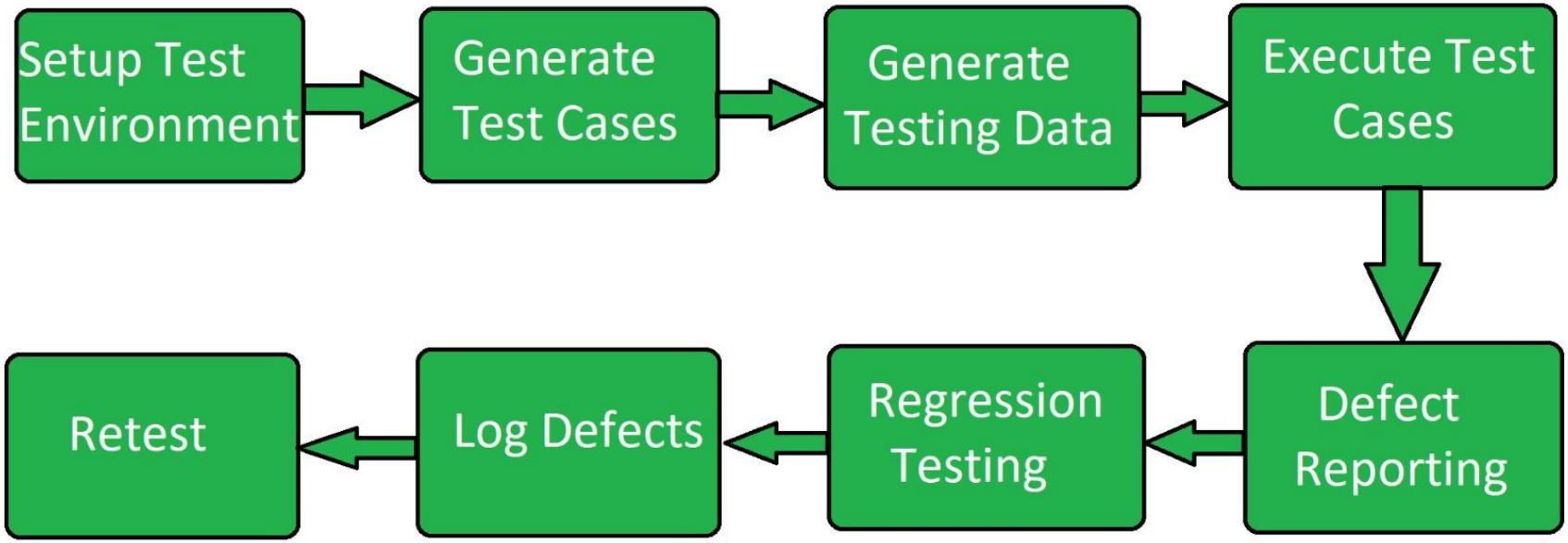
# System Testing

System Testing Process: System Testing is performed in the following steps:

Test Environment Setup: Create testing environment for the better quality testing.
Create Test Case: Generate test case for the testing process.
Create Test Data: Generate the data that is to be tested.
Execute Test Case: After the generation of the test case and the test data, test cases are executed.
Defect Reporting: Defects in the system are detected.
Regression Testing: It is carried out to test the side effects of the testing process.
Log Defects: Defects are fixed in this step.
Retest: If the test is not successful then again test is performed.



# System Testing



# Types of System Testing:

**Performance Testing**: Performance Testing is a type of software testing that is carried out to test the speed, scalability, stability and reliability of the software product or application.

**Load Testing**: Load Testing is a type of software Testing which is carried out to determine the behavior of a system or software product under extreme load.

**Stress Testing**: Stress Testing is a type of software testing performed to check the robustness of the system under the varying loads.

**Scalability Testing:** Scalability Testing is a type of software testing which is carried out to check the performance of a software application or system in terms of its capability to scale up or scale down the number of user request load.

**Tools used for System Testing :**

JMeter
Gallen Framework
Selenium

# User Acceptance Testing (UAT)

Acceptance Testing is a method of software testing where a system is tested for acceptability

After completion of system testing UAT team conducts acceptance testing in two levels.

- Alpha testing
- Beta testing

It is a formal testing according to user needs, requirements and business processes conducted to determine whether a system satisfies the acceptance criteria or not and to enable the users, customers or other authorized entities to determine whether to accept the system or not.

# Types of Acceptance Testing

User Acceptance Testing (UAT): User acceptance testing is used to determine whether the product is working for the user correctly. **Specific requirements** which are quite often used by the customers are primarily picked for the testing purpose. This is also termed as End-User Testing.

Business Acceptance Testing (BAT): BAT is used to determine whether the product meets the business goals and purposes or not.

BAT mainly focuses on business profits which are quite challenging due to the changing market conditions and new technologies so the current implementation may have to being changed which results in extra budgets.

# Types of Acceptance Testing

Contract Acceptance Testing (CAT) :
CAT is a **contract that specifies that once the product goes live, within a predetermined period**, the acceptance test must be performed and it should pass all the acceptance use cases. Here is a contract termed a Service Level Agreement (SLA)

**Regulations Acceptance Testing** (RAT): RAT is used to determine whether the **product violates the rules and regulations that are defined by the government** of the country where it is being released. T

# Types of Acceptance Testing

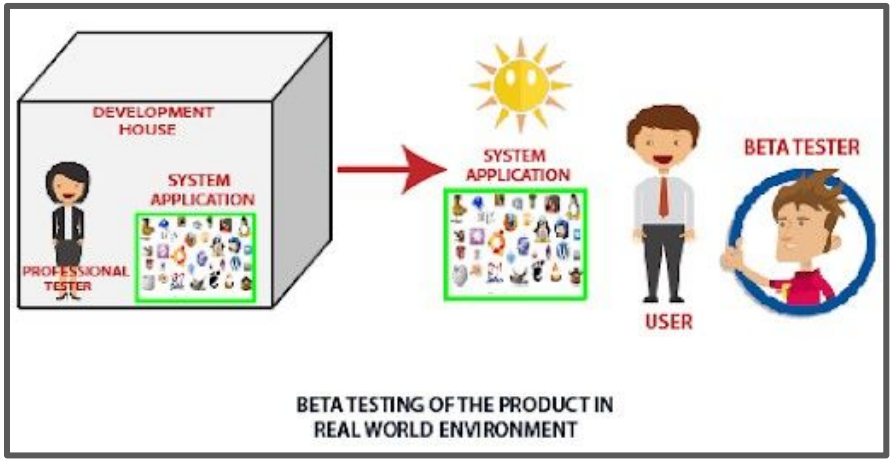
Operational Acceptance Testing (OAT): OAT is used to determine the **operational readiness of the product and is non-functional testing.** It mainly includes testing of recovery, compatibility, maintainability, reliability, etc. OAT assures the stability of the product before it is released to production.

# Types of Acceptance Testing

Alpha Testing: Alpha testing is used to team usually called alpha testers.**determine the product in the development testing environment by a specialized testers**

Beta Testing: Beta testing is used to **assess the product by exposing it to the real end-users,** usually called beta testers in their environment. Feedback is collected from the users and the defects are fixed. Also, this helps in enhancing the product.

# Alpha vs Beta

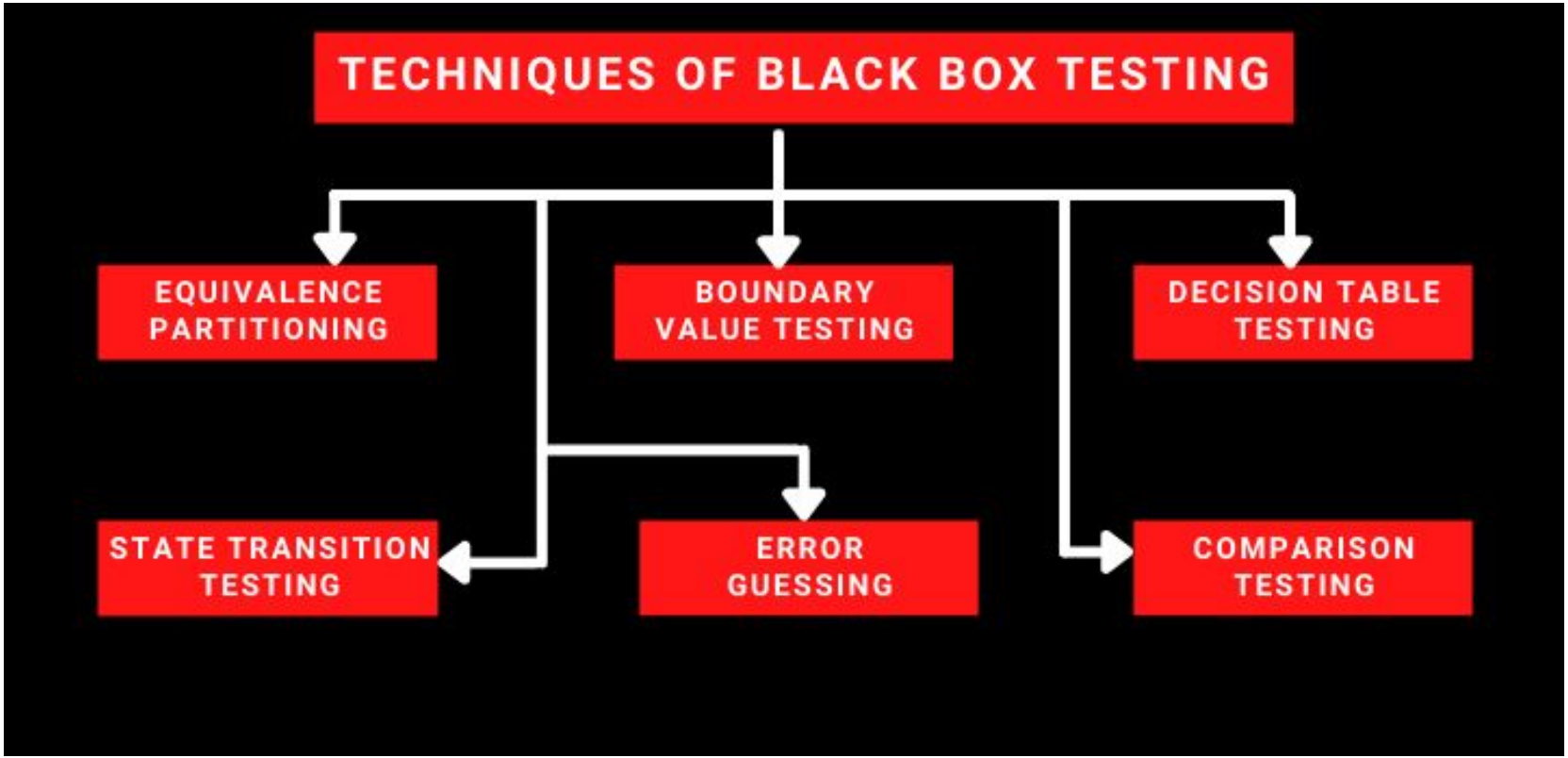




Tester
Dev Env

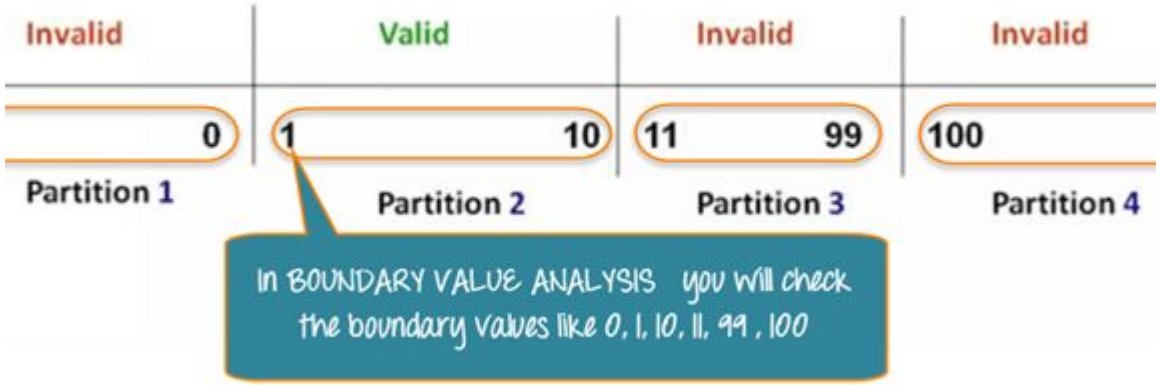
Beta Tester
Real Env or UAT env

# Test Design

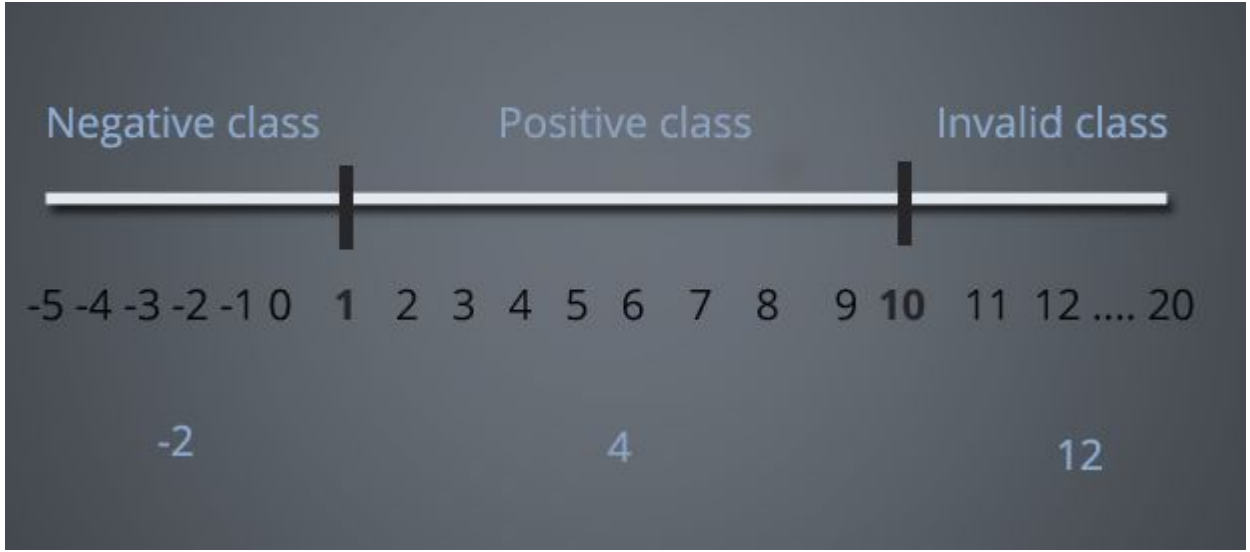
- Reduce Test Case
- More Test Coverage


TECHNIQUES OF BLACK BOX TESTING
EQUIVALENCE PARTITIONING
BOUNDARY VALUE TESTING
DECISION TABLE TESTING
STATE TRANSITION TESTING
ERROR GUESSING
COMPARISON TESTING

# Boundary Value Analysis.



# Equivalence Class Partitioning



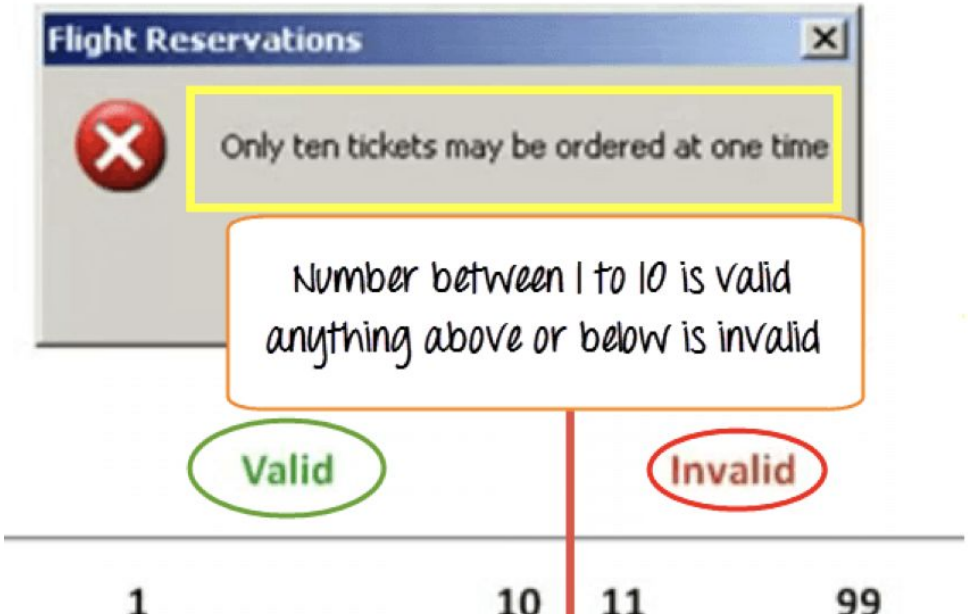
# Practical Test cases

Suppose, **In Ecommerce Sale** we have discounts like this ,
Invalid and Valid Test cases

| Nil | 0- 5000 | 5000-10000 | 10000-20000 |
|---|---|---|---|
| 0% | 10% | 15% | 25% |
| | | | |
| | | | |

https://forms.gle/hfU6xkKT7jVwmhPR6

# ECP Vs BVP

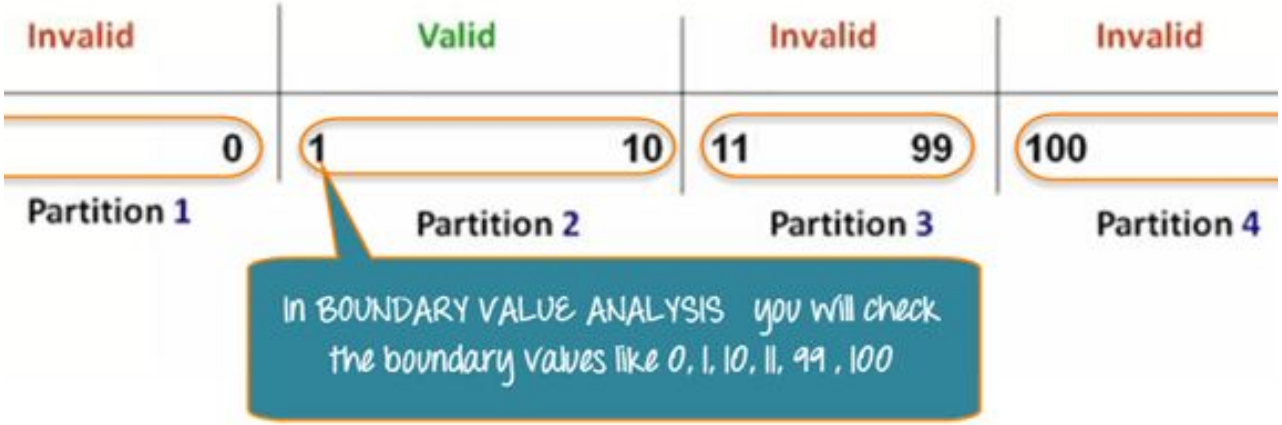



Equivalence partitioning and boundary value analysis(BVA) are closely related and can be used together at all levels of testing.

