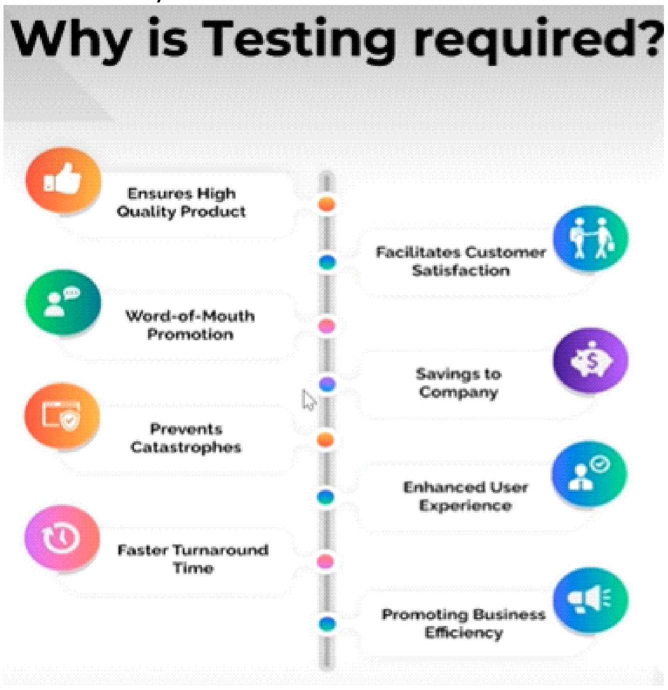
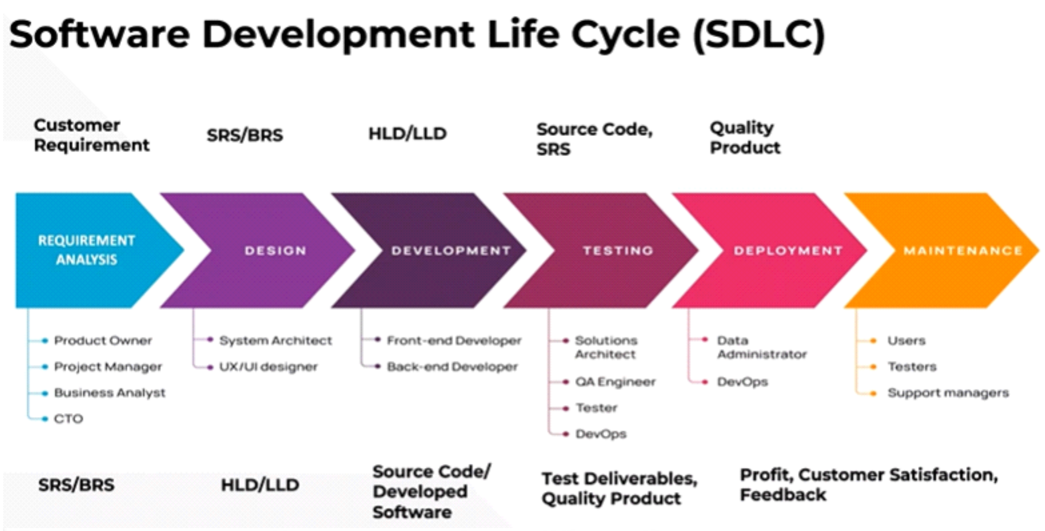
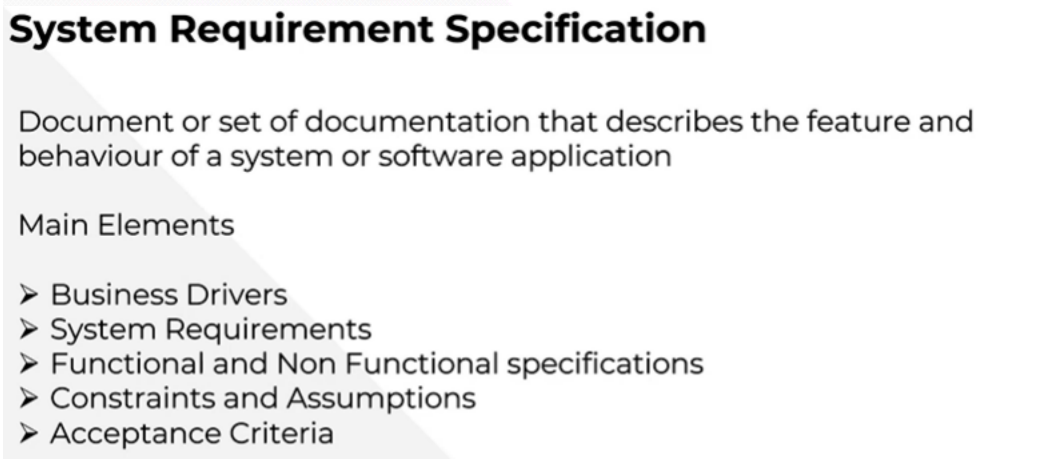
Testing - Introduction

Monday, April 11, 2022 1:25 PM

- Software testing is part of software development process. It is an activity to detect and identify the defects in the software. It's main aim is to release quality product to the client.







System Requirement Specification

Document or set of documentation that describes the feature and behaviour of a system or software application

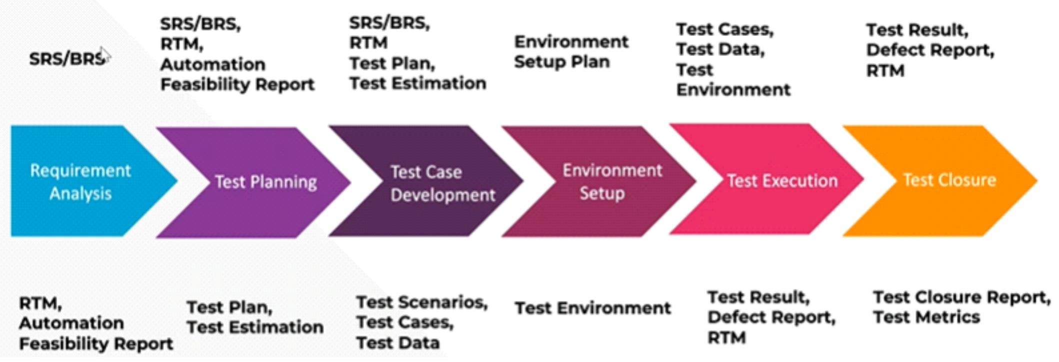
Main Elements

- Business Drivers
- System Requirements
- Functional and Non Functional specifications
- Constraints and Assumptions
- Acceptance Criteria

- In SRS document we have function and non functional specifications, functional is about functioning details of the app and non functional mean it related to the security and more details. As different people work in functional and non functional specifications.