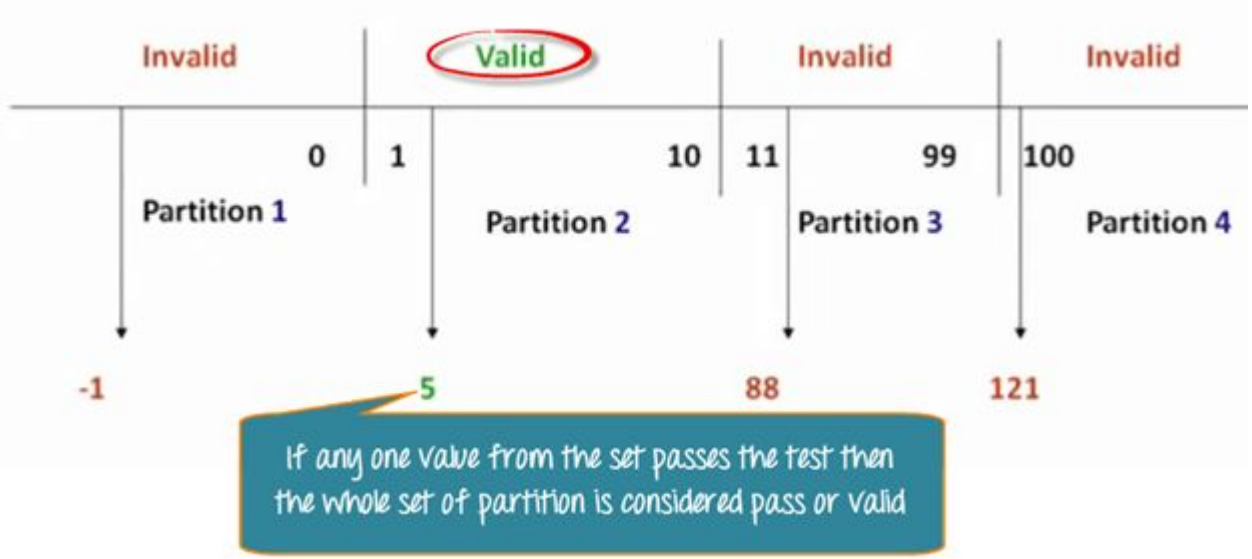
# Problem

Let’s consider the behavior of Order Pizza Text Box Below
Pizza values **1 to 10 is considered valid.** A success message is shown.
While value 11 to 99 are considered invalid for order and an error message will appear, “**Only 10 Pizza can be ordered**”

# Problem


Invalid
Valid
Invalid
Invalid
0
1
10
11
99
100
Partition 1
Partition 2
Partition 3
Partition 4
In BOUNDARY VALUE ANALYSIS you will check the boundary values like 0, 1, 10, 11, 99 , 100

# Problem


Invalid
Valid
Invalid
Invalid
0
1
10
11
99
100
Partition 1
Partition 2
Partition 3
Partition 4
-1
5
88
121
If any one value from the set passes the test then the whole set of partition is considered pass or valid

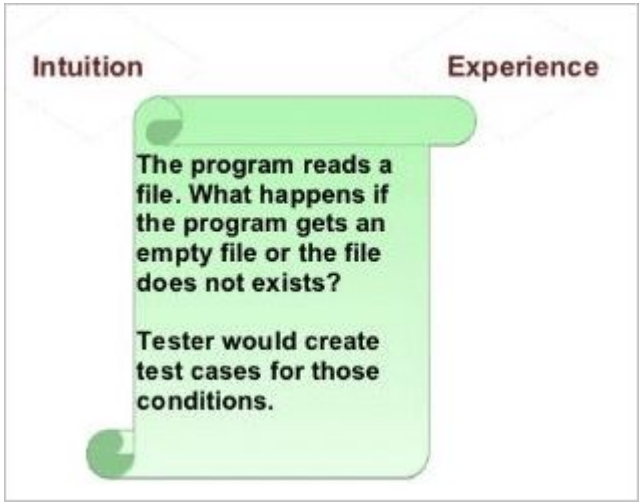
# Decision Table based testing

| ID | CONDITIONS/ACTIONS | TEST CASE 1 | TEST CASE 2 | TEST CASE 3 | TEST CASE 4 |
|---|---|---|---|---|---|
| Condition 1 | Valid User ID | T | T | F | F |
| Condition 2 | Valid Password | T | F | T | F |
| Action 1 | Home Page | Execute | | | |
| Action 2 | Show a Message as 'Invalid User Credentials' | | Execute | Execute | Execute |

# Decision Table based testing

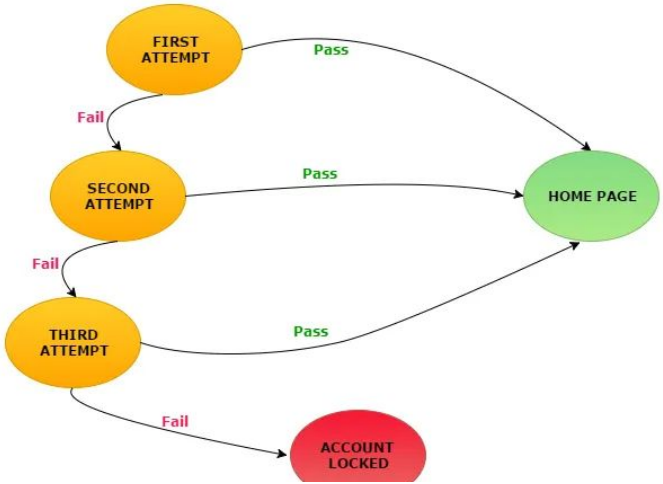
| Conditions | Case 1 | Case 2 | Case 3 | Case 4 | Case 5 | Case 6 | Case 7 | Case 8 |
|---|---|---|---|---|---|---|---|---|
| Format | .jpg | .jpg | .jpg | .jpg | Not .jpg | Not .jpg | Not .jpg | Not .jpg |
| Size | Less than 32kb | Less than 32kb | >= 32kb | >= 32kb | Less than 32kb | Less than 32kb | >= 32kb | >= 32kb |
| resolution | 137*177 | Not 137*177 | 137*177 | Not 137*177 | 137*177 | Not 137*177 | 137*177 | Not 137*177 |
| Output | Photo uploaded | Error message resolution mismatch | Error message size mismatch | Error message size and resolution mismatch | Error message for format mismatch | Error message format and resolution mismatch | Error message for format and size mismatch | Error message for format, size, and resolution mismatch |

# Error Guessing



# State Transition

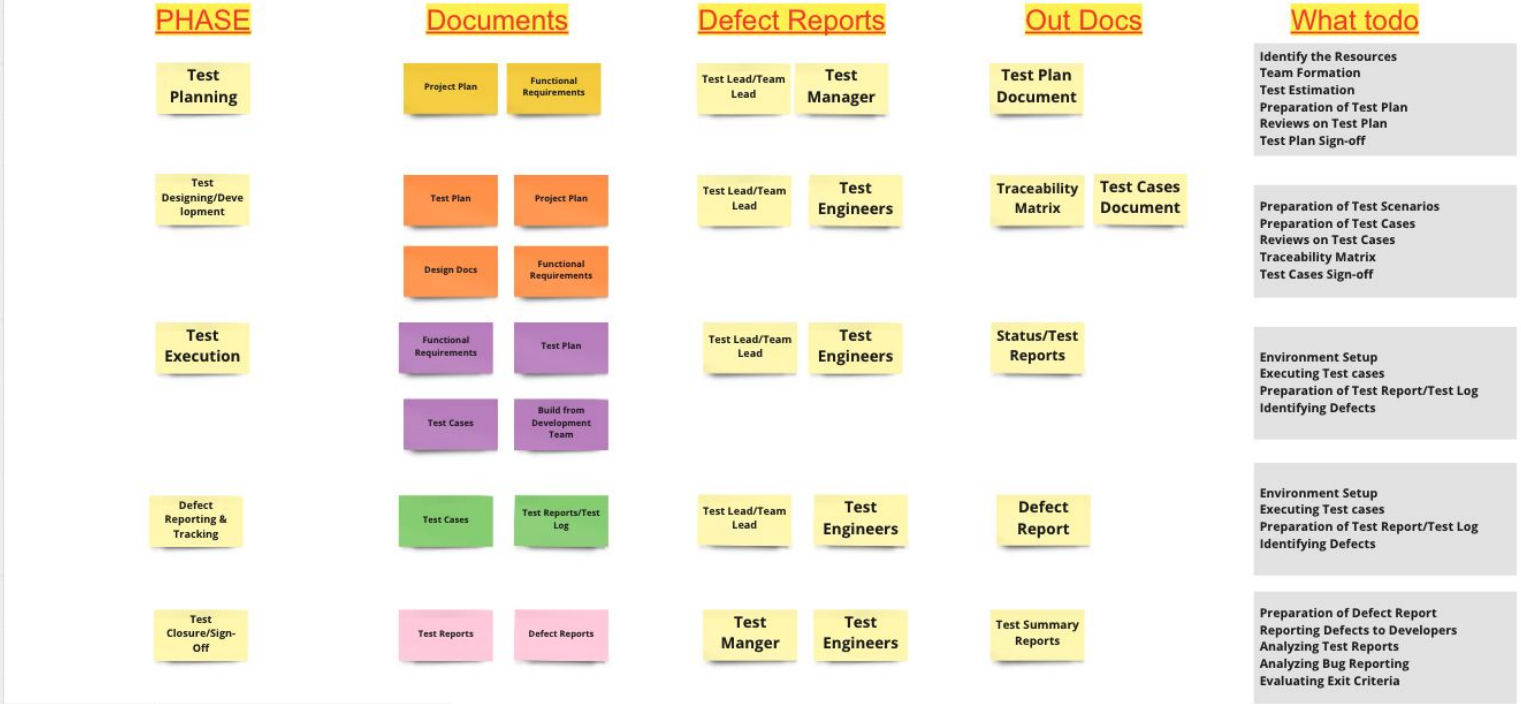
Take an example of login page of an application which locks the user name after three wrong attempts of password.



# Verification vs Validation

| SOFTWARE VERIFICATION | SOFTWARE VALIDATION |
|---|---|
| 1. It means, "Are we building any System/Product in a right manner?" | 1. It means, "Are we building right System/Product ?". |
| 2. Software Verification is the process of evaluating System/Products in the development phase to find whether they meet the specified requirements or not. | 2. Software Validation is the process of evaluating software at the end of development process in order to determine whether software Product/System meets the customer expectations and requirements or not. |
| 3. Software Verification process involves :<br>a) Reviews.<br>b) Meetings.<br>c) Inspections. | 3. Software Validation involves :<br>a) Black Box Testing.<br>b) White Box Testing.<br>c) Grey Box Testing. |
| 4. Verification of software is carried out by quality assurance team. | 4. Validation of software is carried out by the testing team. |
| 5. Execution of code is not done in case of Software Verification and is carried out before Validation process. | 5. Code is executed in case of Software Validation and it is carried out after the Software Verification process. |

# STLC (Software Testing Life Cycle)

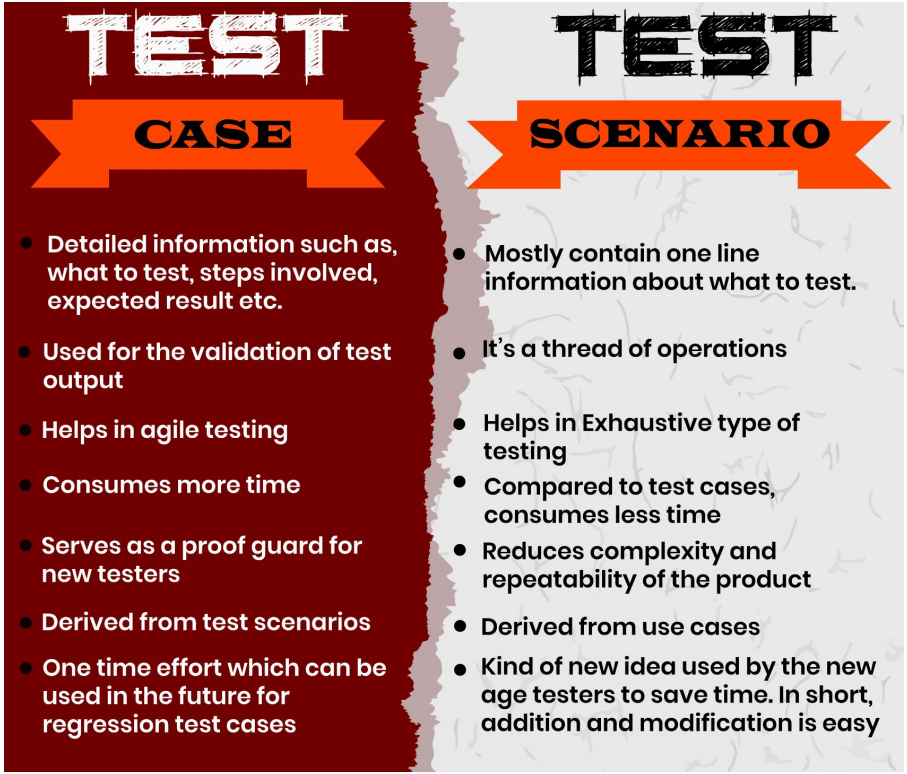


# Test Plan

A Test Plan is a document that describes a software product's test scope, test strategy, objectives, schedule, deliverables, and the resources that are needed to test it.

Overview
- Scope
- Inclusions
- Test Environments
- Exclusions
- Test Strategy
- Defect Reporting Procedure
- Roles/Responsibilities
- Test Schedule
- Test Deliverables
- Entry and Exit Criteria
- Suspension and Resumption Criteria
- Tools
- Risks and Mitigations
- Approvals

# Test Case Vs Test Scenario

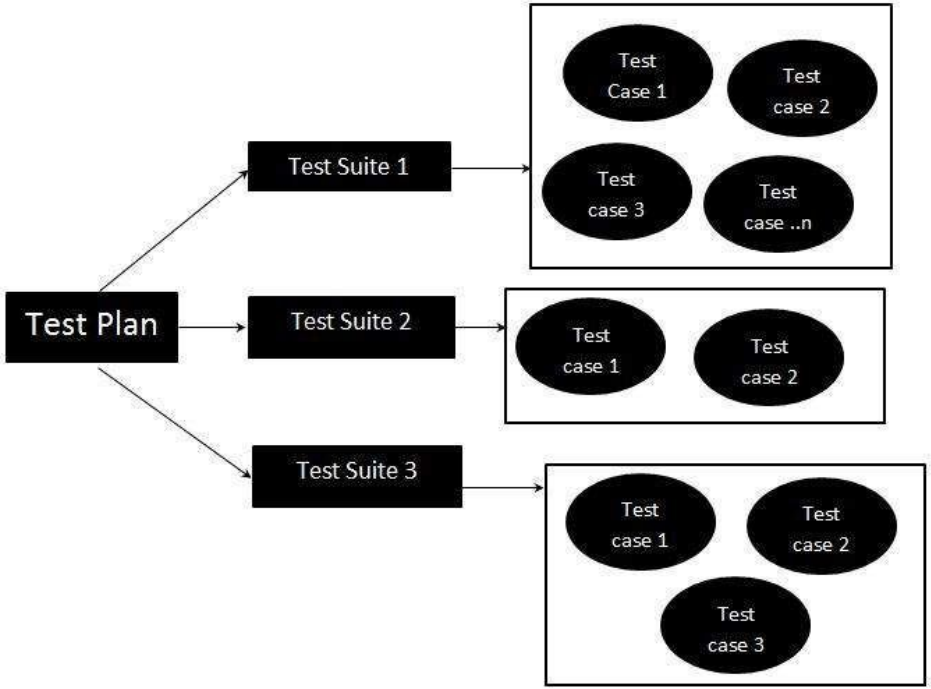


# Use Case V/s Test Case

**Use Case** – Describes functional requirement, prepared by Business Analyst(BA).

- Test Case – Describes Test Steps/ Procedure, prepared by Test Engineer.

# Test Suite


Test Plan
Test Suite 1
Test Suite 2
Test Suite 3
Test Case 1
Test case 2
Test case 3
Test case ..n
Test case 1
Test case 2
Test case 1
Test case 2
Test case 3

# Lets Create..... Test cases

# Remember these TC Columns

- Test Case ID
- Test Case Title
- Description
- Pre-condition
- Priority ( P0, P1,P2,P3) – order
- Requirement ID
- Steps/Actions
- Status
- Expected Result
- Actual Result
- Test data

# Requirements

https://bugz.atlassian.net/l/cp/1ba1G7md

## AB Testing Tool with Login and Dashboard

Created by Pro Mode
Yesterday at 5:30 PM • 2 min read

AB testing is a method of comparing two versions of a web page, landing page, or email message to determine which version performs better. It is a key tool in conversion optimization. The most common form of AB testing is the A/B test, which compares two versions of a page and measures which version performs better.

This can be accomplished by displaying the same page twice, each time with a different version of the content.

# Requirements

Go to this https://awesomeqa.com/ui/

Registration

TC - https://sdet.live/tc-template

Acceptance Criteria - User Able to register

# Designing

https://www.figma.com/file/IjMAQ772fjgrZy3wyfYIjB/Docushack?node-id=0%3A1

# Create Test Cases for Login VWO.com

https://sdet.live/tc-template

# What is Requirement Traceability Matrix ?

A document that demonstrates the relationship between requirements and other artifacts

| Req No | Req Desc | Testcase ID | Status |
|---|---|---|---|
| 123 | Login to the application | TC01,TC02,TC03 | TC01-Pass<br>TC02-Pass |
| 345 | Ticket Creation | TC04,TC05,TC06,<br>TC07,TC08,TC09<br>TC010 | TC04-Pass<br>TC05-Pass<br>TC06-Pass<br>TC06-Fail<br>TC07-No Run |
| 456 | Search Ticket | TC011,TC012,<br>TC013,TC014 | TC011-Pass<br>TC012-Fail<br>TC013-Pass<br>TC014-No Run |

# Test Environment

Test Environment is a platform specially build for test case execution on the software product.

- It is created by integrating the required software and hardware along with proper network

configurations.

- Test environment simulates production/real time environment.

- Another name of test environment is Test Bed.

# Test Execution

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

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

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

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

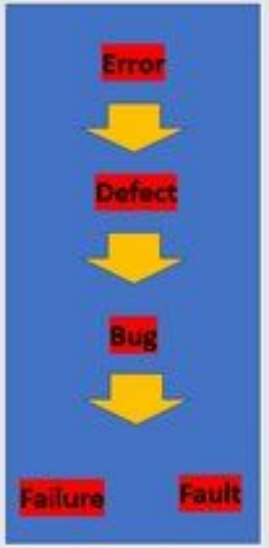
# Bug, Defect and Error

- **Error** is the **mistake** in coding done by developer
- Bug is **something which testers identify due to error in code**
- **Defect** is the **variation from the requirement**

# Failure

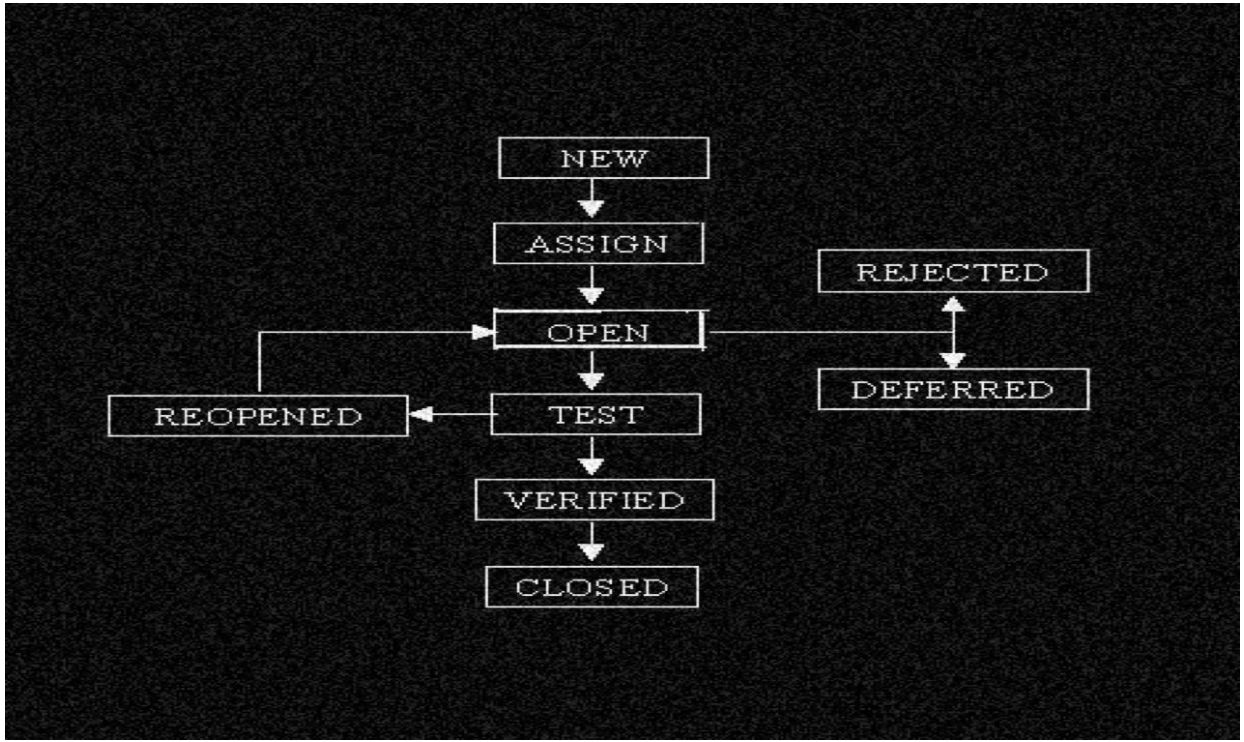
- A defect that reaches a customer.

- Error: A mistake in coding is called Error.
- Defect: Error found by tester is called Defect. The variation between the actual results and expected results is known as defect.
- Bug: Defect accepted by development team then it is called Bug / Anomaly
- Failure: When a defect reaches the end customer it is called a Failure.
- Missing and Wrong: A requirement of the customer that was not fulfilled.
- FAULT: A fault makes an application behave in a wrong manner.

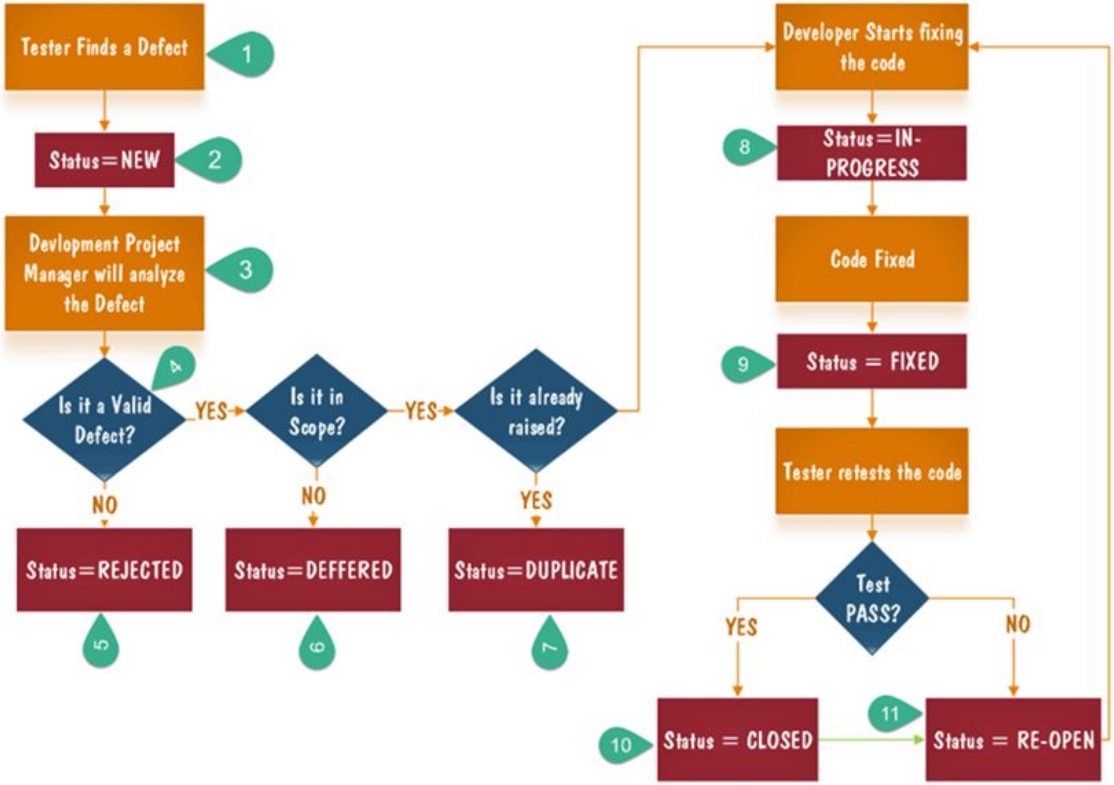



5 Shades of a Defect

# Bug Life Cycle



# Bug Life Cycle



# Bug Reporting Tools

18 Most Popular Bug Tracking Software

- #1) Backlog
- #2) Katalon TestOps
- #3) Kualitee
- #4) QACoverage
- #5) BugHerd
- #6) Userback
- #7) Marker.io
- #8) Bugzilla
- #9) JIRA
- #10) Mantis
- #11) Trac
- #12) Redmine
- #13) Micro Focus ALM/Quality Center
- #14) FogBugz
- #15) IBM Rational ClearQuest
- #16) Lighthouse
- #17) Zoho Bug Tracker
- #18) The Bug Genie
- #19) BugHost

https://www.softwaretestinghelp.com/popular-bug-tracking-software/

# Bug Reporting Tools

https://bugzilla.mozilla.org/show_bug.cgi?id=322608

https://www.softwaretestinghelp.com/popular-bug-tracking-software/


Jira
Mantis
Trac
Defect/Bug
tracking tool
backlog
Backlog
Redmine
Bugnet
Bugzilla

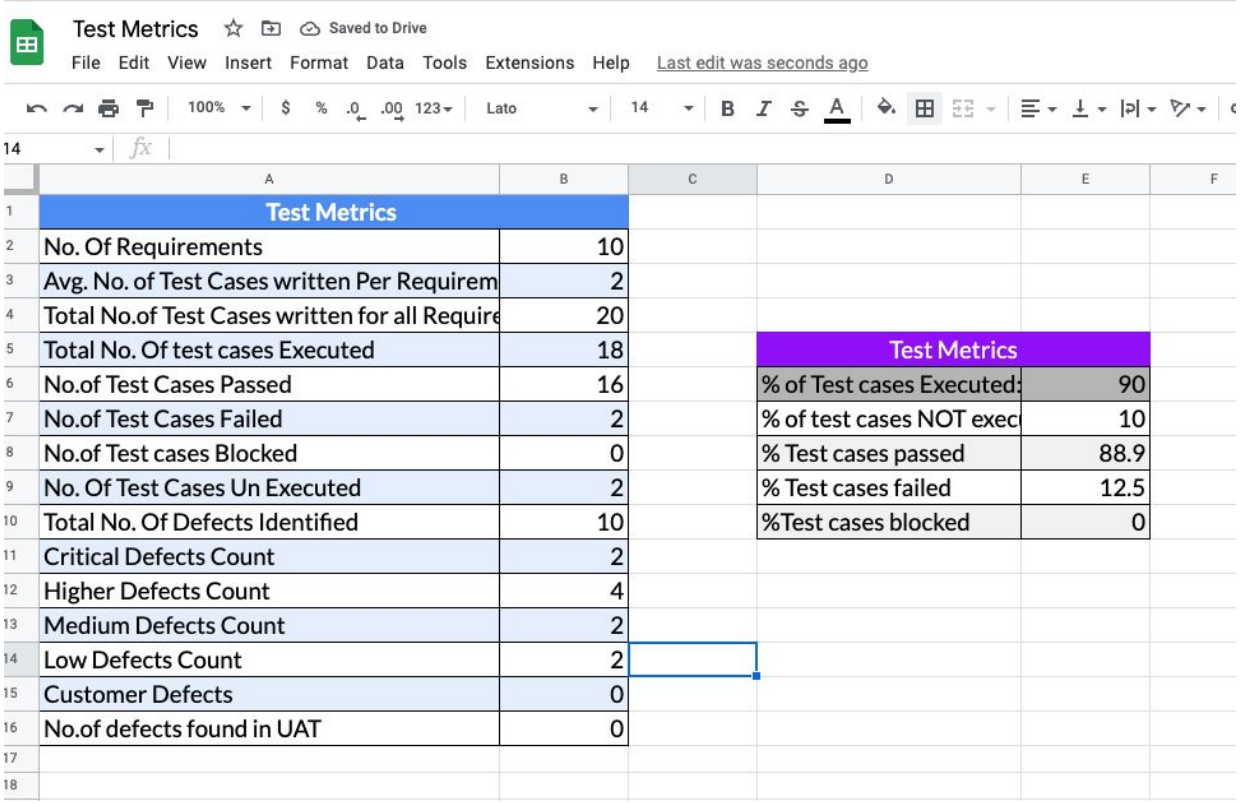
# Test Cycle Closure

**Activities:** Look at the criteria for cycle completion based on time, test coverage, cost, software, critical business goals, and quality.

Using the above parameters, **make test metrics.**
– Write down what you learned from the project. – Get ready. Review of the test
– Give the customer both a qualitative and a quantitative report on the quality of the work.
– Analyze the test results to find out how the bugs are spread out by type and severity.

**Deliverables** – Test Closure report – Test metrics

# Test Metrics




| Test Metrics | |
|---|---|
| No. Of Requirements | 10 |
| Avg. No. of Test Cases written Per Requirem | 2 |
| Total No.of Test Cases written for all Require | 20 |
| Total No. Of test cases Executed | 18 |
| No.of Test Cases Passed | 16 |
| No.of Test Cases Failed | 2 |
| No.of Test cases Blocked | 0 |
| No. Of Test Cases Un Executed | 2 |
| Total No. Of Defects Identified | 10 |
| Critical Defects Count | 2 |
| Higher Defects Count | 4 |
| Medium Defects Count | 2 |
| Low Defects Count | 2 |
| Customer Defects | 0 |
| No.of defects found in UAT | 0 |

| Test Metrics | |
|---|---|
| % of Test cases Executed: | 90 |
| % of test cases NOT exec | 10 |
| % Test cases passed | 88.9 |
| % Test cases failed | 12.5 |
| %Test cases blocked | 0 |

# Test Metrics

**Defect Density**: Number of defects identified per requirement/s
No.of defects found / Size(No. of requirements)

**Defect Removal Efficiency** (DRE):
(A / A+B ) * 100
(Fixed Defects / (Fixed Defects + Missed defects) ) * 100

- A- Defects identified during testing/ Fixed Defects
- B- Defects identified by the customer/Missed defects

**Defect Leakage:**
(No.of defects found in UAT / No. of defects found in Testing) * 100

**Defect Rejection Ratio:**
(No. of defect rejected /Total No. of defects raised) * 100
**Defect Age**: Fixed date-Reported date
**Customer satisfaction** = No.of complaints per Period of time

# What you will be doing?

- Understanding what the application needs and how it works is important.
- Figuring out what test scenarios are needed.
- Creating Test Cases to make sure the application works.
- Creating a place to test (Test Bed)
- Test cases should be run on a valid application.
- Write down the results of the tests (how many test cases pass/fail).
- Reporting and following up on problems.
- The problems with the last build have been fixed.
- Do different kinds of testing in the application.
- Reports to the Test Lead on how the tasks they were given are going
- Took part in regular meetings with the team.
- Making scripts for automation.
- Gives advice on whether or not the application or system is ready for production.

adsda

# 7 principles of software testing

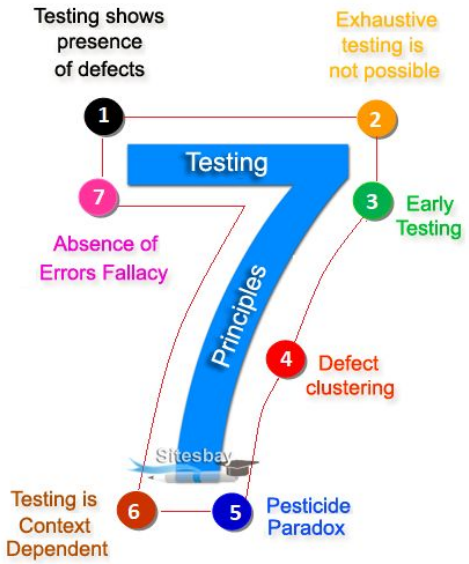



Fig: Software Testing Principles

https://www.softwaretestinghelp.com/popular-bug-tracking-software/

# 7 principles of software testing

| | |
|---|---|
| **Testing shows the presence of defects** | *We can never say that our system is defect free”* |
| **Exhaustive testing is impossible** | *“Impossible to test everything”* |
| **Early testing** | *Early means Cheap* |
| **Defect clustering** | *“Small number of modules contain most of the defects”* |

https://www.softwaretestinghelp.com/popular-bug-tracking-software/

# 7 principles of software testing

**Pesticide Paradox**

*“To detect more defects, we need change the test data”*

**Testing is context dependent**

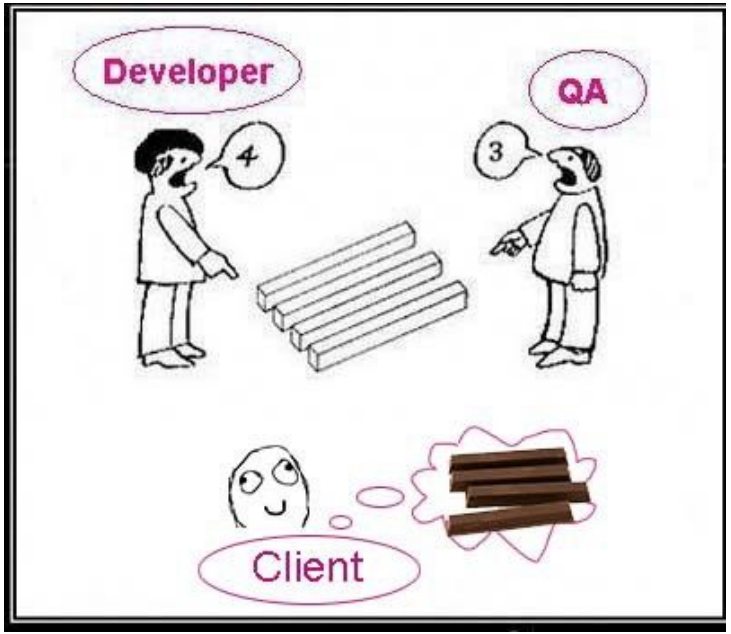
*“Tests are done differently in a different context”*

**Absence of Error Fallacy**

*“focus on the business need”*

https://www.softwaretestinghelp.com/popular-bug-tracking-software/

# 7 principles of software testing




https://www.softwaretestinghelp.com/popular-bug-tracking-software/

# Severity vs Priority

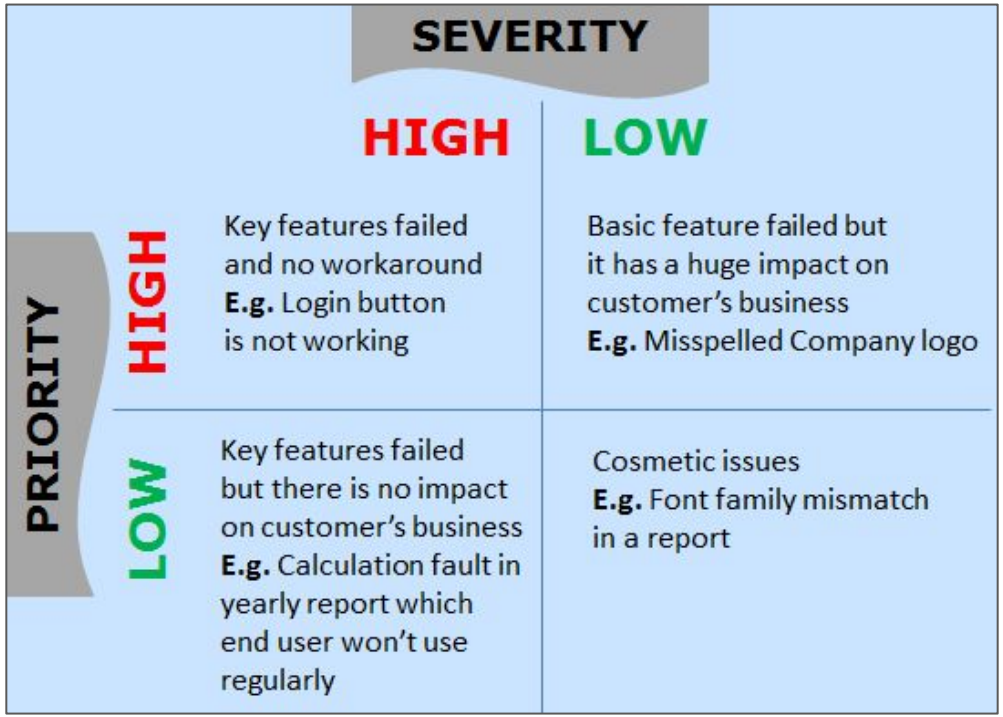
Priority is defined as the order in which a defect should be fixed
How quickly bug should be removed from system
Low, Medium High.

Severity is degree of impact a bug.
It defines the effect the bug has on the system
Minor, Major, Critical

- Severity is about How bad Bugs are
- Priority is about how soon it should be fixed

Severity=Impact **QA**
Priority=Urgent/Urgency

# Severity vs Priority



# Apply Black Box Techniques

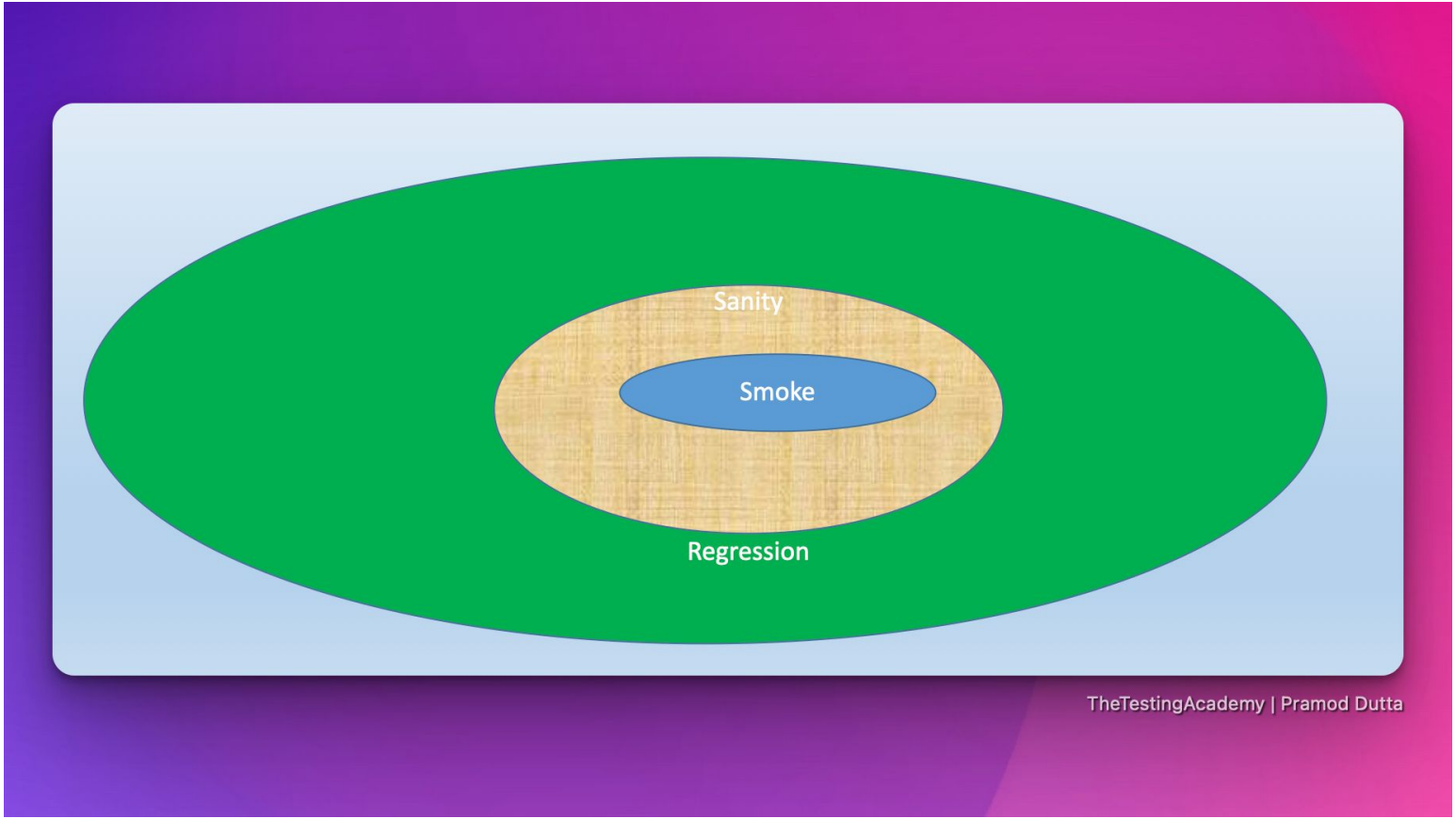
# Smoke Sanity and Regression

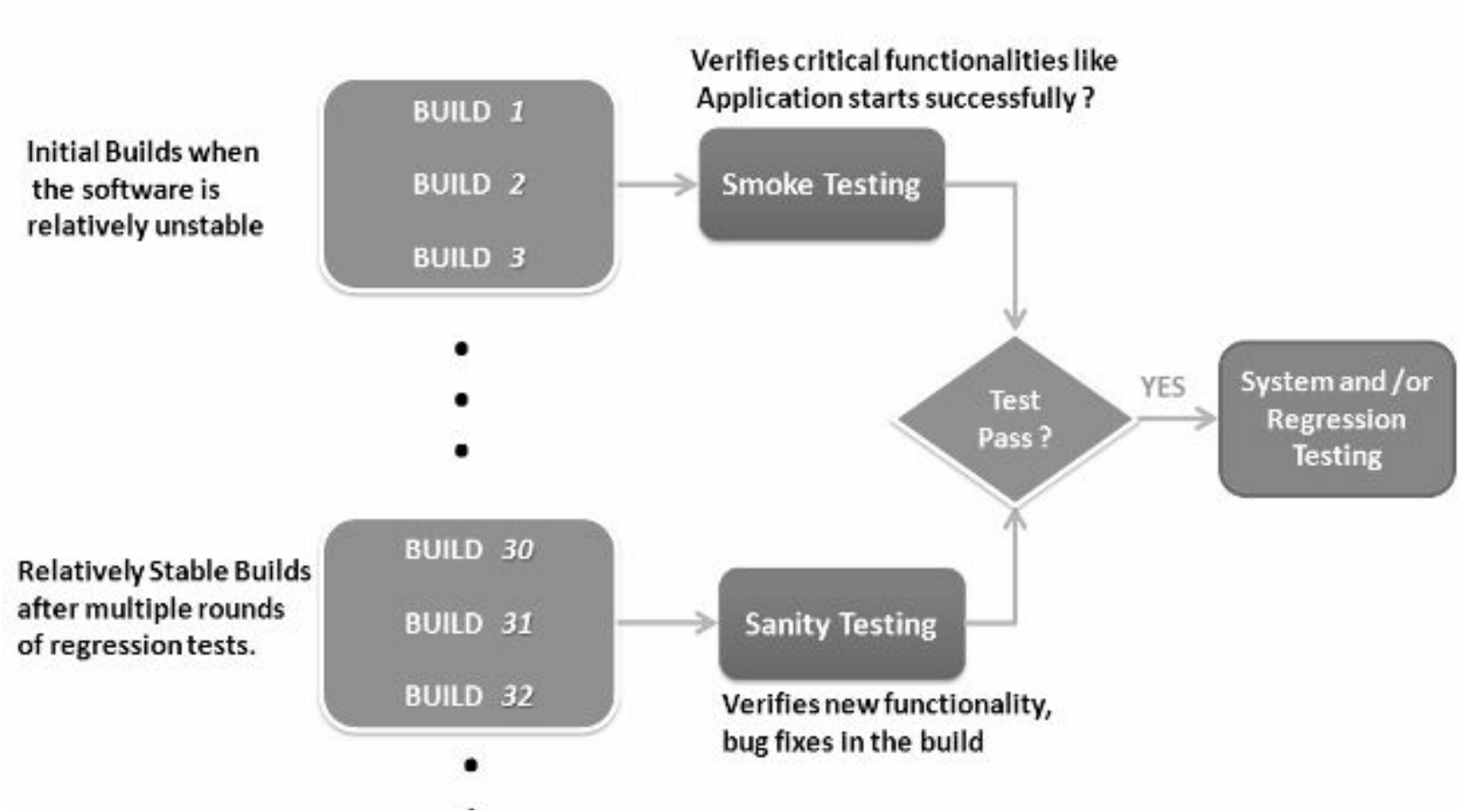
**Smoke Testing** is executed before any detailed functional tests are done on the software.

Smoke testing is also called as Build Verification Test.