Software Testing Cycle (STLC)

Software Testing Cycle (STLC)


SRS/BRS
SRS/BRS, RTM, Automation Feasibility Report
SRS/BRS, RTM Test Plan, Test Estimation
Environment Setup Plan
Test Cases, Test Data, Test Environment
Test Result, Defect Report, RTM
Requirement Analysis
Test Planning
Test Case Development
Environment Setup
Test Execution
Test Closure
RTM, Automation Feasibility Report
Test Plan, Test Estimation
Test Scenarios, Test Cases, Test Data
Test Environment
Test Result, Defect Report, RTM
Test Closure Report, Test Metrics

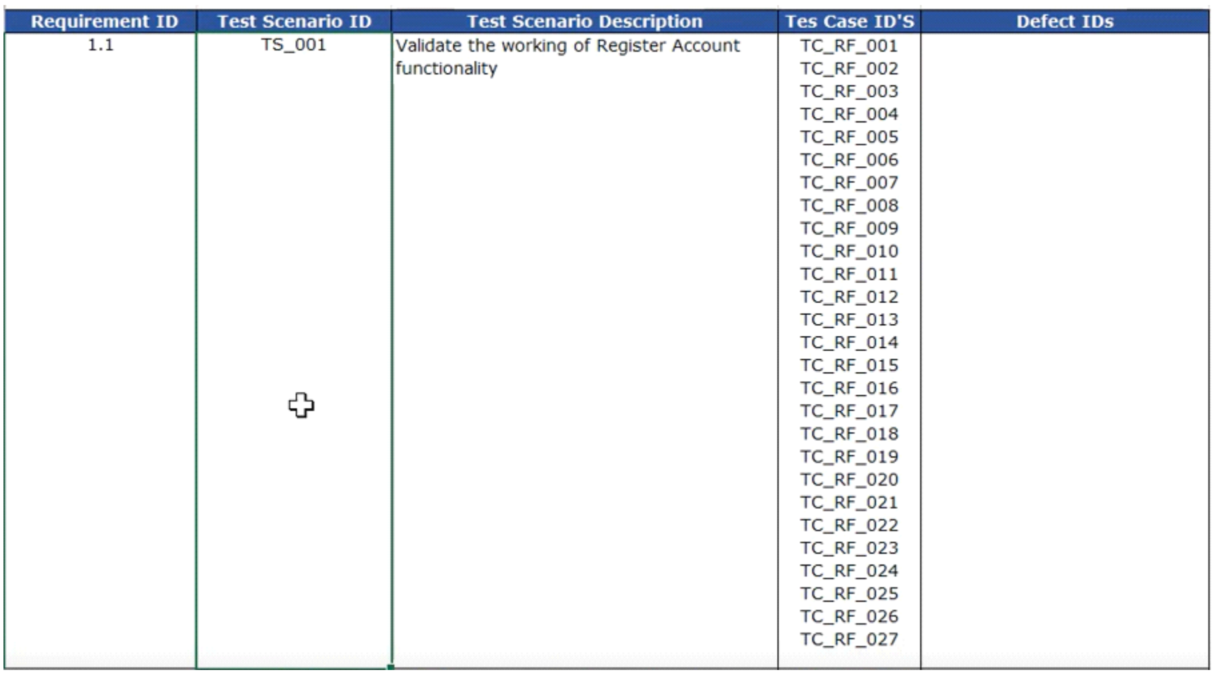
Test Plan

Test plan is a document describing the scope, approach, resource and schedule of intended test activities

Components in Test Plan

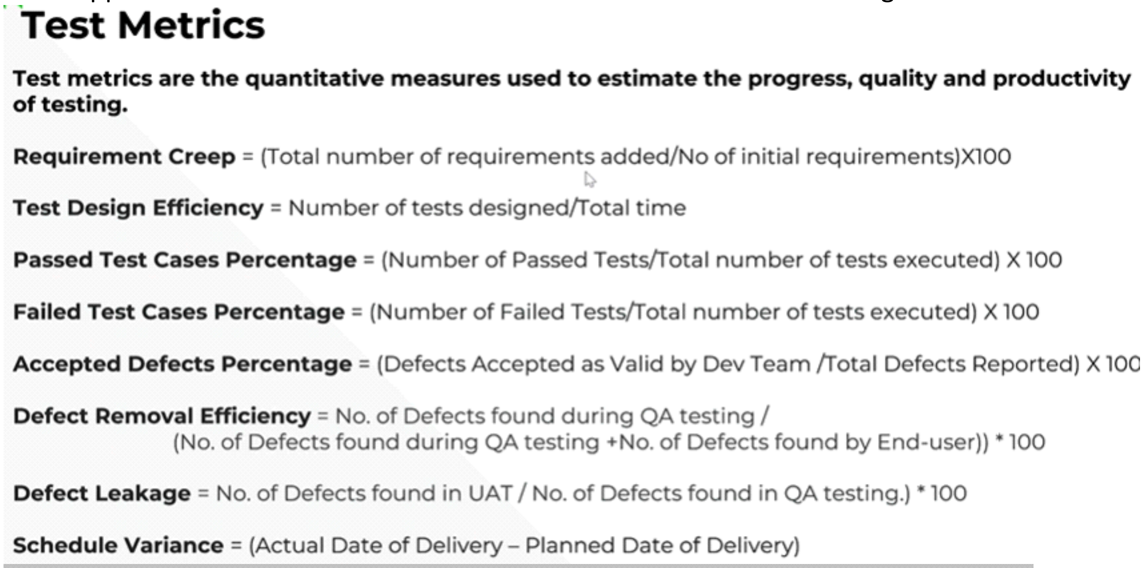
- Scope
- Test Strategy
- Roles and Responsibilities
- Resource planning
- Test Schedule
- Entry and Exit Criteria
- Risk and Mitigation
- Test Deliverables

- Scope determines how much extent we can test the application. Test Strategy determines the schedule of testing phases and what type of testing we need to do. Roles and responsibility determines the mapping of name and his assigned task of the testing phase. Resource planning determines what resources we have and how to utilize those resources. Test Schedule is a detailed plan of all the testing phases that we need to process on . Test Scenarios in the sense that for a test what environment we must have what type of data we need to have and what are steps to test the application and expected result will be mentioned in it. And in the next stage (Test Execution stage) they check the results with actual test data results.
- For testing we are having a separate testing environment. In that environment we are going to execute the tests.
- RTM document will look like,



Requirement ID	Test Scenario ID	Test Scenario Description	Tes Case ID'S	Defect IDs
1.1	TS_001	Validate the working of Register Account functionality	TC_RF_001	
			TC_RF_002	
			TC_RF_003	
			TC_RF_004	
			TC_RF_005	
			TC_RF_006	
			TC_RF_007	
			TC_RF_008	
			TC_RF_009	
			TC_RF_010	
			TC_RF_011	
			TC_RF_012	
			TC_RF_013	
			TC_RF_014	
			TC_RF_015	
			TC_RF_016	
			TC_RF_017	
			TC_RF_018	
			TC_RF_019	
			TC_RF_020	
			TC_RF_021	
			TC_RF_022	
			TC_RF_023	
			TC_RF_024	
			TC_RF_025	
			TC_RF_026	
			TC_RF_027	

- After each phase some of those columns will be filled gradually, at last after filling all the columns, find what test cases are defected in test execution phase.
- And in final test closure phase we find how much affective those test cases are on our application. And there are some metrics to work on that as following.



Test Metrics

Test metrics are the quantitative measures used to estimate the progress, quality and productivity of testing.

Requirement Creep = (Total number of requirements added/No of initial requirements)X100

Test Design Efficiency = Number of tests designed/Total time

Passed Test Cases Percentage = (Number of Passed Tests/Total number of tests executed) X 100

Failed Test Cases Percentage = (Number of Failed Tests/Total number of tests executed) X 100

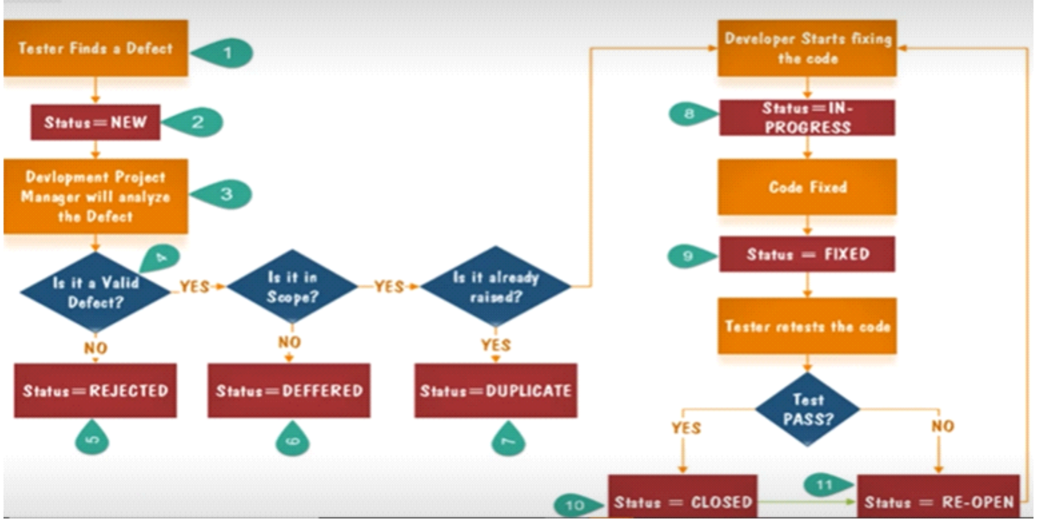
Accepted Defects Percentage = (Defects Accepted as Valid by Dev Team /Total Defects Reported) X 100

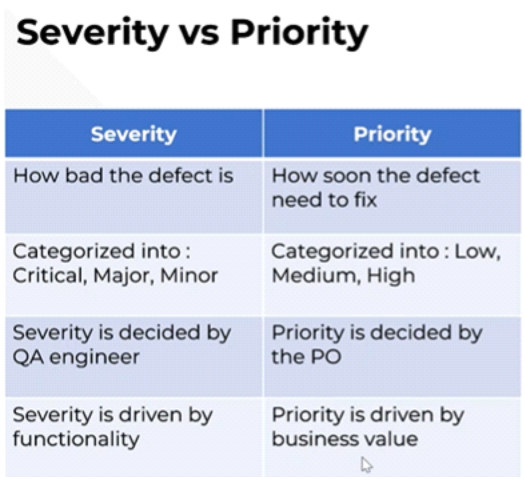
Defect Removal Efficiency = No. of Defects found during QA testing /
(No. of Defects found during QA testing +No. of Defects found by End-user)) * 100

Defect Leakage = No. of Defects found in UAT / No. of Defects found in QA testing.) * 100

Schedule Variance = (Actual Date of Delivery – Planned Date of Delivery)