**Sanity testing** is a software testing technique which does a quick evaluation of the quality of the software release to determine major functionality is working fine or not

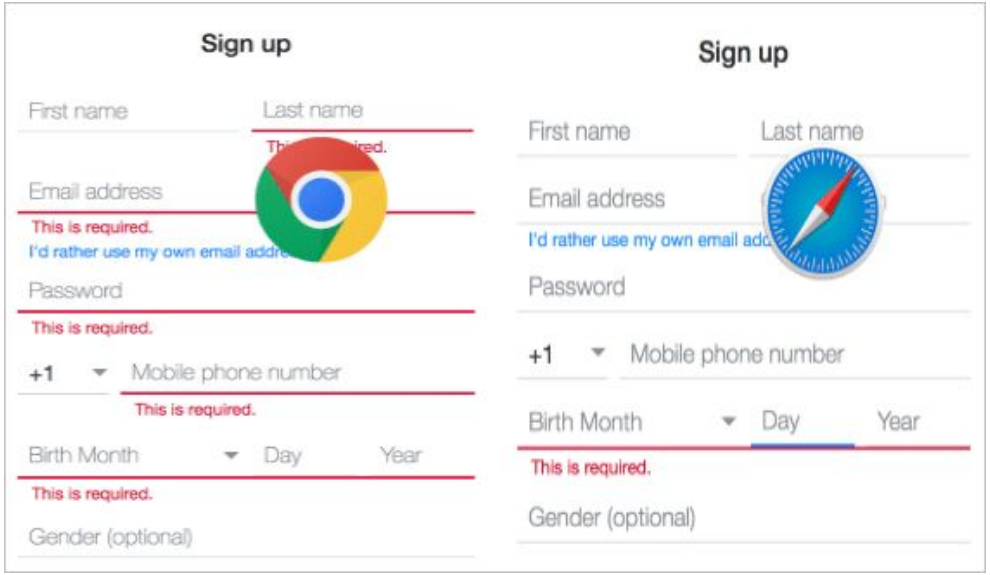
**Regression Testing** Covers detailed testing targeting all the affected areas after new functionalities are added


Sanity
Smoke
Regression
TheTestingAcademy | Pramod Dutta


Verifies critical functionalities like Application starts successfully ?
Initial Builds when the software is relatively unstable
BUILD 1
BUILD 2
BUILD 3
Smoke Testing
Test Pass ?
YES
System and /or Regression Testing
Relatively Stable Builds after multiple rounds of regression tests.
BUILD 30
BUILD 31
BUILD 32
Sanity Testing
Verifies new functionality, bug fixes in the build

# Cross Browser Testing

Cross-browser testing to test your website or application in multiple browsers- and making sure that it works consistently and as in intended without any dependencies



# Demo Cross Browser Testing

BrowserStack

# Agenda

- Agile.
- Scrum Framework
- JIRA Demo
- Test Management Tool Zephyr Demo
- Web Fundamentals

# Commitment!
# Block at least
# 4 hour Per Week.

# Agile

Ability to Move easily and quickly.

**Agile is the ability to create and respond to change.**

# Agile

It is a way of dealing with, and ultimately succeeding in, an uncertain and turbulent environment.

# Agile software development

Agile software development is an **umbrella term** for a set of **frameworks** and **practices** based on the values and principles expressed in the Manifesto for Agile Software Development and the 12 Principles behind it.

https://www.agilealliance.org/agile101/

# Agile software development

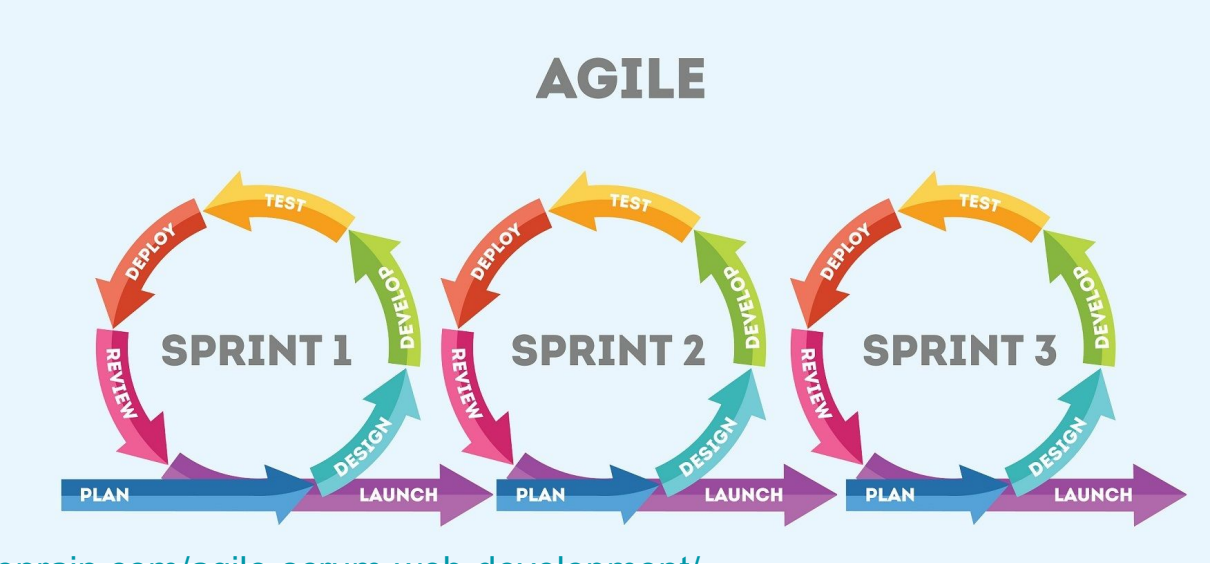
Agile software development is more than frameworks such as **Scrum**, Kanban, Extreme Programming, or Feature-Driven Development (FDD).

Agile software development is more than practices such as pair programming, **test-driven development,** stand-ups, planning sessions, and sprints.

https://www.agilealliance.org/agile101/

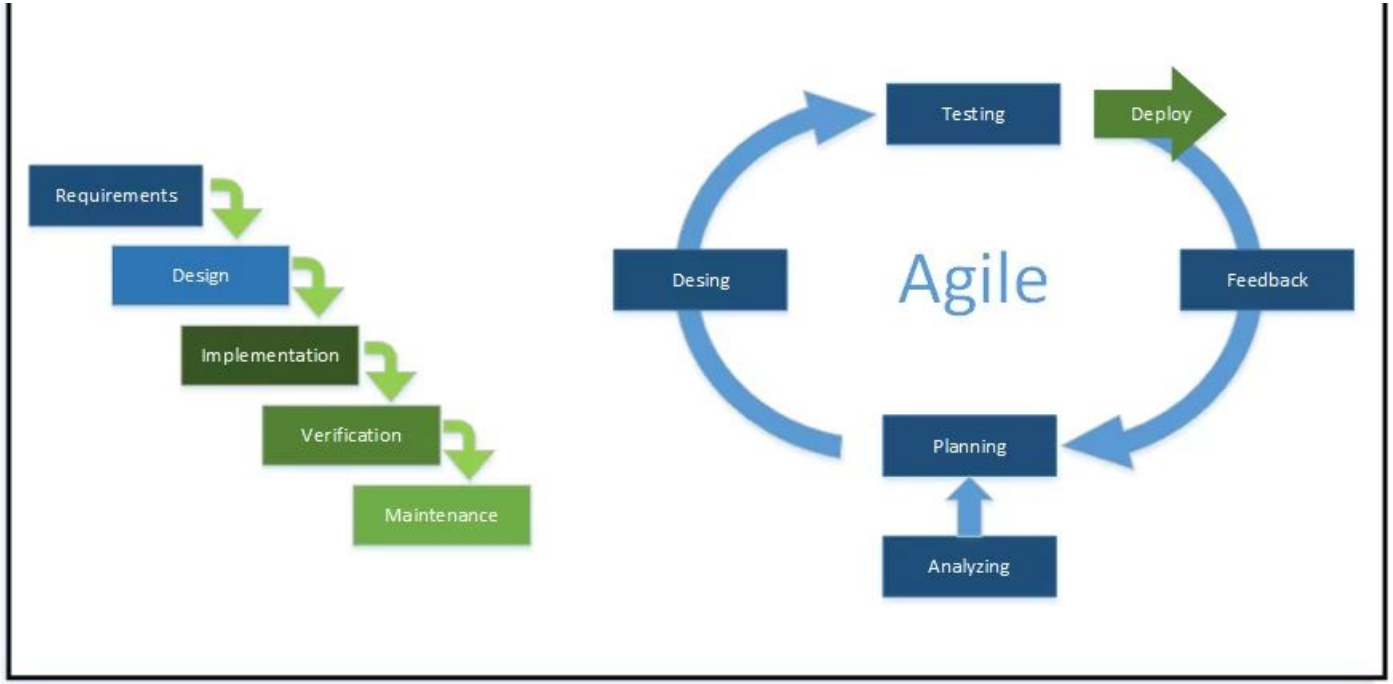
# Agile

Agile is an **iterative approach** to project management and **software development** that helps teams deliver value to their customers faster and with fewer headaches




https://www.neonrain.com/agile-scrum-web-development/

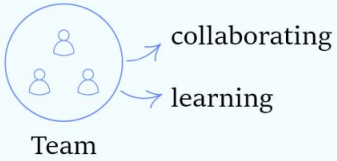
# Waterfall vs Agile

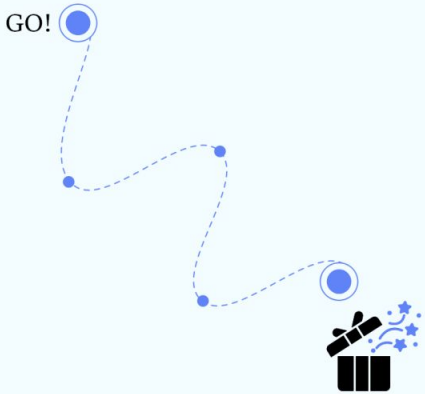


# Agile vs Waterfall

## Agile

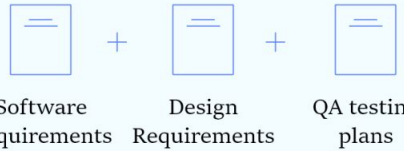
Focus on working product (doing)




## Waterfall

Focus on documenration
& detailed plans

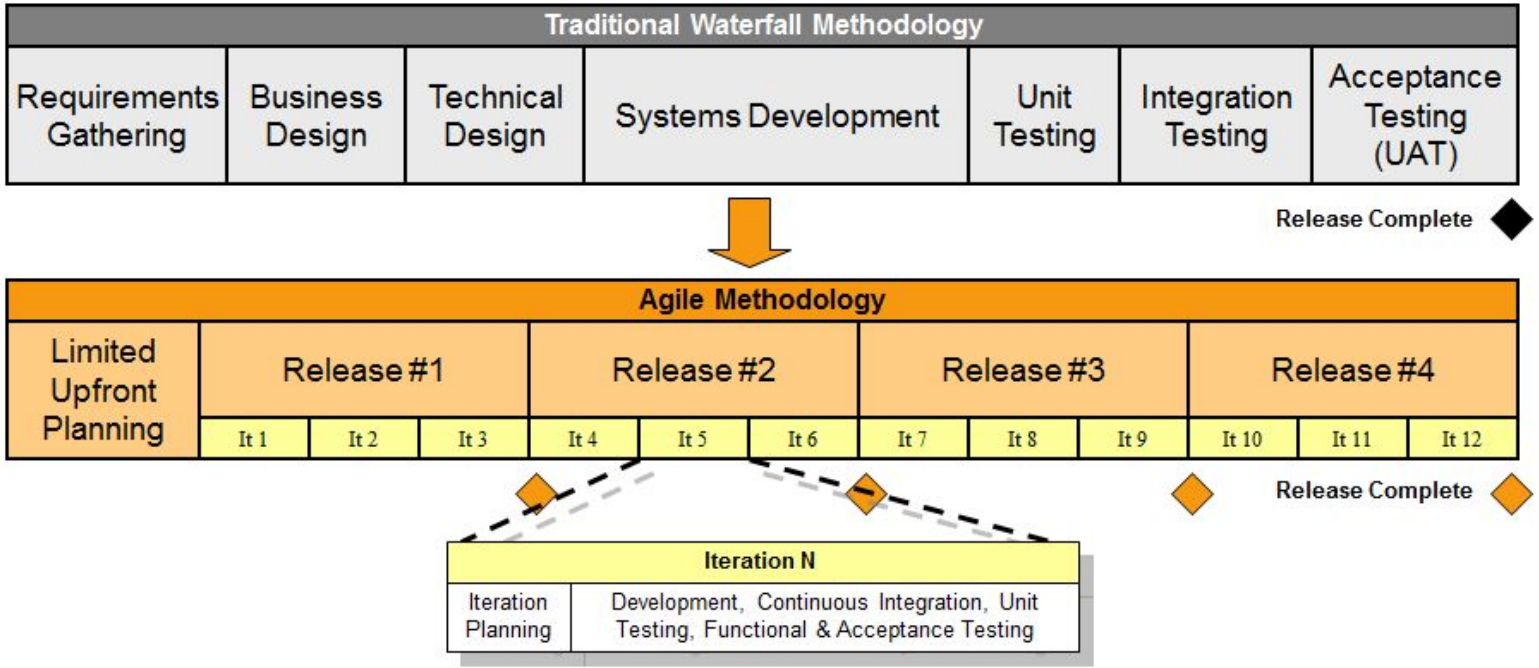



# Waterfall vs Agile

- Waterfall is a **Linear Sequential Life Cycle Model** whereas Agile is a **continuous iteration of development** and testing in the software development process.
- In Agile vs Waterfall difference, the Agile methodology is known for its flexibility whereas Waterfall is a structured software development methodology.
- Agile performs **testing concurrently with software development** whereas in Waterfall methodology testing comes after the “Build” phase.
- Agile allows changes in project development requirement whereas Waterfall has no scope of changing the requirements once the project development starts.

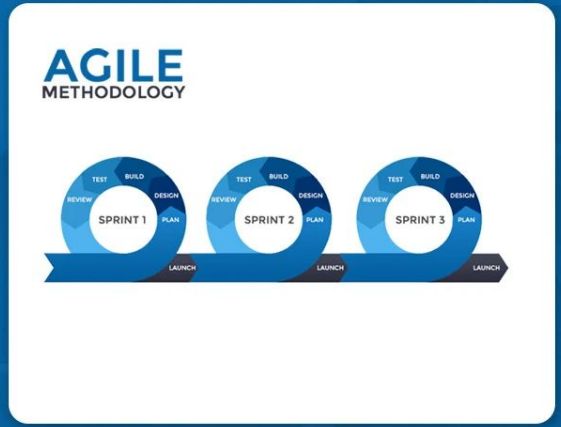
# Waterfall vs Agile

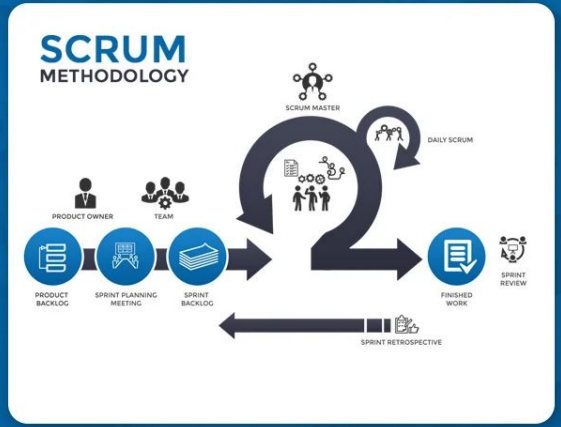
- You can’t go back
- Freeze is Freeze

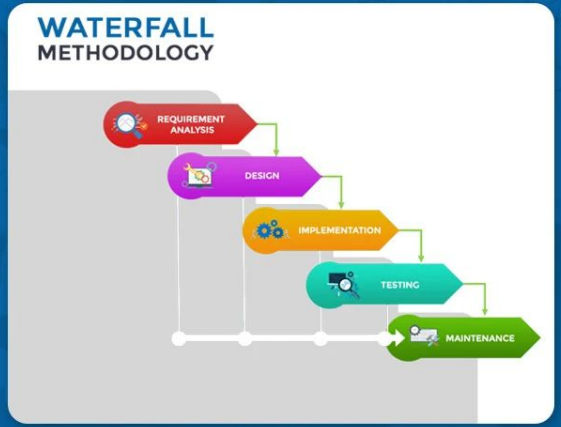



https://www.agilealliance.org/agile101/

# A COMPREHENSIVE COMPARISON BETWEEN THE AGILE, SCRUM, *and* WATERFALL METHODOLOGY



# 12 agile principles

## in software development

Customer satisfactions

Changing requirements

Frequent delivery

Communicate regularly

Support team member

Face-to-face communication

Measure work progress

Development process

Good design

Measure progress

Continue seeking result

Reflect and adjust regularly

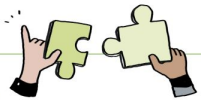
Agile Manifesto

## Agile Values

We are uncovering **better ways of developing software by doing it and helping others do it.** Through this work we have come to value:


Individuals and interactions over processes and tools

Working software over comprehensive documentation



Customer collaboration over contract negotiation

Responding to change over following a plan

# Agile Manifesto

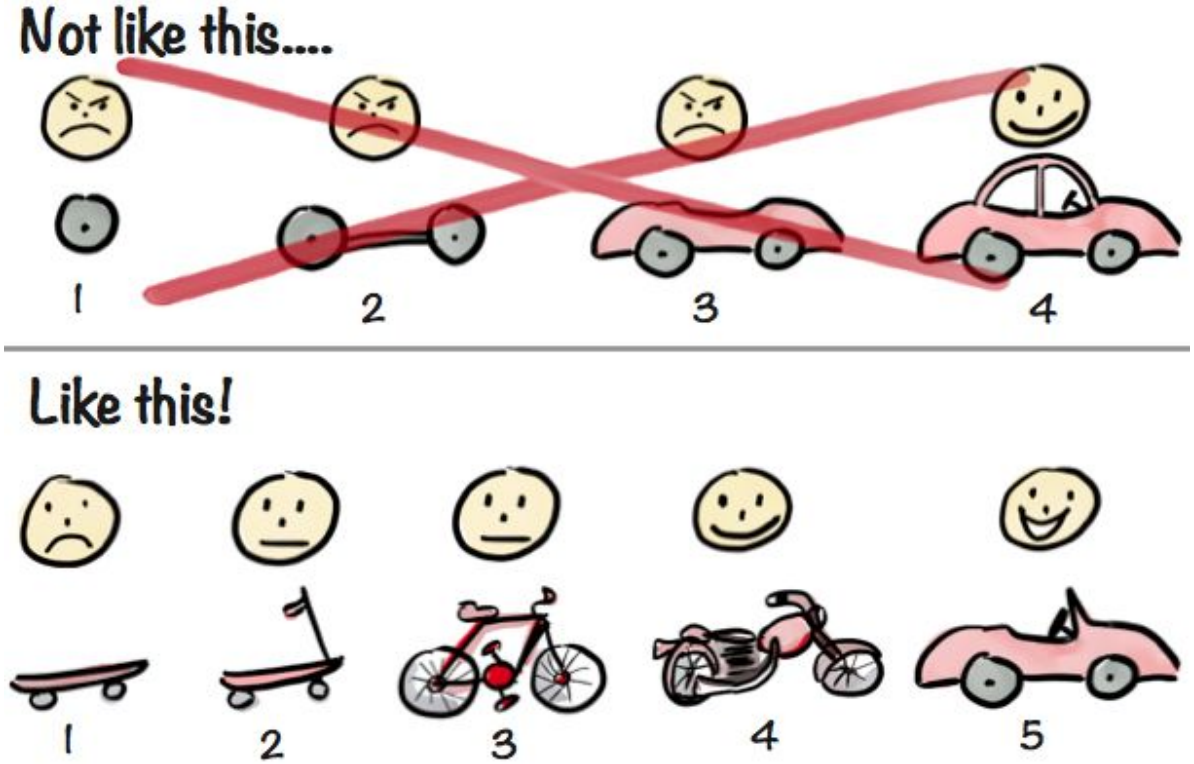
# Agile Pros

- Gives **flexibility to developers.**
- Realistic approach to software development.
- **Encourage teamwork** and cross training.
- Functionality can be developed rapidly and demonstrated.
- **Resource requirements are minimum.**
- Suitable for fixed or changing requirements
- **Delivers early partial working solutions**.
- Good model for **environments that change steadily.**
- **Minimal rules**, documentation easily employed.
- Enables concurrent development and delivery within an overall planned context.
- Little or no planning required.
- Easy to manage.

# Agile Cons

- More **risk of sustainability, maintainability and extensibility**.
- Not **suitable for handling complex dependencies**.
- **Depends heavily on customer interaction,** so if customer is not clear, team can be driven in the wrong direction.
- There is a very **high individual dependency,** since there is minimum documentation generated.
- **Transfer of technology** to new team members may be **quite challenging** due to lack of documentation.

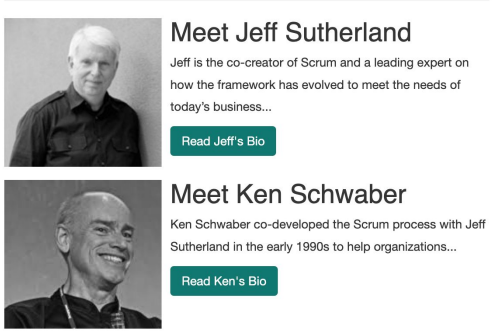
# MVP



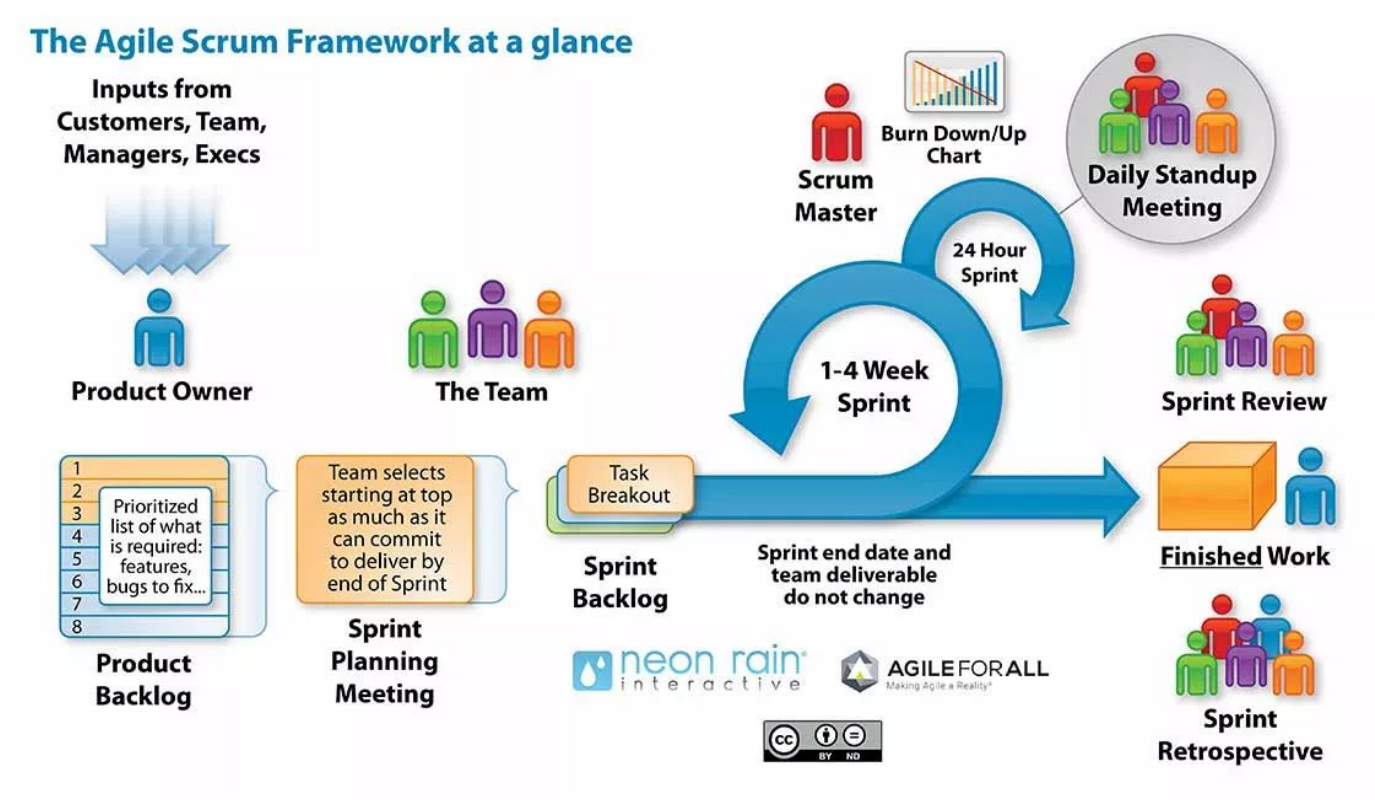
# Waterfall vs Scrum vs Kanban

# Scrum Framework

Scrum is a framework for developing and sustaining complex products.