Defect Life Cycle

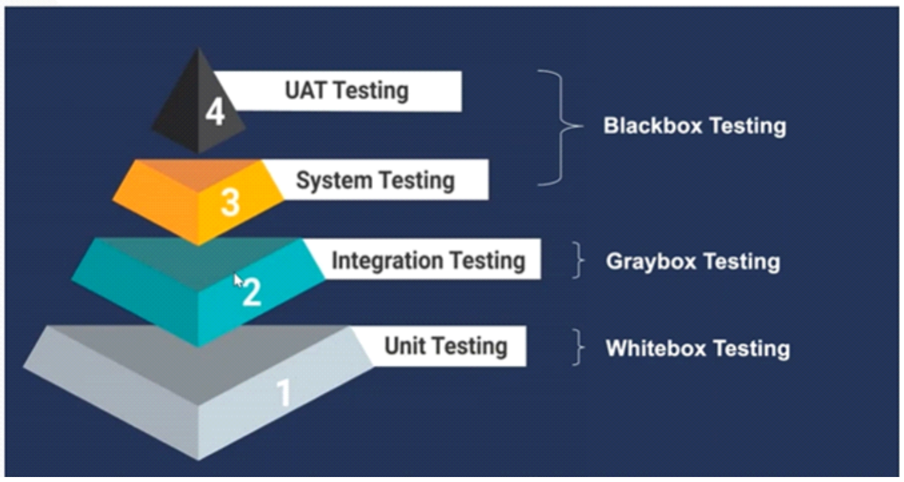




Severity vs Priority

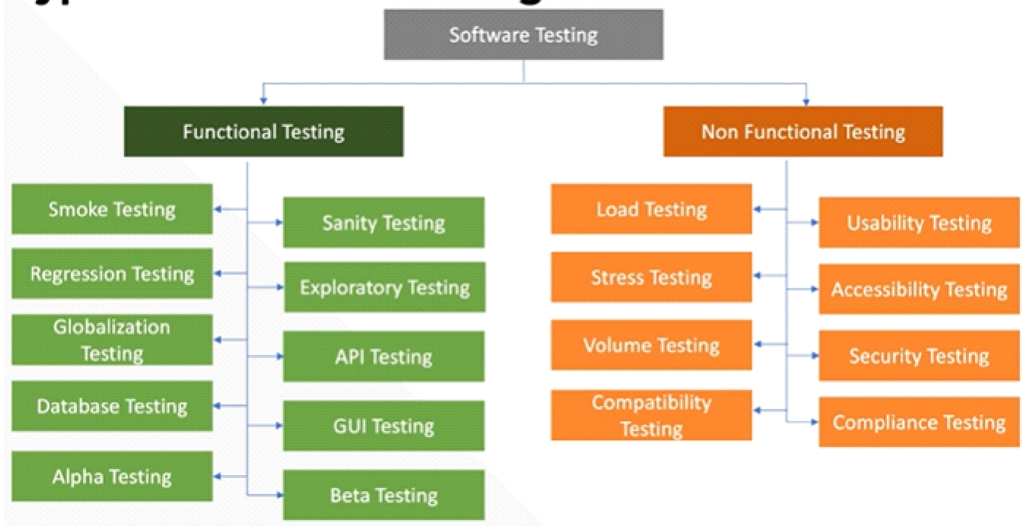
Severity	Priority
How bad the defect is	How soon the defect need to fix
Categorized into : Critical, Major, Minor	Categorized into : Low, Medium, High
Severity is decided by QA engineer	Priority is decided by the PO
Severity is driven by functionality	Priority is driven by business value

Levels of Testing



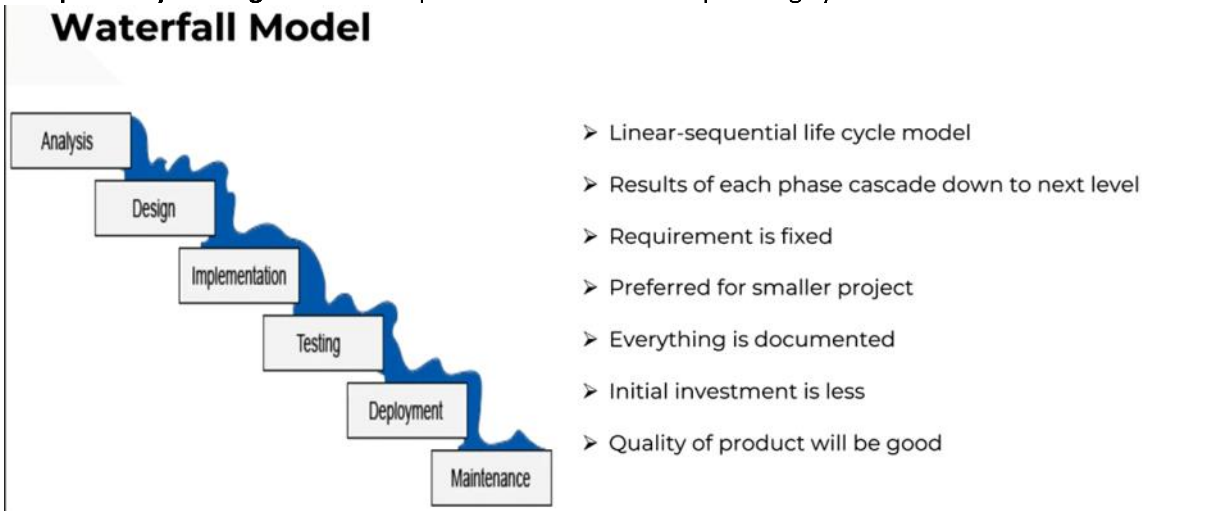
Blackbox testing is based on the functionality level
White box testing is based on the code level
Gray box testing is based on the both code and functionality level. (minimum operations done).
Integration testing is done on connection between modules.