# Scrum Framework



# Scrum Framework

Scrum requires a Scrum Master to foster an environment where:

- A Product Owner orders the work for a complex problem into a Product Backlog.
- The Scrum Team turns a selection of the work into an Increment of value during a Sprint.
- The Scrum Team and its stakeholders inspect the results and adjust for the next Sprint.
- Repeat

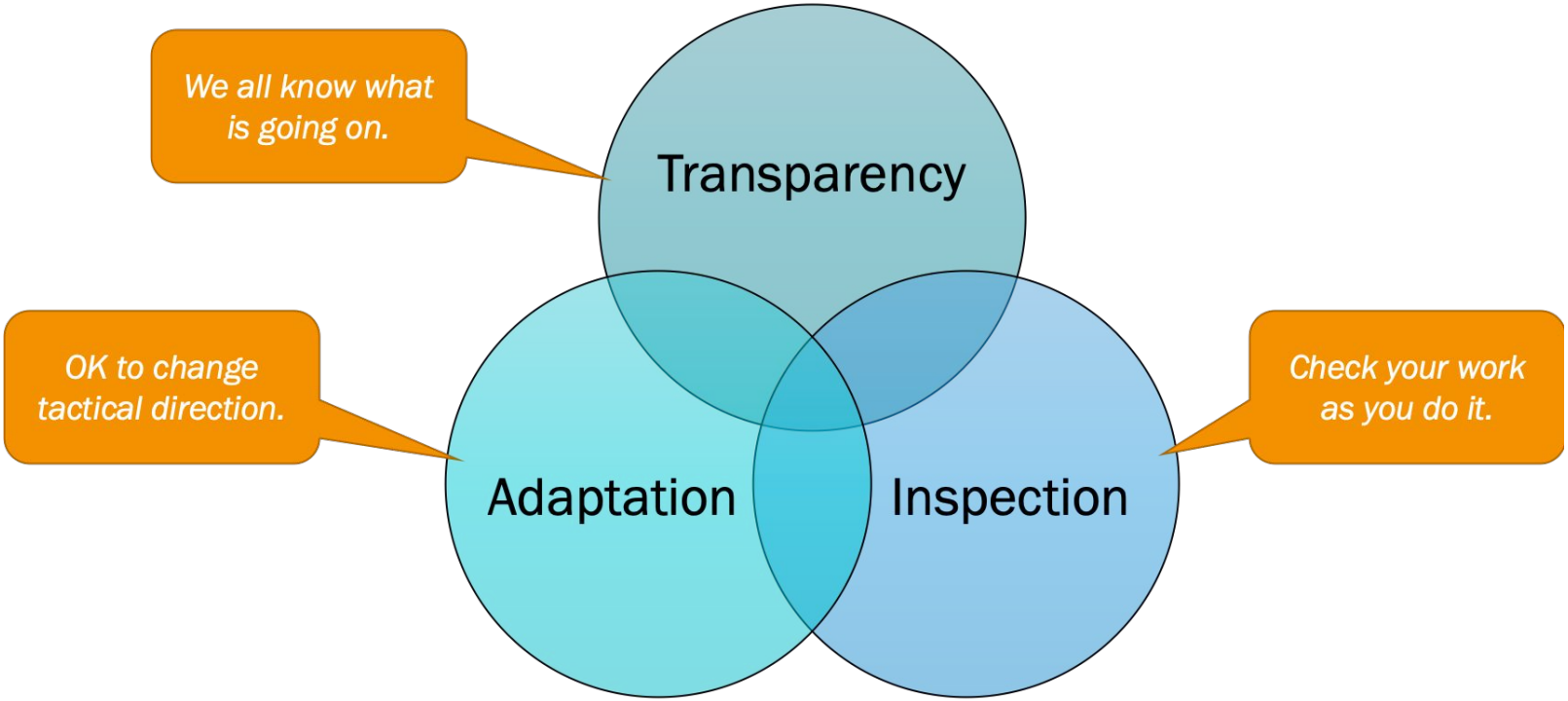
# Scrum Theory

Scrum is founded on empiricism and lean thinking.

Empiricism asserts that **knowledge comes from experience** and making decisions based on what is observed.

Lean thinking **reduces waste and focuses on the essentials.**

# Scrum Theory



# Scrum Values


## Focus

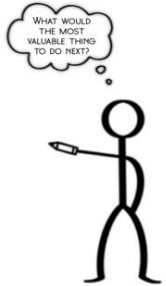
Focus on the work of the Sprint.

## Courage

Have the courage to do the right thing and to work on tough problems.

## Commitment

The Scrum Team commits to achieving its goals and to supporting each other.




## Openness

Open about the work and the challenges.

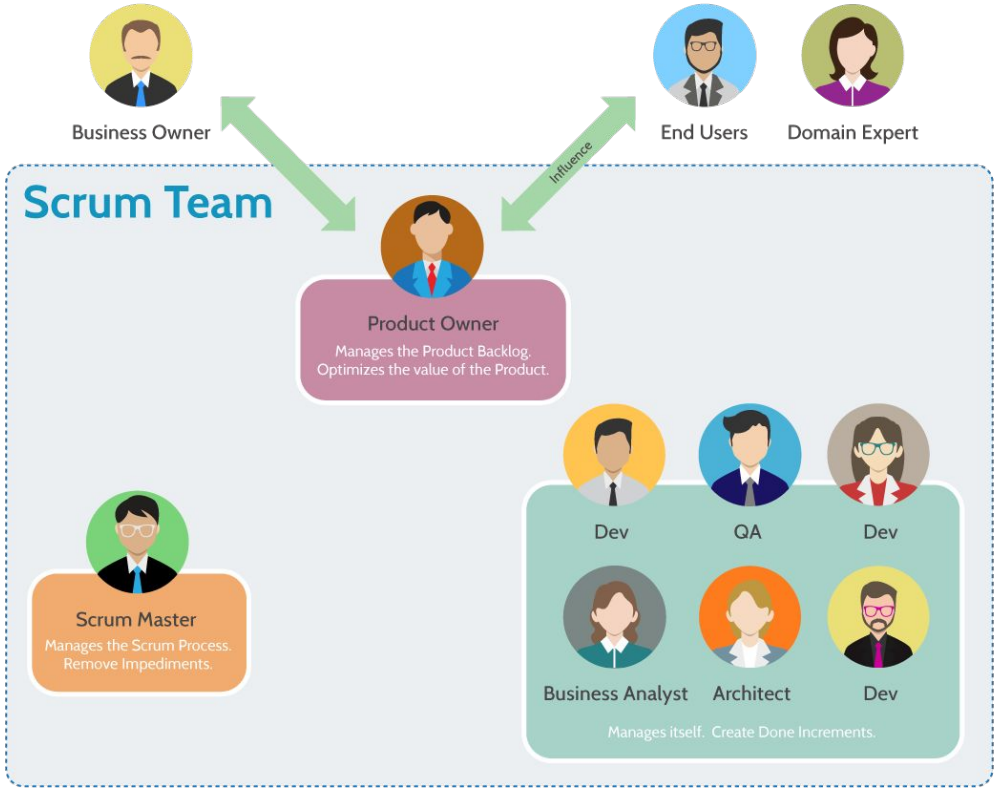
## The Scrum Values

## Respect

Respect each other to be capable, independent people.

Rebel Scrum
This is the training you've been looking for.
www.rebelscrum.site

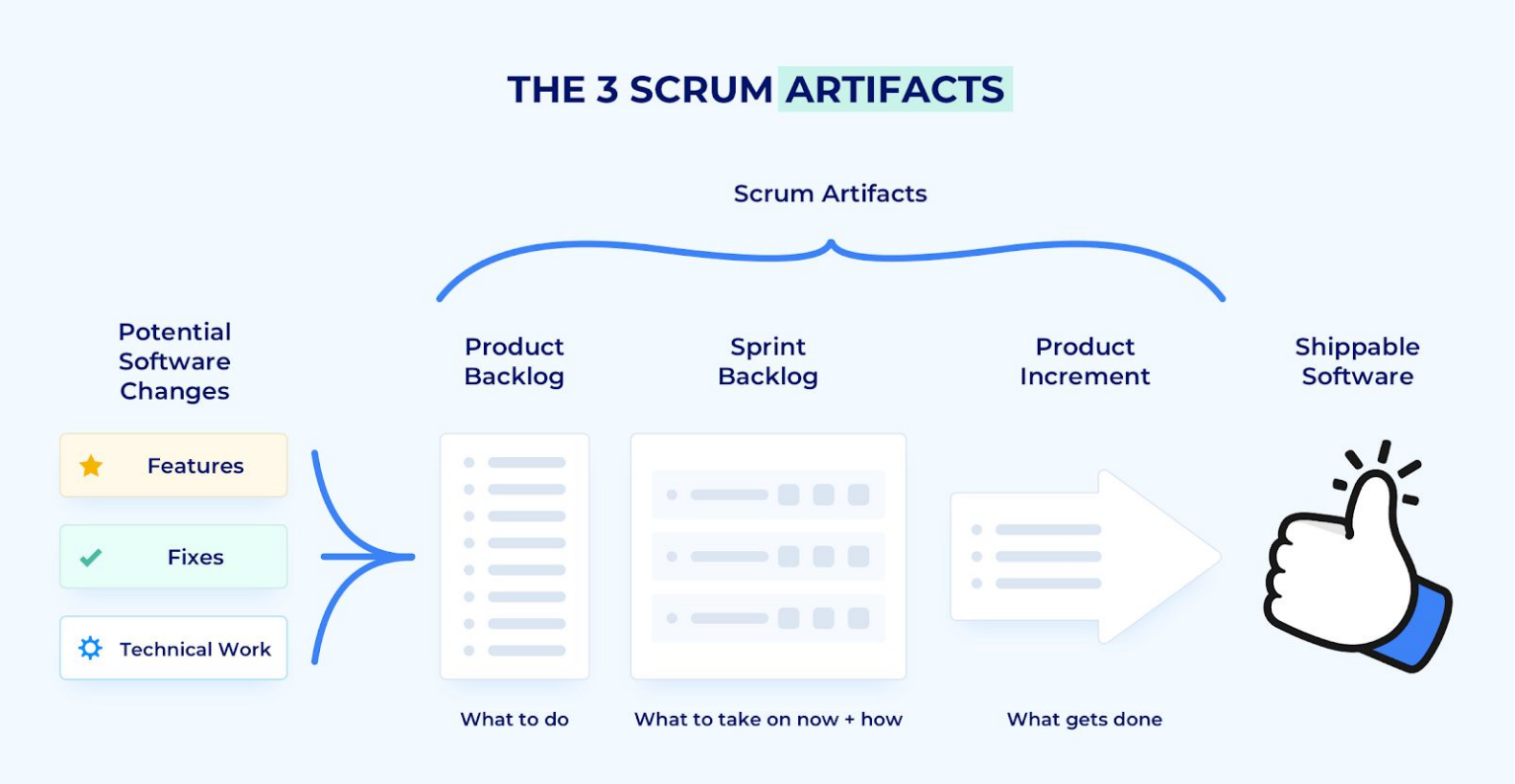
# Scrum Team


Business Owner
End Users
Domain Expert
Influence
Scrum Team
Product Owner
Manages the Product Backlog.
Optimizes the value of the Product.
Dev
QA
Dev
Scrum Master
Manages the Scrum Process.
Remove Impediments.
Business Analyst
Architect
Dev
Manages itself. Create Done Increments.

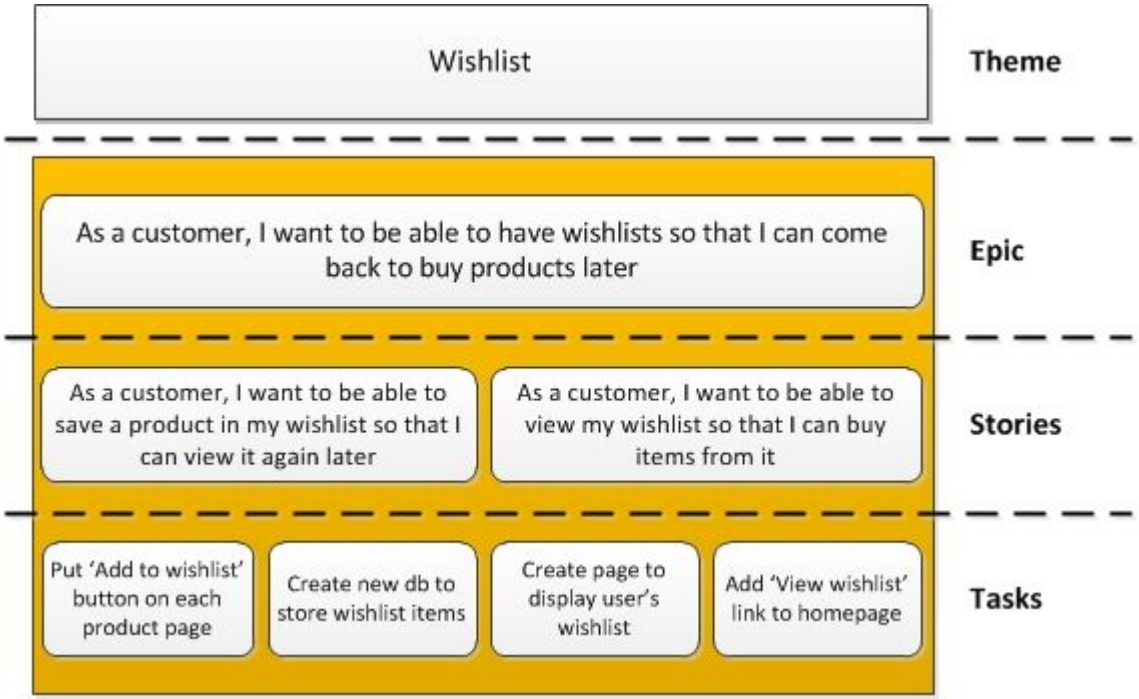
# Scrum Events

| Event | Inspection | Adaptation |
|---|---|---|
| Sprint Planning | • Product Backlog<br>• (Commitments Retrospective)<br>• (Definition of Done) | • Sprint Goal<br>• Forecast<br>• Sprint Backlog |
| Daily Scrum | • Progress toward Sprint Goal | • Sprint Backlog<br>• Daily Plan |
| Sprint Review | • Product Increment<br>• Product Backlog (Release)<br>• Market-business conditions | • Product Backlog |
| Sprint Retrospective | • Team & collaboration<br>• Technology & engineering<br>• Definition of Done | • Actionable improvements |

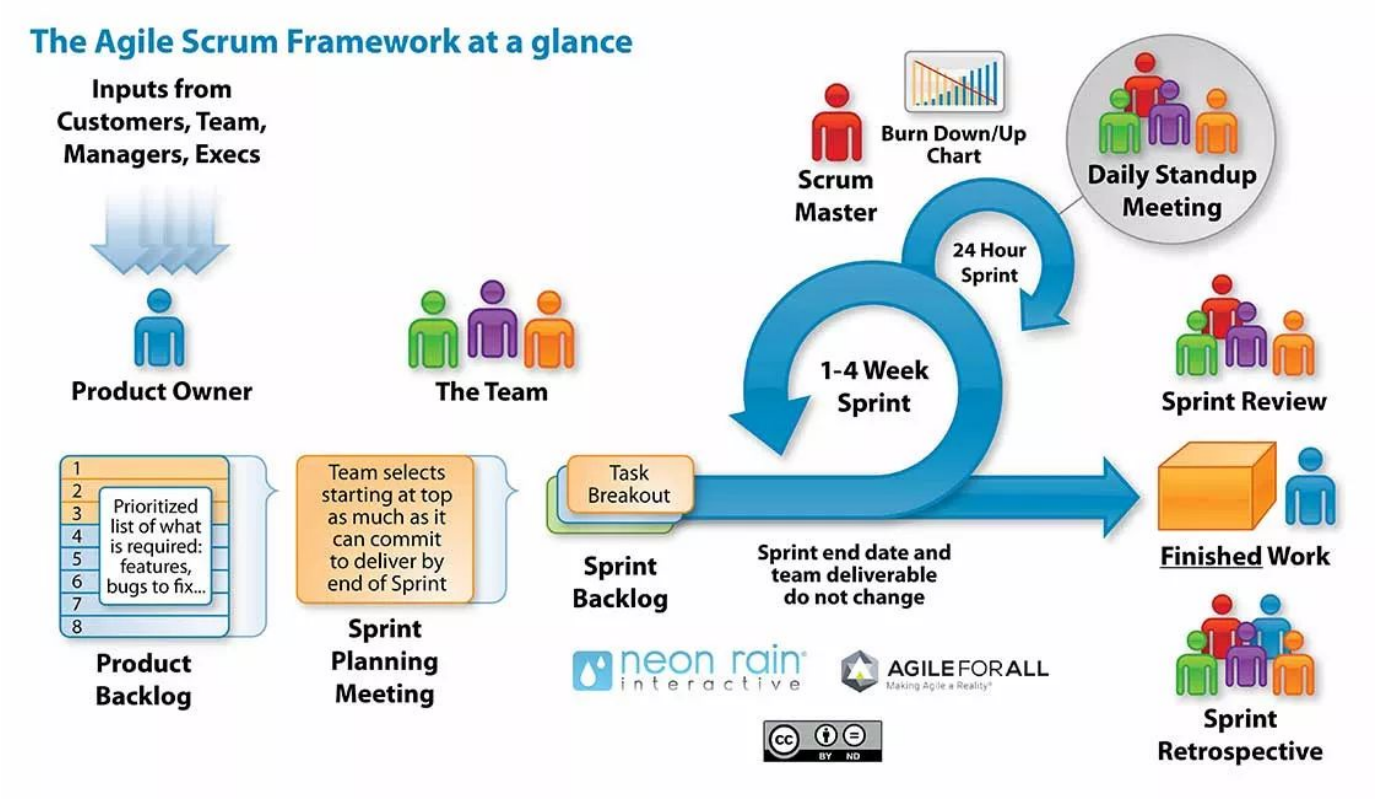
# Scrum Artifacts



# Scrum Artifacts



# Scrum Framework



# Read more

https://scrumguides.org/scrum-guide.html

# Professional Scrum Master™ I Certification

(PSM I) certification validates your knowledge of the Scrum framework, the Scrum Master accountabilities and how to apply Scrum.