Types of Software Testing



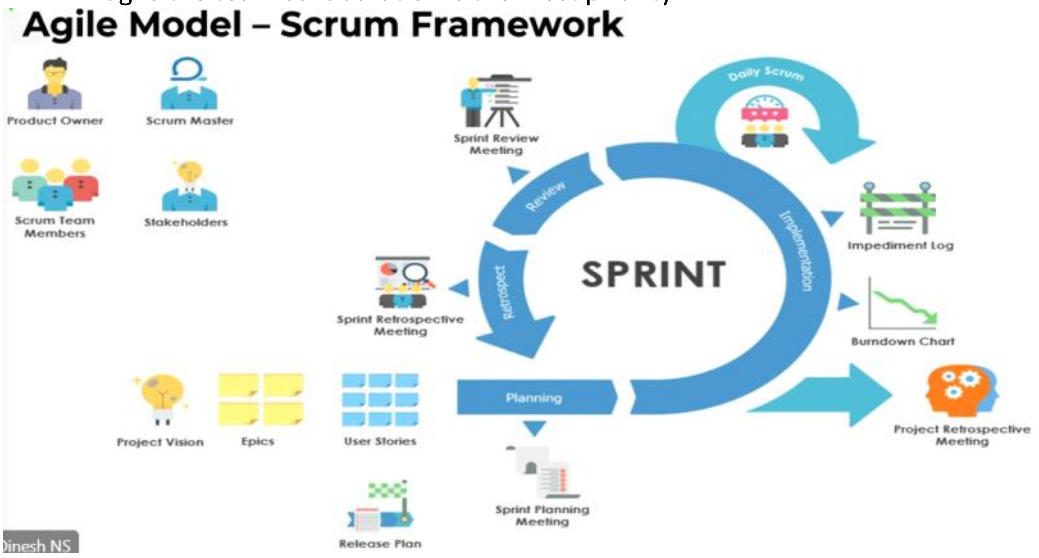
Functional Testing: Depends on the code and it affects actual application directly.
Non Functional Testing: Depends on the end product and users.
Smoke Testing: build verification testing . It is done specifically on the new added feature, and later we do it on the whole product to check compatibility between modules. Usually done in 10 - 15 mins.
Regression Testing: It is done when ever if we add a new feature to the product , we must not have impact on already created product. For that we do regression testing .
Sanity Testing: When we have limited time, we do this type of testing where we test all the important features of the application to check the reliability of the application.

Exploratory Testing: It is just do testing process while we explore the application.
Globalization Testing: Checks compatibility of the application based on the country region or any specific location.
API Testing: We check API layers, did we get the values to and fro from the database using API Layer.
Database Testing: Schema Checking etc..
GUI Testing: Look and Feel of the application , text alignment and look alignment.
Alpha Testing: Testing is done internally in the company on the product
Beta Testing: Testing is done on the released product by the company and get help and feedback from customers.
Load Testing: Check the threshold that the application can hold.
Stress Testing: Check the stress that the application can hold, a few users to the excessive amount of users.
Compliance Testing: Does it follows the standards or not.
Compatibility Testing: Does it compatible for all kinds of Operating system.


Waterfall Model

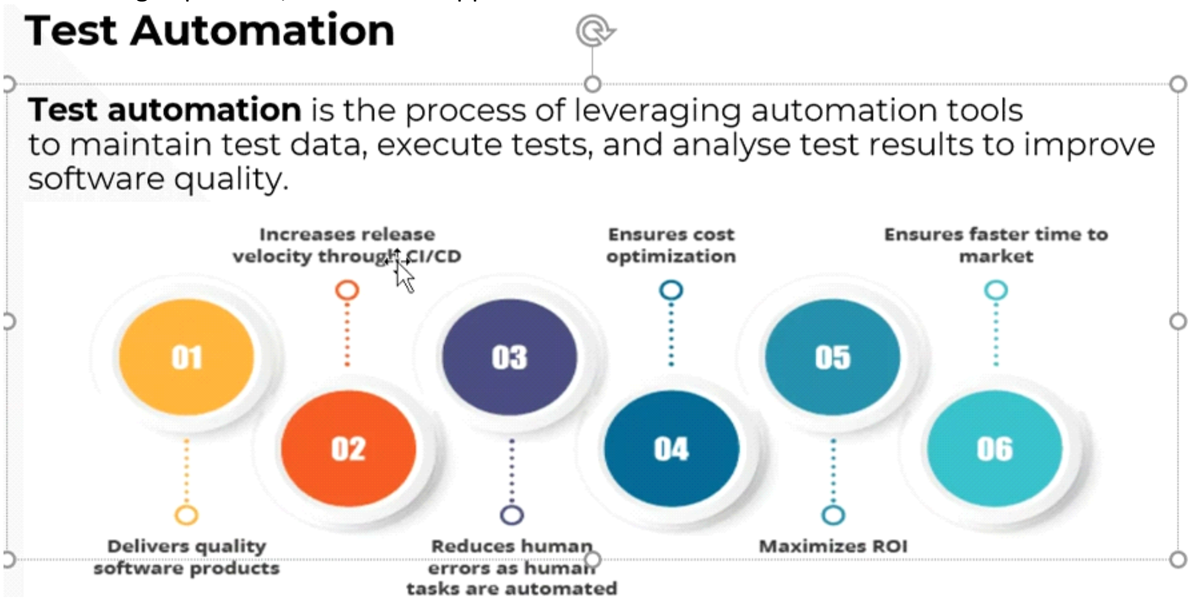
- Linear-sequential life cycle model
- Results of each phase cascade down to next level
- Requirement is fixed
- Preferred for smaller project
- Everything is documented
- Initial investment is less
- Quality of product will be good

- In waterfall model, client can only give his requirements at requirement analysis stage. He couldn't interfere in the cycle. We couldn't add the recommends in between the cycle. This is one of the disadvantage. If requirement is fixed and the project is very well documented , waterfall model is preferred.
- In agile the team collaboration is the most priority.



- The documentation is lesser in agile than that of waterfall model.
- In agile , progress comes on each sprint cycle (2-4 weeks).
- In agile , we won't have entire team. It consists product owner, scrum master (mediator between product owner and team) , scrum team members (developers , testers)
- A single scrum team will have maximum ten members.
- EPIC is a single big task, and epic is split further into multiple user stories. And then sprint starts .
- In each sprint , user stories must be developed and tested and must be ready to deploy. All the tasks that we need to do in a sprint will be in the backlog.

- Daily stand up meeting is conducted by scrum master. Where we discuss our daily goals.
- After two weeks a sprint review will be conducted and send a report to the product owner or business people.
- In agile we can adapt the changes by adding requirements in the cycle.
- Retrospective meeting is a key outline on what happened in the previous sprint and what to do best in the next sprint.
- For agile process , we use JIRA application.

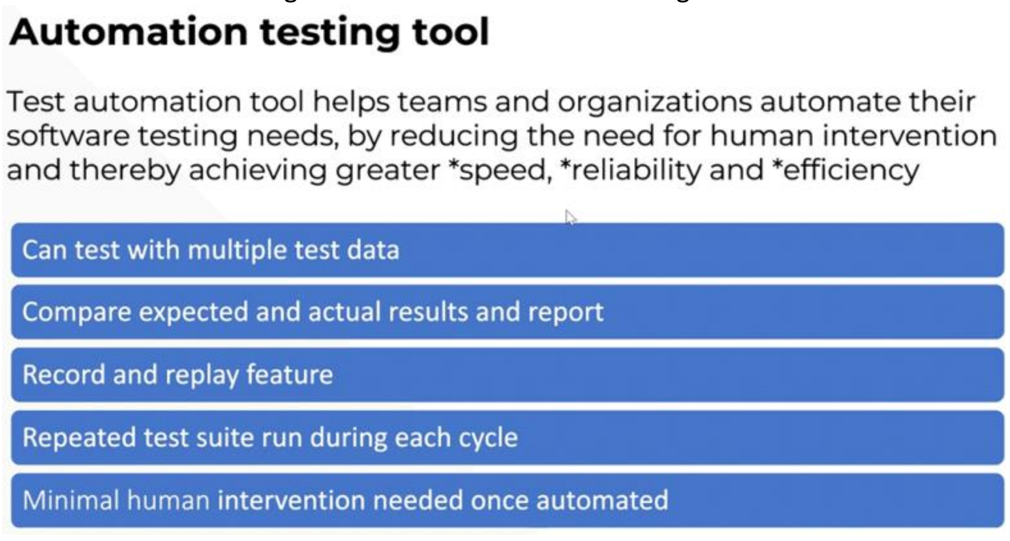


Test Automation

Test automation is the process of leveraging automation tools to maintain test data, execute tests, and analyse test results to improve software quality.

01
Delivers quality software products
02
Increases release velocity through CI/CD
03
Reduces human errors as human tasks are automated
04
Ensures cost optimization
05
Maximizes ROI
06
Ensures faster time to market

- If we want to test the application repeatedly , we use test automation. If we use manual testing we need many resources , and it will be use less because we are doing same work repeatedly.
- Automated testing is faster than the manual testing.



Automation testing tool

Test automation tool helps teams and organizations automate their software testing needs, by reducing the need for human intervention and thereby achieving greater *speed, *reliability and *efficiency

- Can test with multiple test data
- Compare expected and actual results and report
- Record and replay feature
- Repeated test suite run during each cycle
- Minimal human intervention needed once automated

- Reliability in the sense believability.

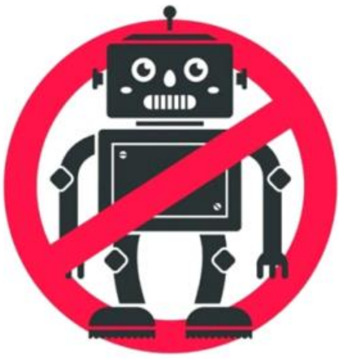
What to automate?

- Business Critical test cases
- Repeatedly executing test cases
- Test Cases that are very tedious or difficult to perform manually
- The test is time consuming
- Test has significant downtime between steps
- Test is subjective to human errors

- Business critical test cases in the sense , the test cases which are critical for the applications, such as automated testing on login page or transaction or payment page etc.

What not to Automate?

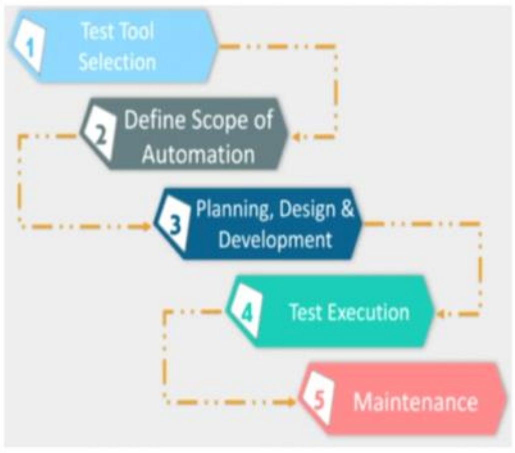
- Scenarios that frequently change
- Ad-hoc scenarios
- One time testing scenarios
- User experience validations
- Newly designed slides - not manually executed



AD HOC Scenarios is like sample testing that we do when we visit the site for the first time.