## Certification Details

- $150 USD per attempt
- Passing score: 85%
- Time limit: 60 minutes
- Number of Questions: 80
- Format: Multiple Choice, Multiple Answer, True/False
- Free Credly digital credential included
- Recommended courses: Applying Professional Scrum and/or Professional Scrum Master
- Practice Assessment: Scrum Open
- Passwords have no expiration date, but are valid for one attempt only
- Lifetime certification - no annual renewal fee required

https://www.scrum.org/assessments/professional-scrum-master-i-certification

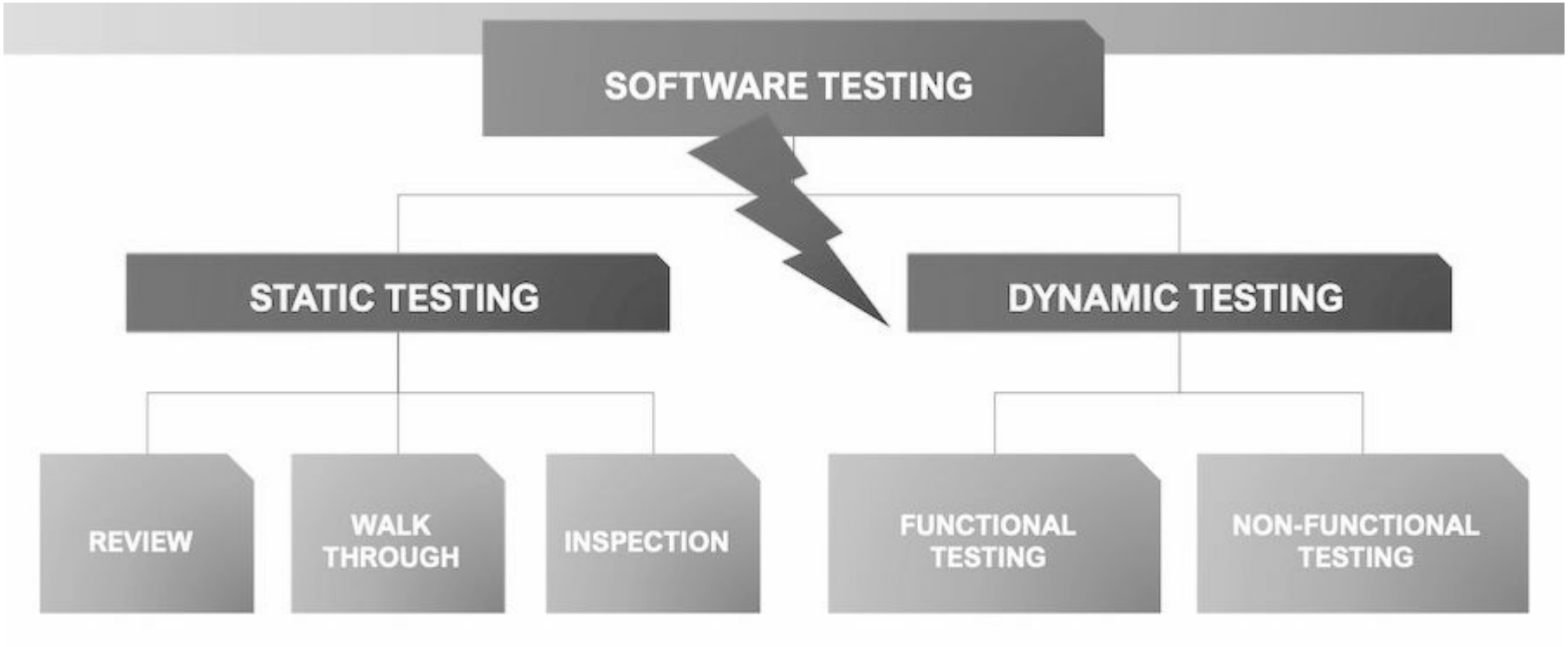
# Scrum, JIRA Demo

1. JIRA cloud managed is FREE for 10 Users
2. Create a Scrum Board
3. Add Epic and Stories
4. Add Task and Perform the activities
5. Sprint Backlog, Retro and other documents.
6. Gantt Chart
7. Sprint Burn Down Report
8. Test Management - Zephyr

https://www.scrum.org/assessments/professional-scrum-master-i-certification

# Verification vs Validation

| SOFTWARE VERIFICATION | SOFTWARE VALIDATION |
|---|---|
| 1. It means, "Are we building any System/Product in a right manner?" | 1. It means, "Are we building right System/Product ?". |
| 2. Software Verification is the process of evaluating System/Products in the development phase to find whether they meet the specified requirements or not. | 2. Software Validation is the process of evaluating software at the end of development process in order to determine whether software Product/System meets the customer expectations and requirements or not. |
| 3. Software Verification process involves :<br>a) Reviews.<br>b) Meetings.<br>c) Inspections. | 3. Software Validation involves :<br>a) Black Box Testing.<br>b) White Box Testing.<br>c) Grey Box Testing. |
| 4. Verification of software is carried out by quality assurance team. | 4. Validation of software is carried out by the testing team. |
| 5. Execution of code is not done in case of Software Verification and is carried out before Validation process. | 5. Code is executed in case of Software Validation and it is carried out after the Software Verification process. |


SOFTWARE TESTING
STATIC TESTING
DYNAMIC TESTING
REVIEW
WALK THROUGH
INSPECTION
FUNCTIONAL TESTING
NON-FUNCTIONAL TESTING

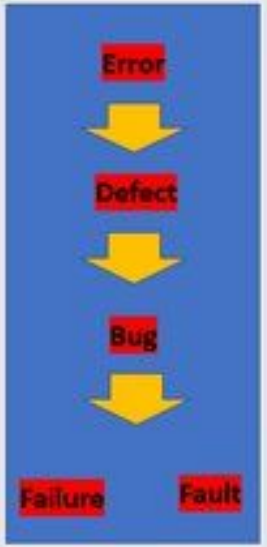
# Bug, Defect and Error

- **Error** is the **mistake** in coding done by developer
- Bug is **something which testers identify due to error in code**
- Defect is the **variation from the requirement**

# Failure

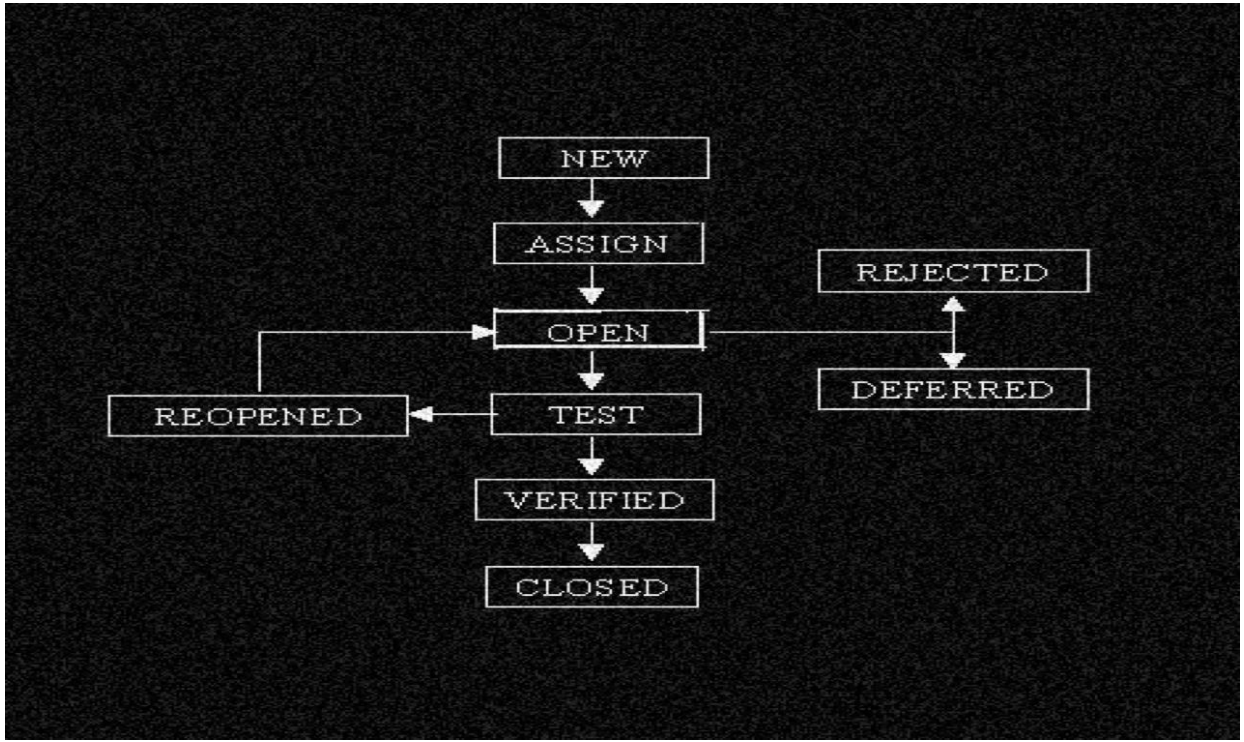
- A defect that reaches a customer.

- Error: A mistake in coding is called Error.
- Defect: Error found by tester is called Defect. The variation between the actual results and expected results is known as defect.
- Bug: Defect accepted by development team then it is called Bug / Anomaly
- Failure: When a defect reaches the end customer it is called a Failure.
- Missing and Wrong: A requirement of the customer that was not fulfilled.
- FAULT: A fault makes an application behave in a wrong manner.




5 Shades of a Defect

# Bug Life Cycle



# Bug Reporting Tools

18 Most Popular Bug Tracking Software

- #1) Backlog
- #2) Katalon TestOps
- #3) Kualitee
- #4) QACoverage
- #5) BugHerd
- #6) Userback
- #7) Marker.io
- #8) Bugzilla
- #9) JIRA
- #10) Mantis
- #11) Trac
- #12) Redmine
- #13) Micro Focus ALM/Quality Center
- #14) FogBugz
- #15) IBM Rational ClearQuest
- #16) Lighthouse
- #17) Zoho Bug Tracker
- #18) The Bug Genie
- #19) BugHost

https://www.softwaretestinghelp.com/popular-bug-tracking-software/

# Bug Reporting Tools

https://bugzilla.mozilla.org/show_bug.cgi?id=322608

https://www.softwaretestinghelp.com/popular-bug-tracking-software/

# 7 principles of software testing

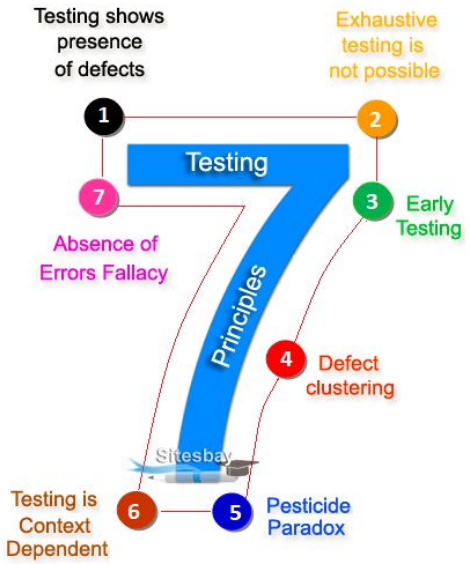



Fig: Software Testing Principles

https://www.softwaretestinghelp.com/popular-bug-tracking-software/

# 7 principles of software testing

| | |
|---|---|
| **Testing shows the presence of defects** | *We can never say that our system is defect free”* |
| **Exhaustive testing is impossible** | *“Impossible to test everything”* |
| **Early testing** | *Early means Cheap* |
| **Defect clustering** | *“Small number of modules contain most of the defects”* |

https://www.softwaretestinghelp.com/popular-bug-tracking-software/

# 7 principles of software testing

**Pesticide Paradox**

*“To detect more defects, we need change the test data”*

**Testing is context dependent**

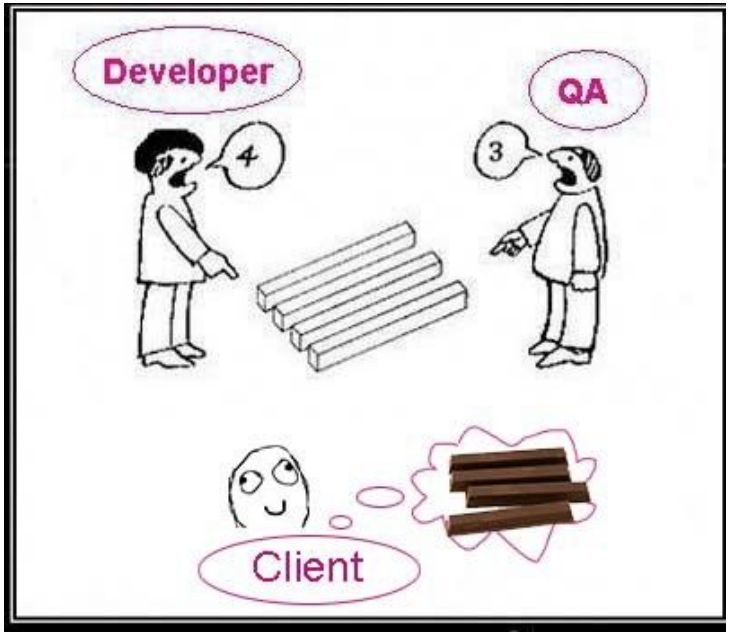
*“Tests are done differently in a different context”*

**Absence of Error Fallacy**

*“focus on the business need”*

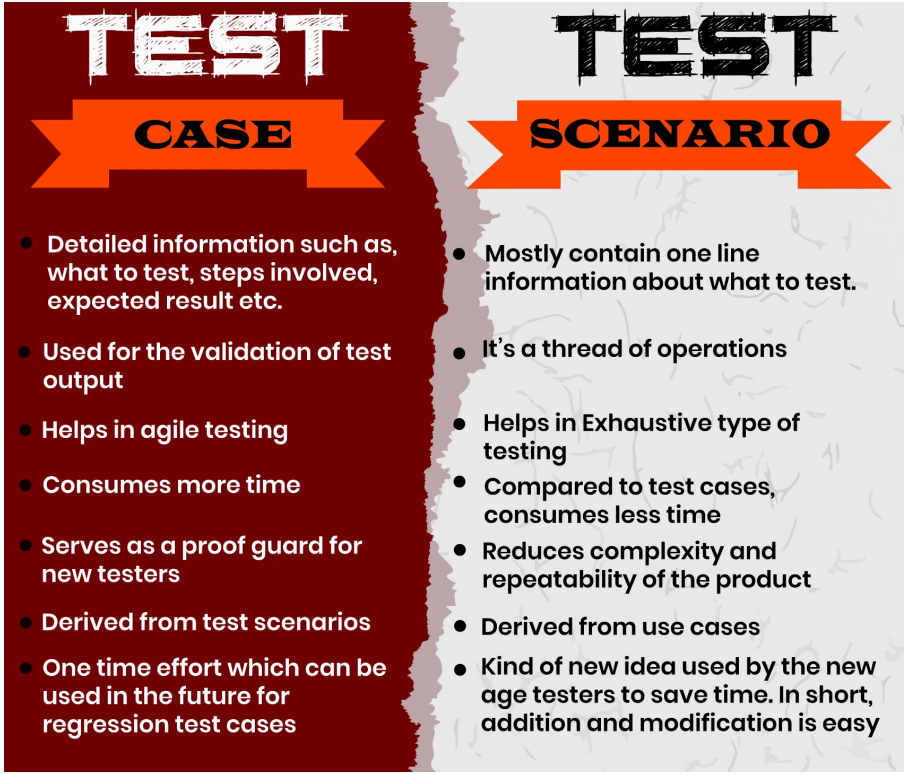
https://www.softwaretestinghelp.com/popular-bug-tracking-software/

# 7 principles of software testing




https://www.softwaretestinghelp.com/popular-bug-tracking-software/

# Test Case Vs Test Scenario



# What is Requirement Traceability Matrix ?

A document that demonstrates the relationship between requirements and other artifacts

| Req No | Req Desc | Testcase ID | Status |
|---|---|---|---|
| 123 | Login to the application | TC01,TC02,TC03 | TC01-Pass<br>TC02-Pass |
| 345 | Ticket Creation | TC04,TC05,TC06,<br>TC07,TC08,TC09<br>TC010 | TC04-Pass<br>TC05-Pass<br>TC06-Pass<br>TC06-Fail<br>TC07-No Run |
| 456 | Search Ticket | TC011,TC012,<br>TC013,TC014 | TC011-Pass<br>TC012-Fail<br>TC013-Pass<br>TC014-No Run |

# Severity vs Priority

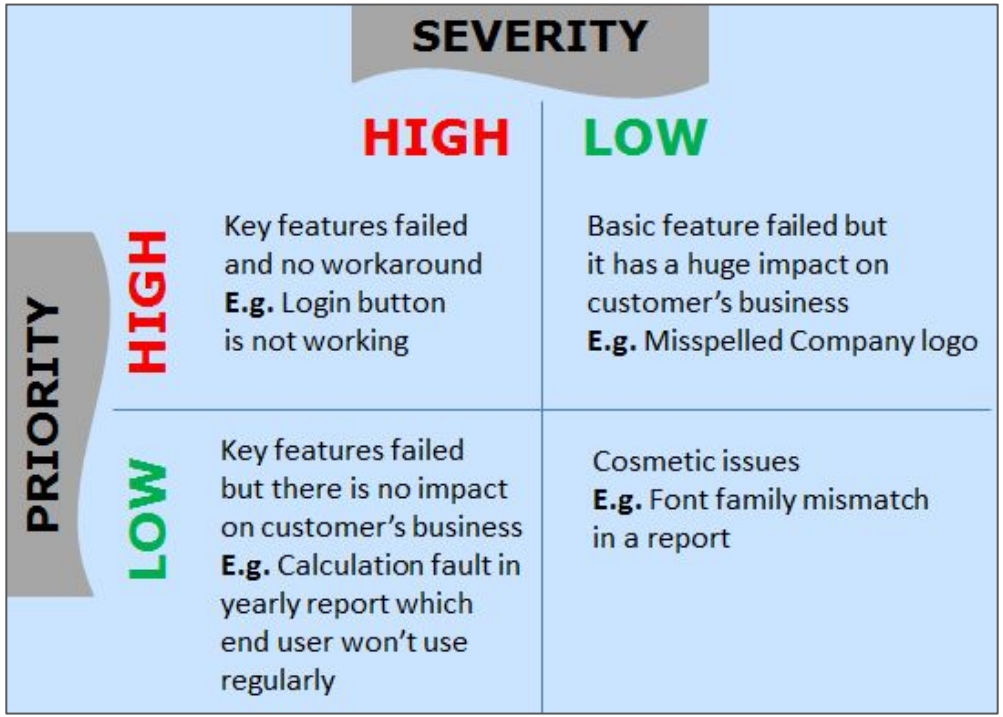
**Priority is defined as the order in which a defect should be fixed**
**How quickly bug should be removed from system.**
Low, Medium High.

Severity is degree of impact a bug.
It defines the effect the bug has on the system
Minor, Major, Critical

- Severity is about How bad Bugs are
- Priority is about how soon it should be fixed

Severity=Impact and
Priority=Urgent/Urgency

# Severity vs Priority



# Lets Create..... Test cases

# Requirements

https://bugz.atlassian.net/l/cp/1ba1G7md

## AB Testing Tool with Login and Dashboard

Created by Pro Mode
Yesterday at 5:30 PM • 2 min read

AB testing is a method of comparing two versions of a web page, landing page, or email message to determine which version performs better. It is a key tool in conversion optimization. The most common form of AB testing is the A/B test, which compares two versions of a page and measures which version performs better.

This can be accomplished by displaying the same page twice, each time with a different version of the content.

# Requirements

Go to this https://awesomeqa.com/ui/

Registration

TC - https://sdet.live/tc-template

Acceptance Criteria - User Able to register

# Designing

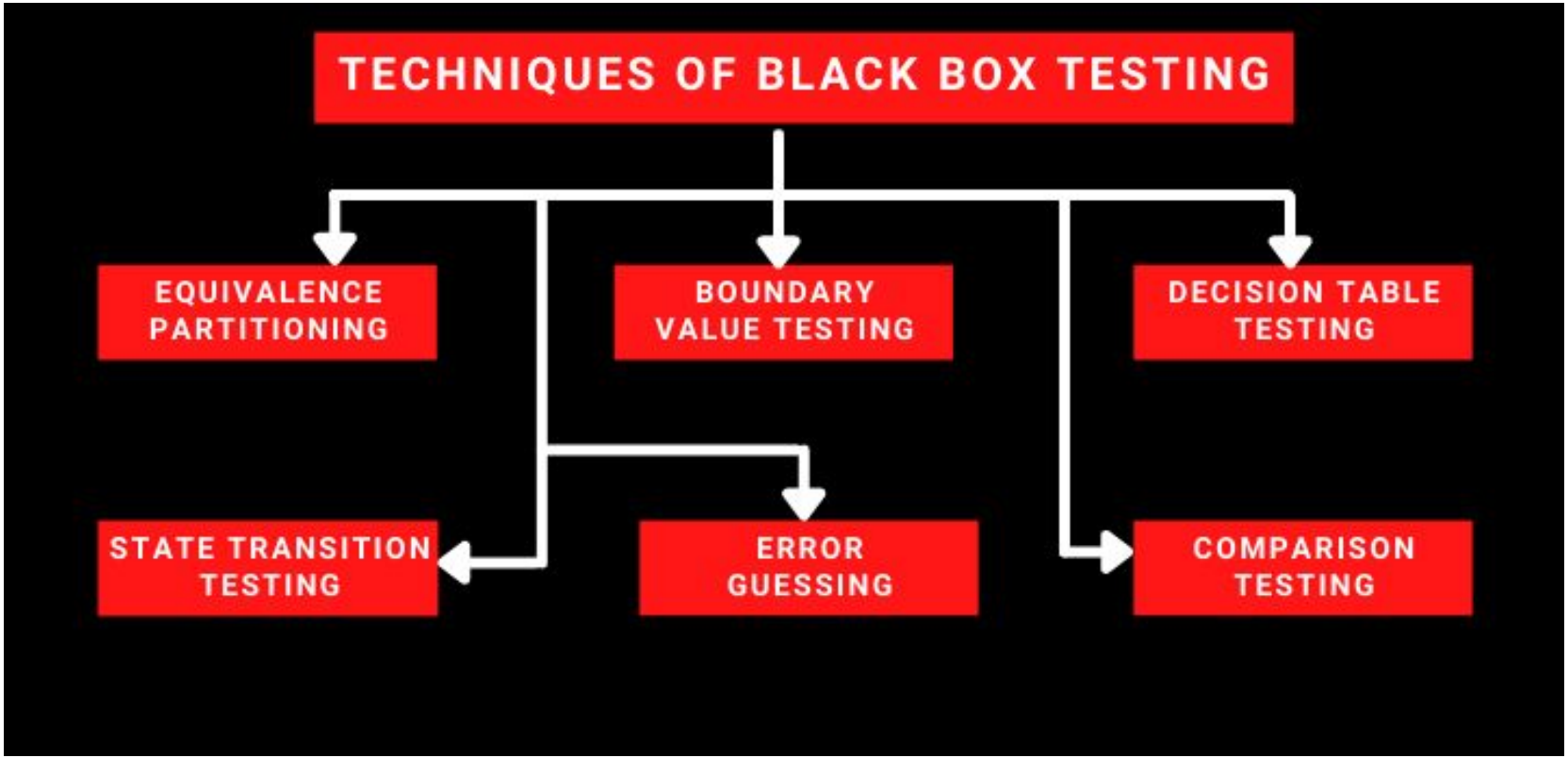
https://www.figma.com/file/IjMAQ772fjgrZy3wyfYIjB/Docushack?node-id=0%3A1

# Create Test Cases for Login VWO.com

https://sdet.live/tc-template

# Apply Black Box Techniques


The Testing
Academy


TECHNIQUES OF BLACK BOX TESTING
EQUIVALENCE PARTITIONING
BOUNDARY VALUE TESTING
DECISION TABLE TESTING
STATE TRANSITION TESTING
ERROR GUESSING
COMPARISON TESTING

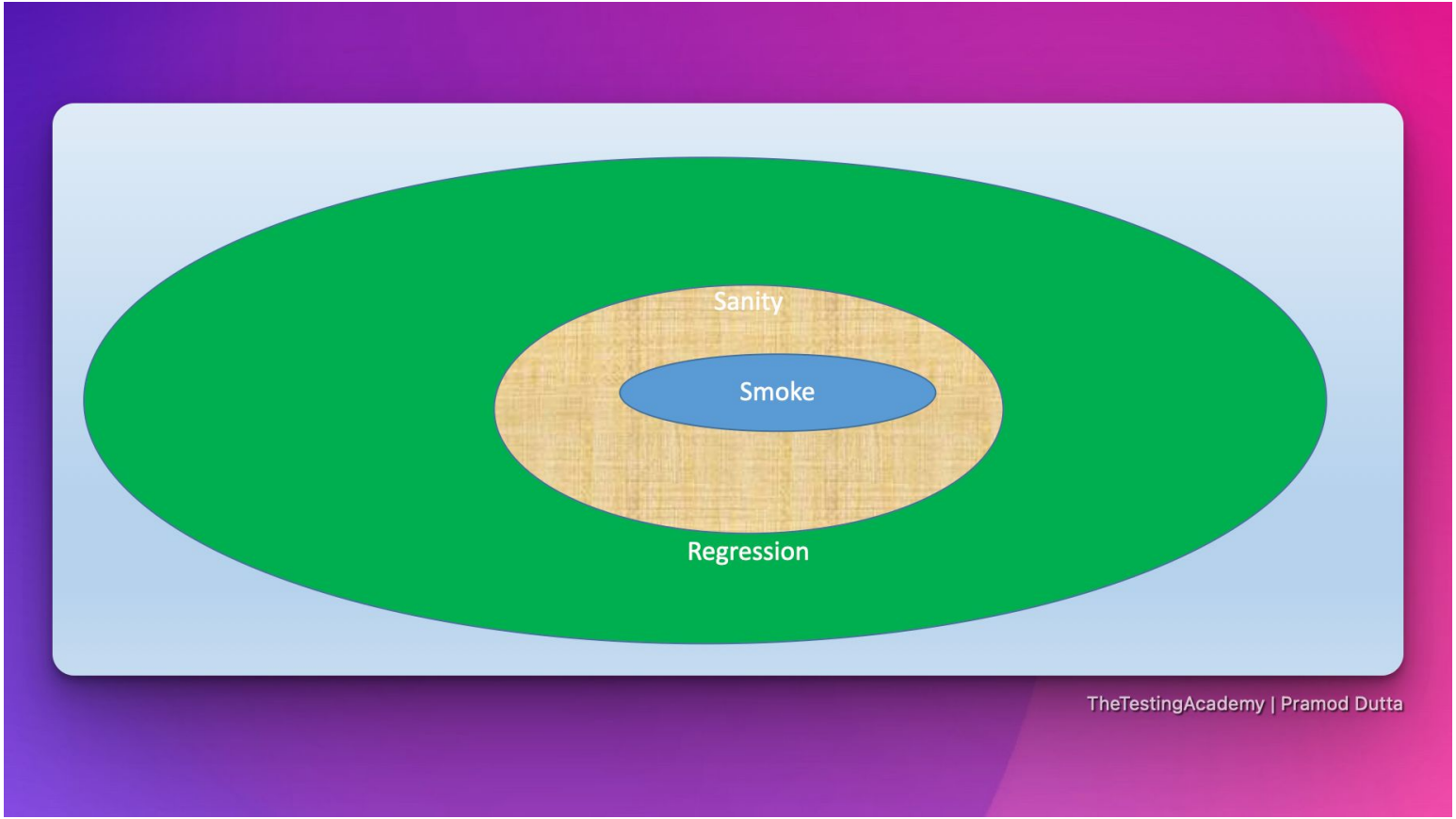
# Smoke Sanity and Regression

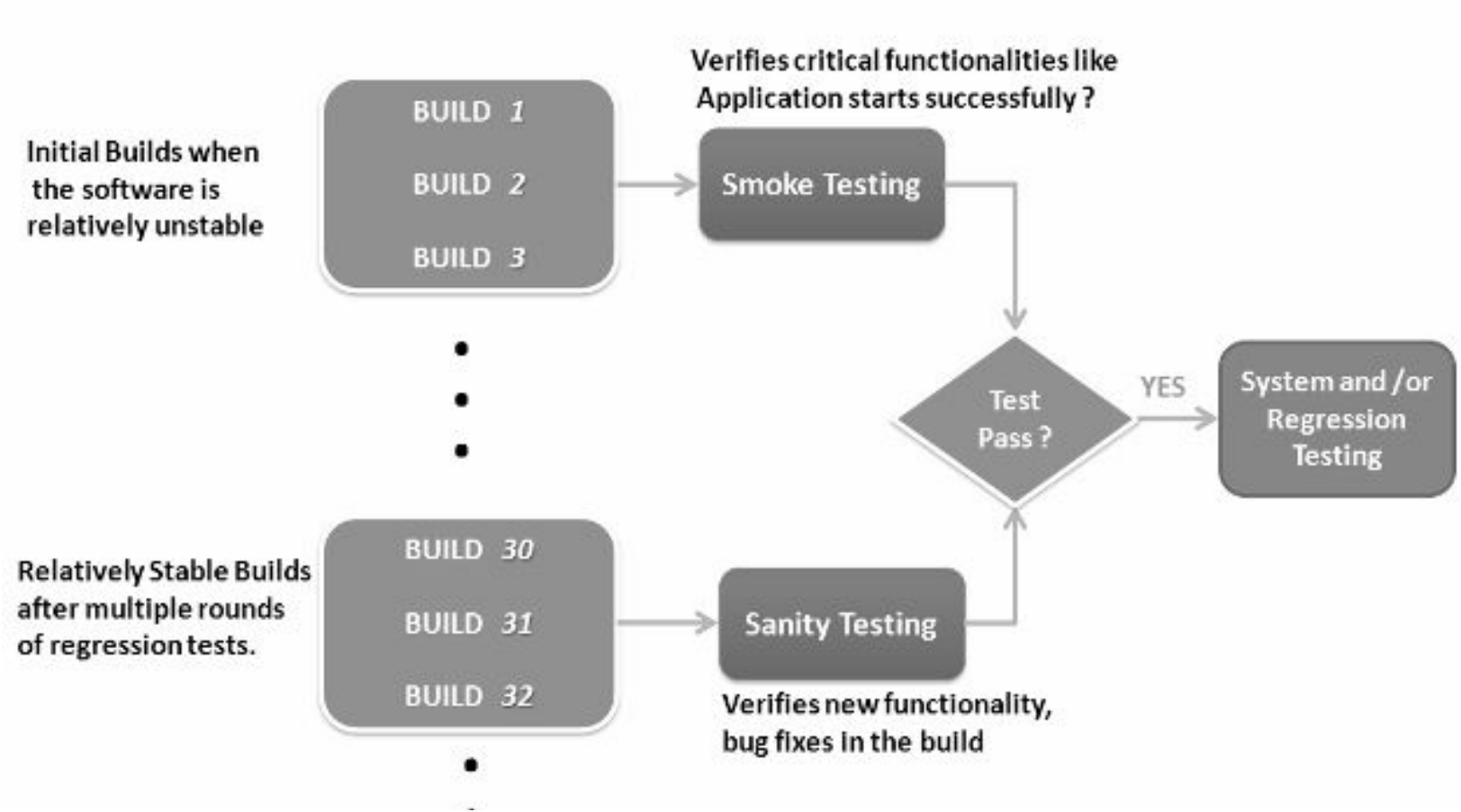
**Smoke Testing** is executed before any detailed functional tests are done on the software.

Smoke testing is also called as Build Verification Test.

**Sanity testing** is a software testing technique which does a quick evaluation of the quality of the software release to determine major functionality is working fine or not

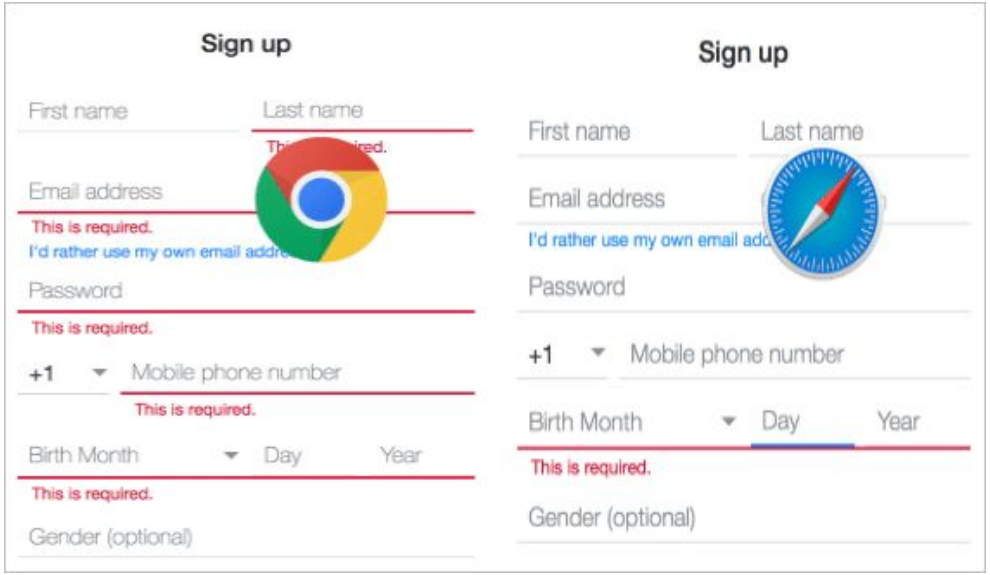
**Regression Testing** Covers detailed testing targeting all the affected areas after new functionalities are added


Sanity
Smoke
Regression
TheTestingAcademy | Pramod Dutta


Verifies critical functionalities like Application starts successfully ?
Initial Builds when the software is relatively unstable
BUILD 1
BUILD 2
BUILD 3
Smoke Testing
Test Pass ?
YES
System and /or Regression Testing
Relatively Stable Builds after multiple rounds of regression tests.
BUILD 30
BUILD 31
BUILD 32
Sanity Testing
Verifies new functionality, bug fixes in the build

# Cross Browser Testing

Cross-browser testing to test your website or application in multiple browsers- and making sure that it works consistently and as in intended without any dependencies



# Demo Cross Browser Testing

BrowserStack

# Do you know What is HTTP?

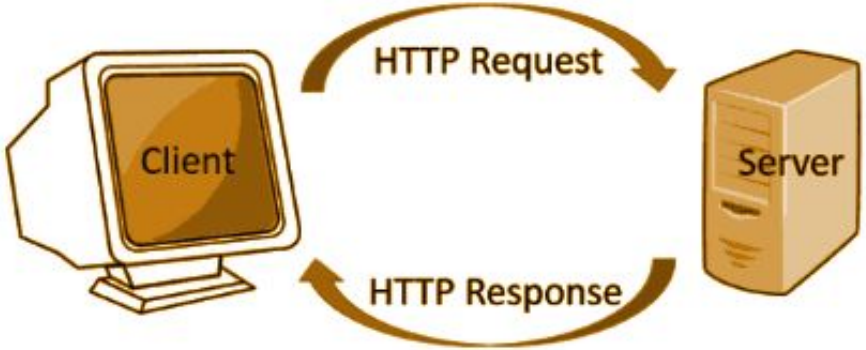
Yes

No

In Comments

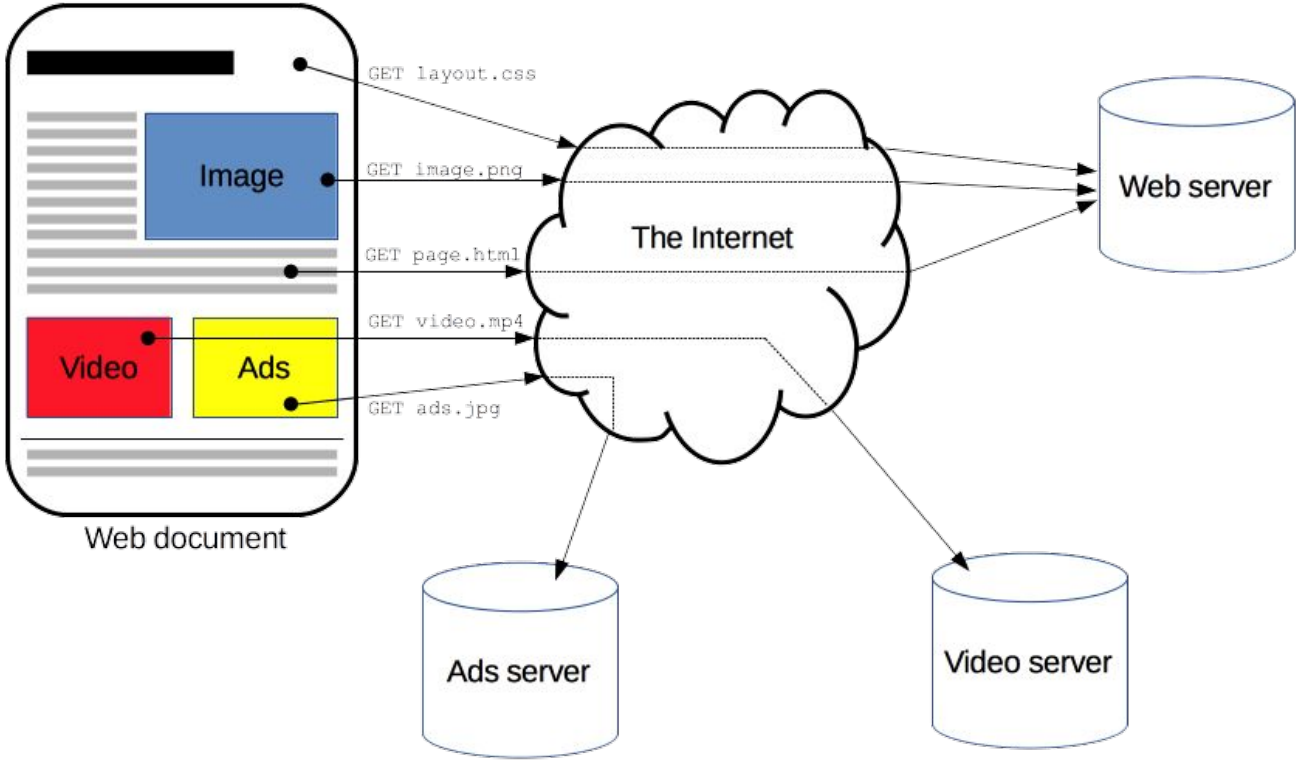
# Overview of HTTP

- HTTP is a protocol for fetching resources such as HTML documents.
- It is the foundation of any data exchange on the Web and it is a client-server protocol.
- HTTP is stateless: there is no link between two requests being successively carried out on the same connection
-



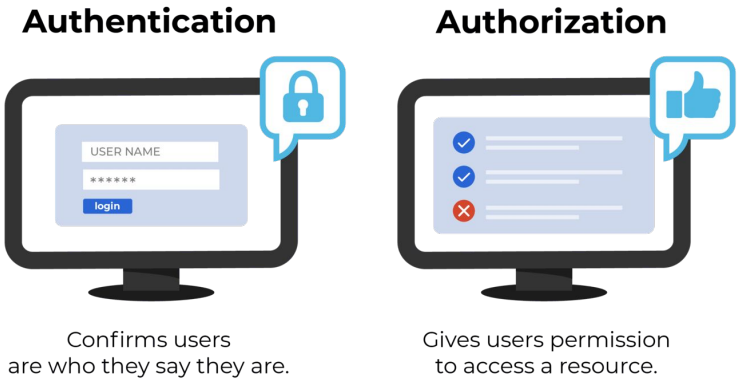

https://developer.mozilla.org/en-US/docs/Web/HTTP

# Overview of HTTP

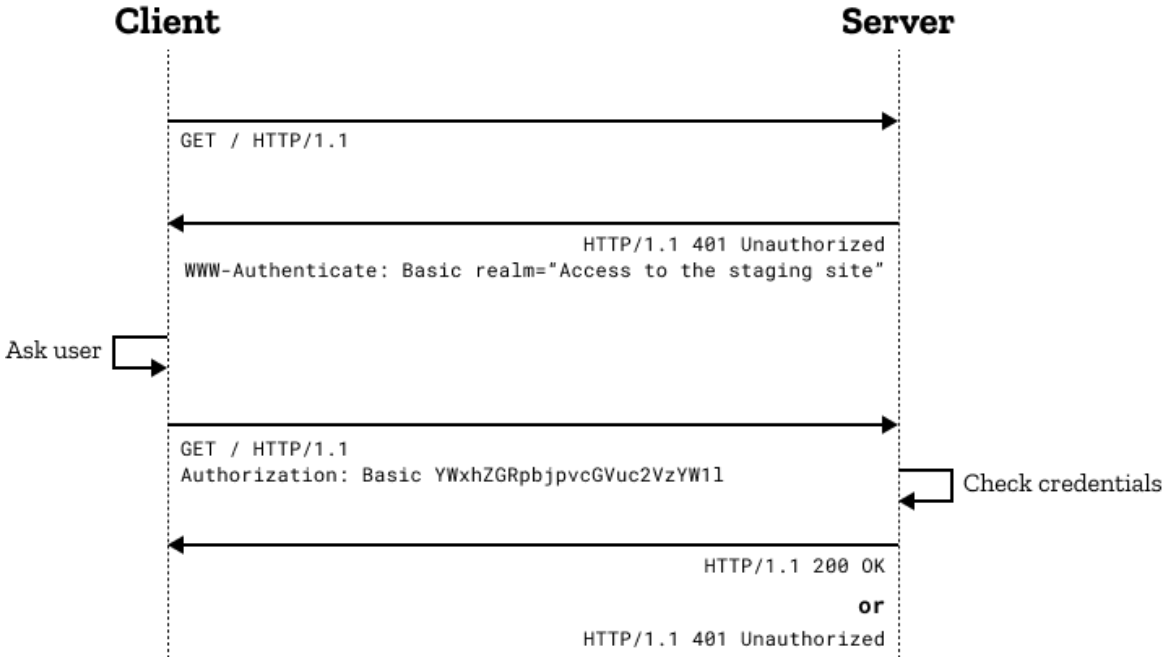


# HTTP Authentication

App.vwo.com Authentication



# HTTP Authentication



# HTTP Cookies

An HTTP cookie (web cookie, browser cookie) is a small piece of data that a server sends to a user's web browser.
The browser may store the cookie and send it back to the same server with later requests

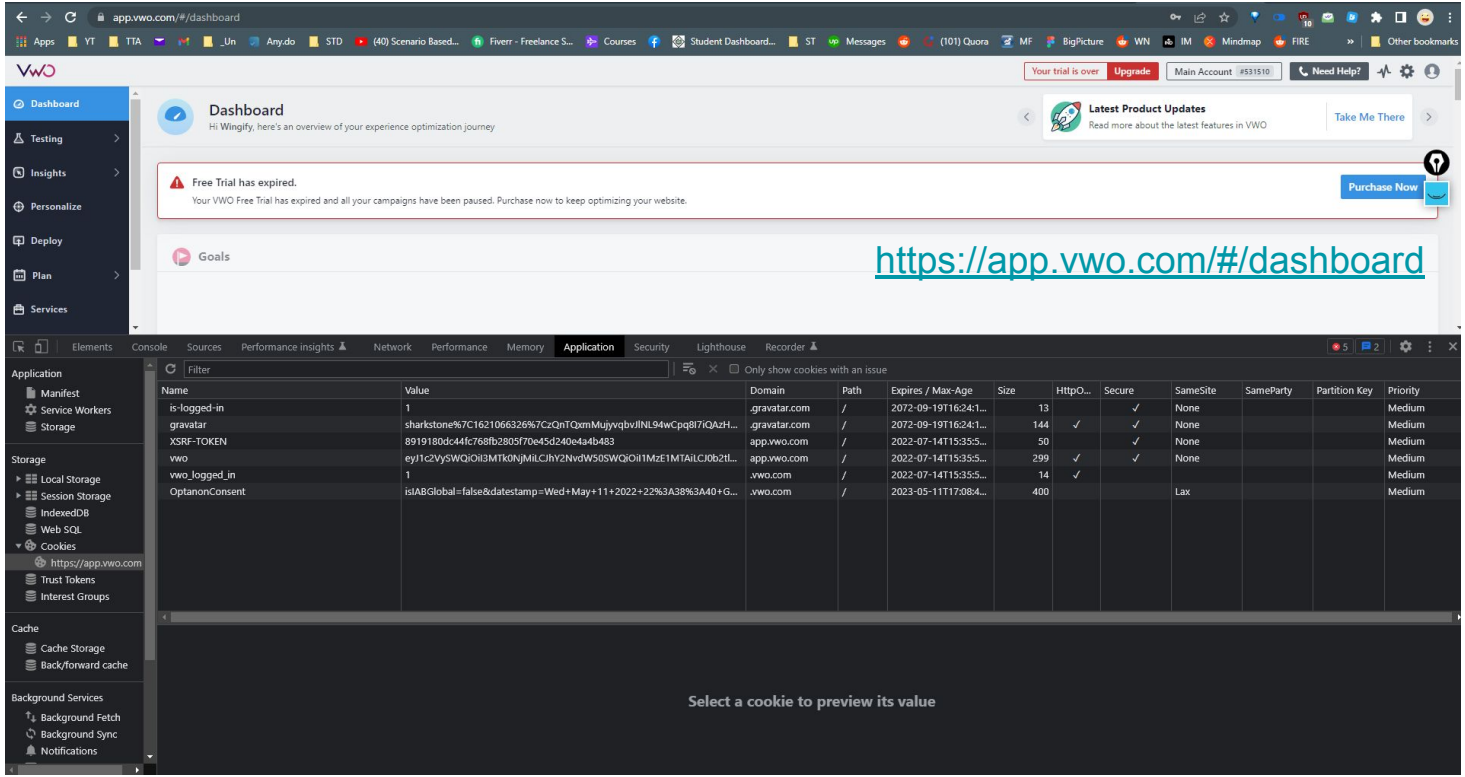
**Cookies are mainly used for three purposes:**

Session management
Logins, shopping carts, game scores, or anything else the server should remember

Personalization
User preferences, themes, and other settings

Tracking
Recording and analyzing user behavior

# HTTP Cookies



# Headers

HTTP headers let the client and the server pass **additional information** with an HTTP request or response.