Automated Testing Process

- Test tool depends on the technology the AUT is built on
- Scope defines area of your AUT that will be automated
- Planning phase includes choosing tools, framework and Data for automation
- Scripts will be run during the execution phase
- Maintenance deals with changes in script needed, based on new development and enhancement

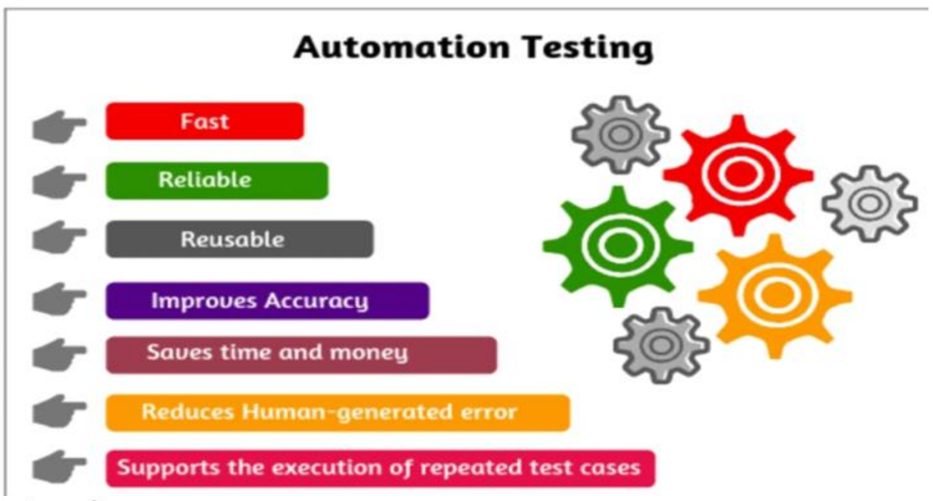

1 Test Tool Selection
2 Define Scope of Automation
3 Planning, Design & Development
4 Test Execution
5 Maintenance

Automation tool selection

Tool must not be selected based on popularity, but should fit the automation testing requirements

- Ease of Developing and Maintaining the Scripts
- Ease of Test Execution for Non-Technical user
- Support to Web, Desktop & Mobile application
- Intuitive Test Report
- Browser and language support
- Technical Support and Assistance
- Available budget, time and resources
- Interaction with external devops tools

Benefits of Automation



Selenium

- To automate web application
- Open-source
- Cross browser support (Firefox, Chrome, IE, Opera, Safari etc...)
- Multiple OS support (MS Windows, Linux, Macintosh etc...)
- Supports many programming languages (Java,C#,Python,Ruby,JavaScript,and Kotlin)

Disadvantage

- It supports Web-based applications only.
- Compared to paid tools, takes more knowledge and efforts to setup and use
- Limited support for Image Testing.
- No Test Tool integration for Test Management.
- No Built-in Reporting facility

Appium

- Open-source
- Automates native, hybrid, web mobile apps
- Automates both Android and iOS devices
- Automates both real devices and emulators
- Built on top of selenium
- Multiple language support
- Based on client server architecture

Disadvantage

- Initial setup time for iOS and Android takes time
- No support for Android 4.1 or lower
- Need Mac machine to automate iOS devices

Katalon Studio

- All-in-one test automation solution
- Supports both web and mobile app development
- Works on top of Selenium and Appium
- Simplifies API, web, and mobile automation tests
- Possible to integrate with other tools

Features of Katalon

- License: Proprietary
- Cross platform support
- Many built-in keywords
- Less technical skill required
- User-friendly GUI
- Integrated with CI/DevOps workflow
- Intuitive analytics dashboard and reports

Disadvantage

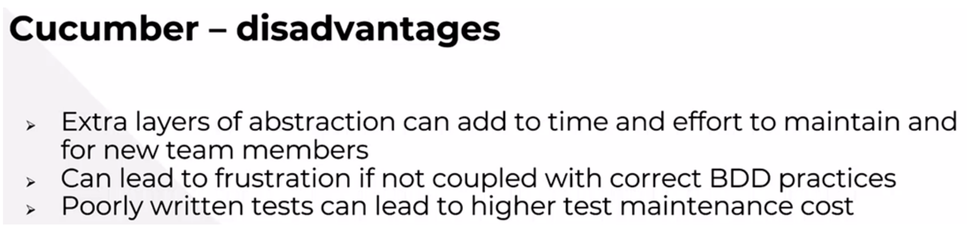
Script creation is limited to Java and Groovy
No support for Distributed testing
cannot automate desktop applications

Cucumber

- Open-source
- Behavior Driven Development (BDD) tool
- Supports only the web environment
- Enhanced end-user experience
- Easy setup and execution

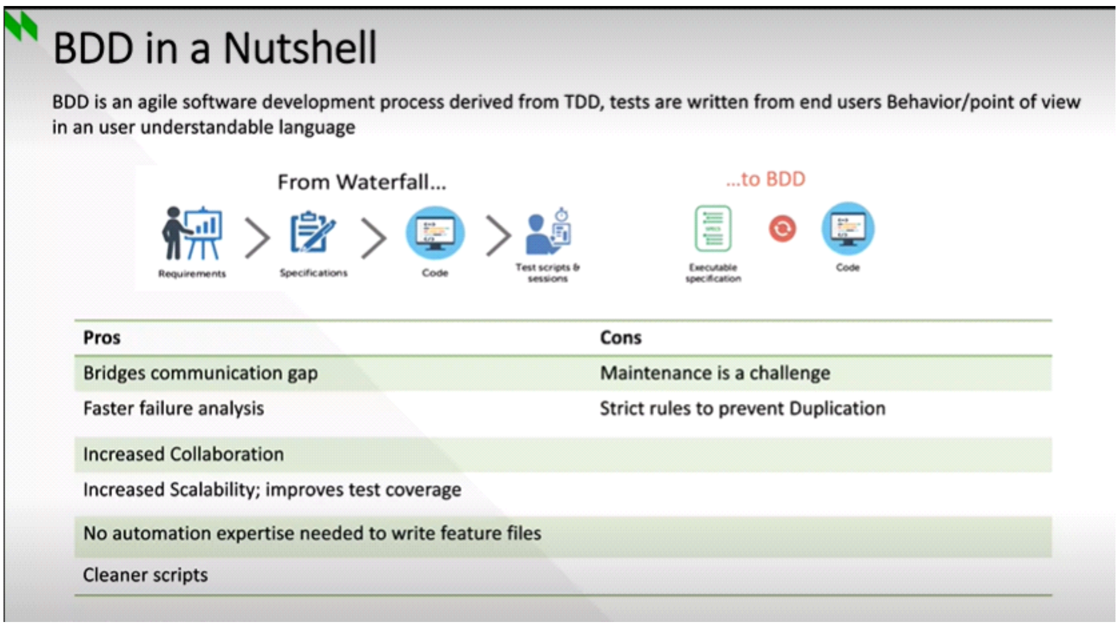
Features:

- Supports different frameworks like Selenium
- Supports languages like Ruby, Java, Scala, Groovy, etc.