- Request headers contain more information about the resource to be fetched, or about the client requesting the resource.
- Response headers hold additional information about the response, like its location or about the server providing it.
- Representation headers contain information about the body of the resource, like its MIME type, or encoding/compression applied.
- Payload headers contain representation-independent information about payload data, including content length and the encoding used for transport.

HTTP header indicates which content types, expressed as MIME types, the client is able to understand.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

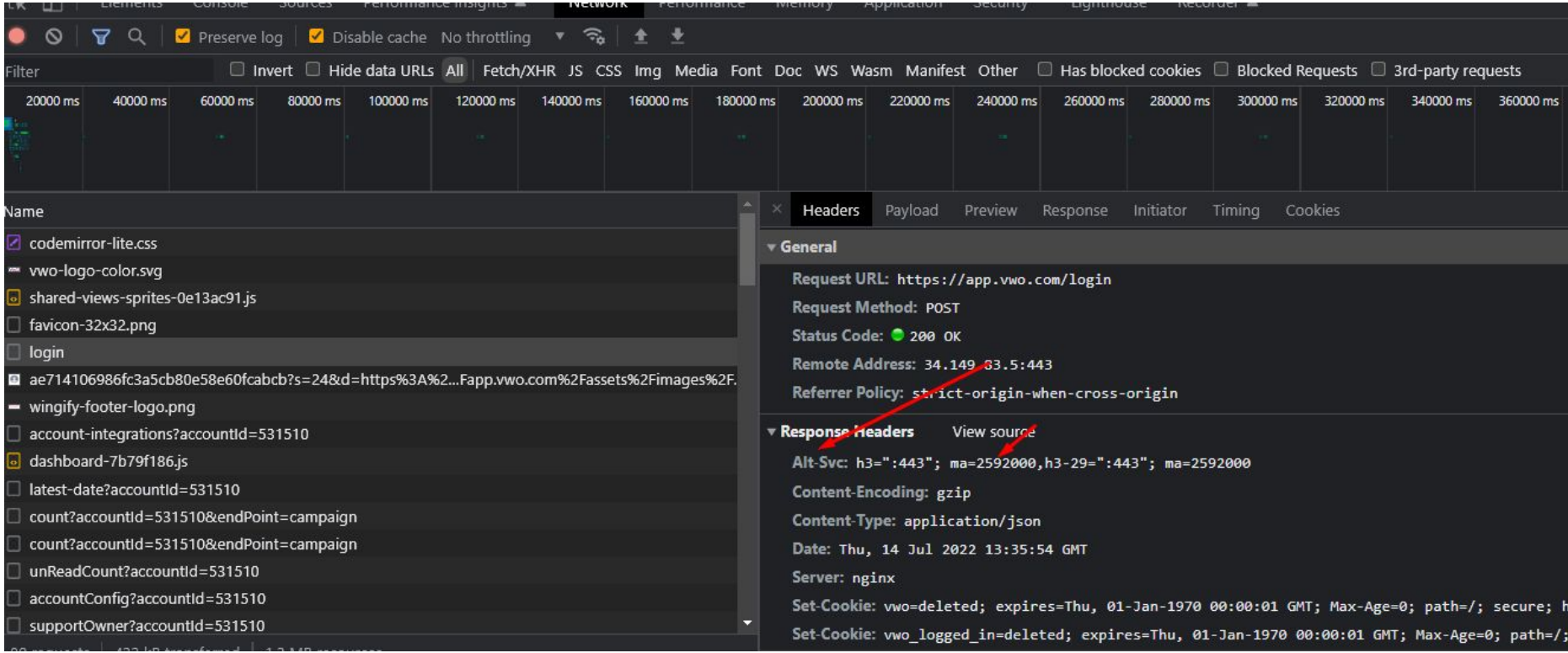
**Authentication**

**Caching**

**User agent client hints**

**Conditionals**

# Headers


Preserve log
Disable cache
No throttling
Filter
Invert
Hide data URLs
All
Fetch/XHR
JS
CSS
Img
Media
Font
Doc
WS
Wasm
Manifest
Other
Has blocked cookies
Blocked Requests
3rd-party requests
20000 ms
40000 ms
60000 ms
80000 ms
100000 ms
120000 ms
140000 ms
160000 ms
180000 ms
200000 ms
220000 ms
240000 ms
260000 ms
280000 ms
300000 ms
320000 ms
340000 ms
360000 ms
Name
codemirror-lite.css
vwo-logo-color.svg
shared-views-sprites-0e13ac91.js
favicon-32x32.png
login
ae714106986fc3a5cb80e58e60fcabcb?s=24&d=https%3A%2...Fapp.vwo.com%2Fassets%2Fimages%2F.
wingify-footer-logo.png
account-integrations?accountId=531510
dashboard-7b79f186.js
latest-date?accountId=531510
count?accountId=531510&endPoint=campaign
count?accountId=531510&endPoint=campaign
unReadCount?accountId=531510
accountConfig?accountId=531510
supportOwner?accountId=531510
Headers
Payload
Preview
Response
Initiator
Timing
Cookies
General
Request URL: https://app.vwo.com/login
Request Method: POST
Status Code: 200 OK
Remote Address: 34.149.83.5:443
Referrer Policy: strict-origin-when-cross-origin
Response Headers
View source
Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000
Content-Encoding: gzip
Content-Type: application/json
Date: Thu, 14 Jul 2022 13:35:54 GMT
Server: nginx
Set-Cookie: vwo=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; Max-Age=0; path=/; secure;
Set-Cookie: vwo_logged_in=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; Max-Age=0; path=/;

# Headers

The Content-Type representation header is used to indicate the original media type of the resource (prior to any content encoding applied for sending).

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Type

# Tools for Cookie, Headers View

https://chrome.google.com/webstore/detail/editthiscookie/fngmhnnpilhplaeedifhccceomclgfbg?hl=en

# URL Basics

# URL BASICS

Uniform Resource Locators = **Address**

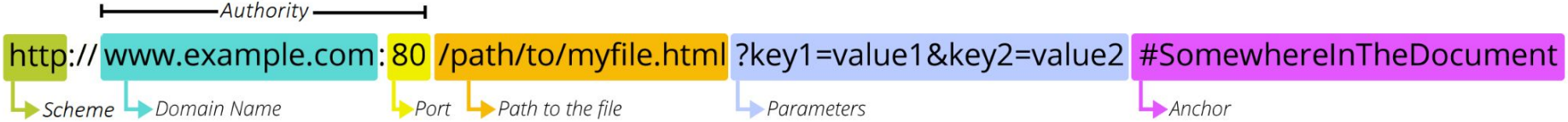
**It is the mechanism used by browsers to retrieve any published resource on the web.**

**A URL is nothing more than the address of a given unique resource on the Web**

Such resources can be an HTML page, a CSS document, an image, etc

```
https://developer.mozilla.org
https://developer.mozilla.org/en-US/docs/Learn/
https://developer.mozilla.org/en-US/search?q=URL
```

# URL BASICS

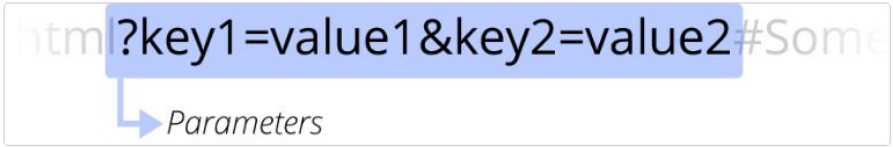

Authority
http://
www.example.com
:
80
/path/to/myfile.html
?key1=value1&key2=value2
#SomewhereInTheDocument
Scheme
Domain Name
Port
Path to the file
Parameters
Anchor

# URL BASICS

## Path to resource


`/path/to/myfile.html` is the path to the resource on the Web server. In the early days of the Web, a path like this represented a physical file location on the Web server. Nowadays, it is mostly an abstraction handled by Web servers without any physical reality.

## Parameters



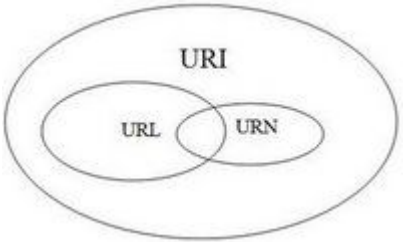
# Quiz

https://abc.com/xyz/?q=1

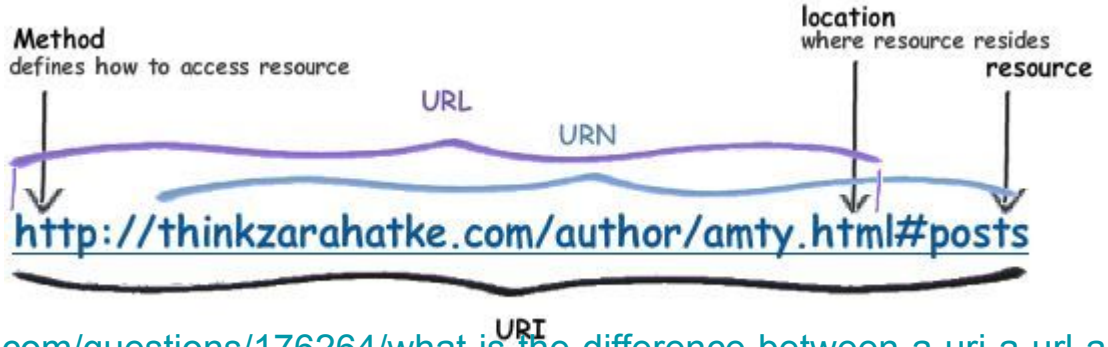
Which one is path param
Which one is query param

# URI VS URL VS URN

**A URI has two specializations known as URL and URN.**




- URI (uniform resource identifier) identifies a resource (text document, image file, etc)
- URL (uniform resource locator) is a subset of the URIs that include a network location
- URN (uniform resource name) is a subset of URIs that include a name within a given space, but no location




https://stackoverflow.com/questions/176264/what-is-the-difference-between-a-uri-a-url-and-a-urn/1984225#1984225

# URI VS URL VS URN

## URL -- Uniform Resource Locator

- http://example.com/mypage.html
- ftp://example.com/download.zip
- mailto:user@example.com
- file:///home/user/file.txt
- http://example.com/resource?foo=bar#fragment

https://www.geeksforgeeks.org/difference-between-url-uri-and-urn-in-java/

## URN -- Uniform Resource Name

- urn:isbn:0451450523 to identify a book by its ISBN number.
- urn:uuid:6e8bc430-9c3a-11d9-9669-0800200c9a66 a globally unique identifier
- urn:publishing:book - An XML namespace that identifies the document as a type of book.

# Absolute URLs?

## What are Absolute URLs?

An absolute URL contains the entire address from the protocol (HTTPS) to the domain name (www.example.com) and includes the location within your website in your folder system (/foldernameA or /foldernameB) names within the URL.

Basically, it's the full URL of the page that you link to.

**An example of an absolute URL is:**

<a href = **http://www.example.com/xyz.html**>

# What are Relative URLs?

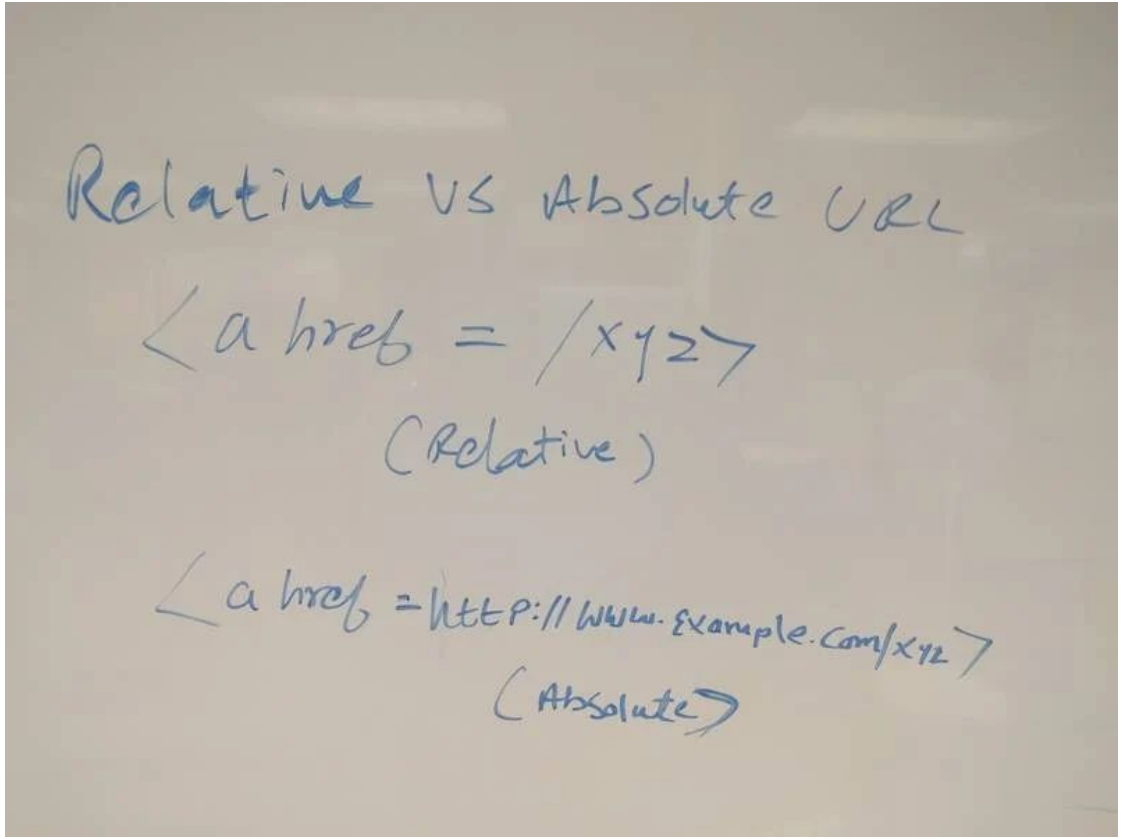
## What are Relative URLs?

The relative URL, on the other hand, does not use the full web address and only contains the location following the domain. It assumes that the link you add is on the same site and is part of the same root domain.

The relative path starts with the forward slash and leads the browser to stay within the current site.

**An example of a relative URL is:**

<a href = "**/xyz.html**">


Relative VS Absolute URL
<a href = /xyz>
(Relative)
<a href = http://www.example.com/xyz>
(Absolute)

# MIME types

A media type (also known as a Multipurpose Internet Mail Extensions or MIME type) indicates the nature and format of a document, file, or assortment of bytes.

```
application/pdf
application/json
```

multipart/form-data

The multipart/form-data type can be used when sending the values of a completed HTML Form from browser to server.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types

https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types#multipartform-data

# MIME types

| MIME type | Audio or video type |
| --- | --- |
| `audio/wave` `audio/wav` `audio/x-wav` `audio/x-pn-wav` | An audio file in the WAVE container format. The PCM audio codec (WAVE codec "1") is often supported, but other codecs have limited support (if any). |
| `audio/webm` | An audio file in the WebM container format. Vorbis and Opus are the codecs officially supported by the WebM specification. |
| `video/webm` | A video file, possibly with audio, in the WebM container format. VP8 and VP9 are the most common video codecs; Vorbis and Opus the most common audio codecs. |
| `audio/ogg` | An audio file in the Ogg container format. Vorbis is the most common audio codec used in such a container; however, Opus is now supported by Ogg as well. |
| `video/ogg` | A video file, possibly with audio, in the Ogg container format. Theora is the usual video codec used within it; Vorbis is the usual audio codec, although Opus is becoming more common. |
| `application/ogg` | An audio or video file using the Ogg container format. Theora is the usual video codec used within it; Vorbis is the usual audio codec. |

# JSON Server

- Install Node JS
- Check the Node and NPM version
  - Open CMD and type node –version
  - Open CMD and type npm –version
- Open CMD and Type npm i -g json-server
- Open CMD and Type json-server -w db.json
- Create a db.json file before like this

```
{
  "posts": [
    { "id": 1, "title":
"json-server", "author":
"typicode" }
  ],
  "comments": [
    { "id": 1, "body": "some
comment", "postId": 1 }
  ],
  "profile": { "name":
"typicode" }
}
```

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

# HTTP request methods

`GET`

The `GET` method requests a representation of the specified resource. Requests using `GET` should only retrieve data.

`HEAD`

The `HEAD` method asks for a response identical to a `GET` request, but without the response body.

`POST`

The `POST` method submits an entity to the specified resource, often causing a change in state or side effects on the server.

`PUT`

The `PUT` method replaces all current representations of the target resource with the request payload.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

# HTTP request methods

`DELETE`

The `DELETE` method deletes the specified resource.

`CONNECT`

The `CONNECT` method establishes a tunnel to the server identified by the target resource.

`OPTIONS`

The `OPTIONS` method describes the communication options for the target resource.

`TRACE`

The `TRACE` method performs a message loop-back test along the path to the target resource.

`PATCH`

The `PATCH` method applies partial modifications to a resource.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

# PUT VS POST

| PUT | POST |
|---|---|
| Replacing existing resource<br>or Creating if resource is not exist<br>*http://www.example.com/customer/{id}*<br>*http://www.example.com/customer/123/orders/456*<br>Identifier is chosen by the client | Creating new resources(and subordinate resources)<br>*http://www.example.com/ customer/*<br>*http://www.example.com/customer/123/orders*<br>Identifier is returned by server |
| Idempotent i.e. if you PUT a resource twice, it has no effect.<br>Ex: Do it as many times as you want, the result will be same. x=1; | POST is neither safe nor idempotent. It is therefore recommended for non-idempotent resource requests.<br>Ex: x++; |
| Works as specific | Works as abstractive |
| If you create or update a resource using PUT and then make that same call again, the resource is still there and still has the same state as it did with the first call. | Making two identical POST requests will most-likely result in two resources containing the same information. |

# Quiz

**Can we use POST request to do all Request?**

# JSON

How to start a JSON Server for the API Testing

JSON Server is a Dummy Serve based on the JSON Database

1. Install Node JS nodejs.org
2. Open CMD -> Write  npm i -g json-server
3. Create a db.json in the working dir.
4. Open CMD -> json-server -w db.json

# JSON Dummy Data


```
{
  "users": [
    {
      "id": 1,
      "data": {
        "id": 2,
        "email": "Pramod@reqres.in",
        "first_name": "Pramod",
        "last_name": "Weaver",
        "avatar": "https://reqres.in/img/faces/2-image.jpg"
      },
      "support": {
        "url": "https://reqres.in/#support-heading",
        "text": "To keep ReqRes free, contributions towards server costs are appreciated!"
      }
    },
    {
      "id": 2,
      "data": {
        "id": 2,
        "email": "janet@reqres.in",
        "first_name": "Janet",
        "last_name": "Weaver",
        "avatar": "https://reqres.in/img/faces/2-image.jpg"
      },
      "support": {
        "url": "https://reqres.in/#support-heading",
        "text": "To keep ReqRes free, contributions towards server costs are appreciated!"
      }
    }
  ]
}
```

# HTTP response status codes

HTTP response status codes indicate whether a specific HTTP request has been successfully completed. Responses are grouped in five classes:

Informational responses (100–199)
Successful responses (200–299)
Redirection messages (300–399)
Client error responses (400–499)
Server error responses (500–599)

https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

# User agent

A user agent is a computer program representing a person, for example, a browser in a Web context.

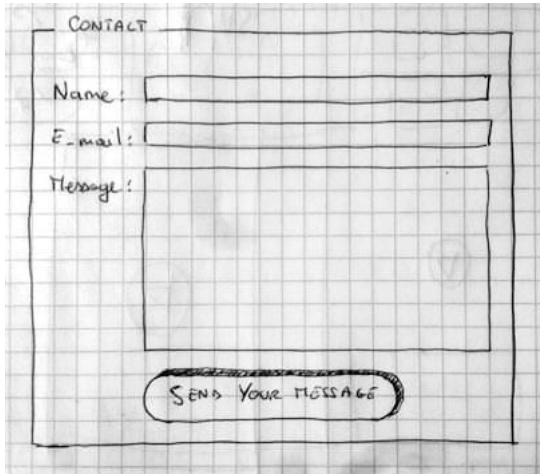
The user agent string can be accessed with JavaScript on the client side using the NavigatorID.userAgent property.

A typical user agent string looks like this: "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:35.0) Gecko/20100101 Firefox/35.0".

https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

# HTML Forms

Web forms — Working with user data




```
<form action="/my-handling-form-page"
method="post">
 <ul>
  <li>
    <label for="name">Name:</label>
    <input type="text" id="name" name="user_name">
  </li>
  <li>
    <label for="mail">E-mail:</label>
    <input type="email" id="mail" name="user_email">
  </li>
  <li>
    <label for="msg">Message:</label>
    <textarea id="msg"
name="user_message"></textarea>
  </li>
 </ul>
</form>
```