Cucumber – disadvantages

- Extra layers of abstraction can add to time and effort to maintain and for new team members
- Can lead to frustration if not coupled with correct BDD practices
- Poorly written tests can lead to higher test maintenance cost



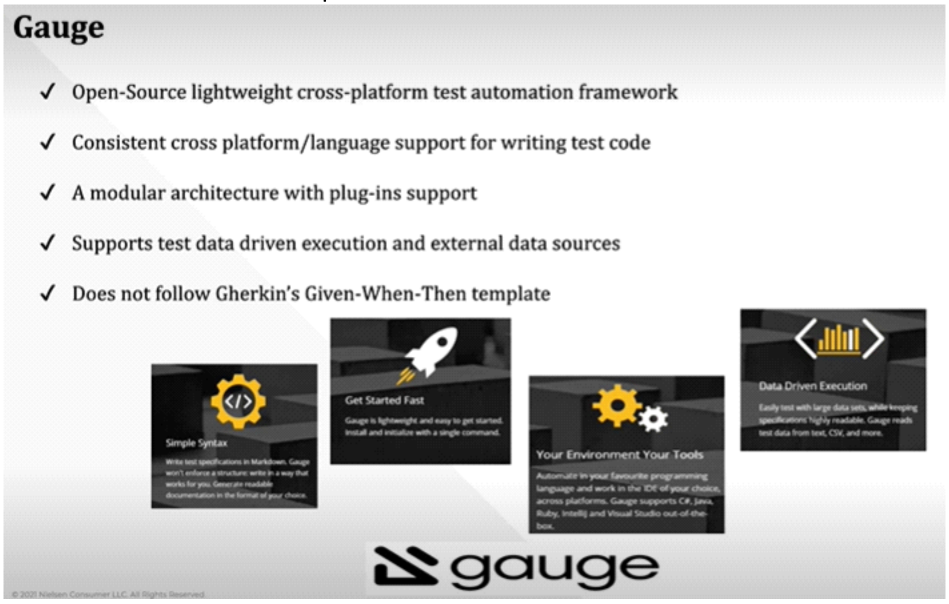
BDD in a Nutshell

BDD is an agile software development process derived from TDD, tests are written from end users Behavior/point of view in an user understandable language

From Waterfall...
Requirements
Specifications
Code
Test scripts & sessions
...to BDD
Executable specification
Code

Pros	Cons
Bridges communication gap	Maintenance is a challenge
Faster failure analysis	Strict rules to prevent Duplication
Increased Collaboration	
Increased Scalability; improves test coverage	
No automation expertise needed to write feature files	
Cleaner scripts	

BDD => Behavior driven development (The testing is done based on user perspective)
TDD => Test Driven Development



Gauge

✓ Open-Source lightweight cross-platform test automation framework

✓ Consistent cross platform/language support for writing test code

✓ A modular architecture with plug-ins support

✓ Supports test data driven execution and external data sources

✓ Does not follow Gherkin's Given-When-Then template

Simple Syntax
Write test specifications in Markdown. Gauge won't enforce a structure: write in a way that works for you. Generate readable documentation in the format of your choice.

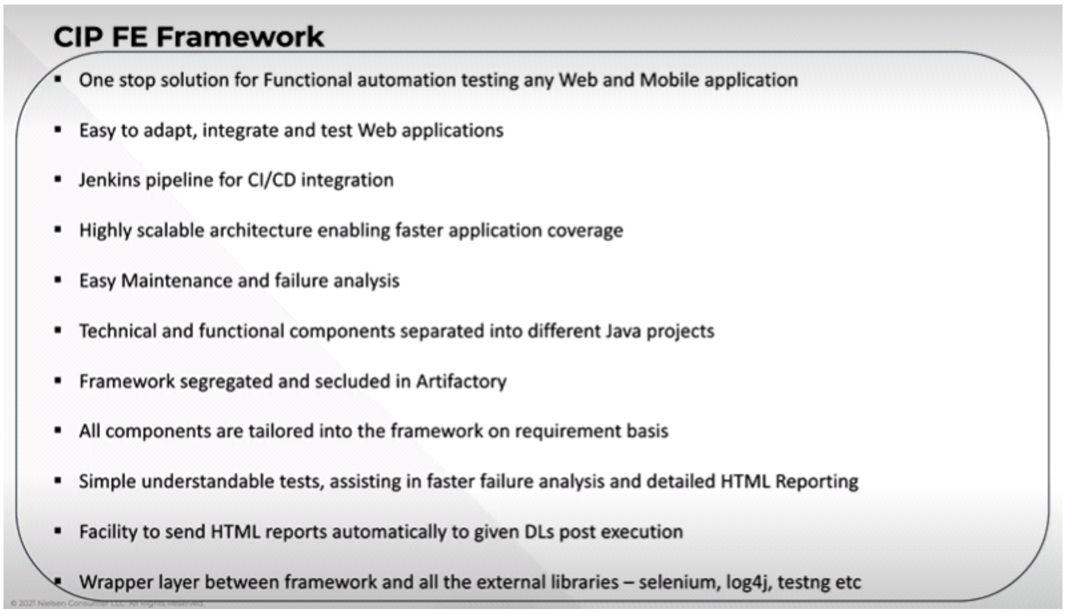
Get Started Fast
Gauge is lightweight and easy to get started. Install and initialize with a single command.

Your Environment Your Tools
Automate in your favourite programming language and work in the IDE of your choice, across platforms. Gauge supports C#, Java, Ruby, IntelliJ and Visual Studio out-of-the-box.

Data Driven Execution
Easily test with large data sets, while keeping specifications highly readable. Gauge reads test data from text, CSV, and more.

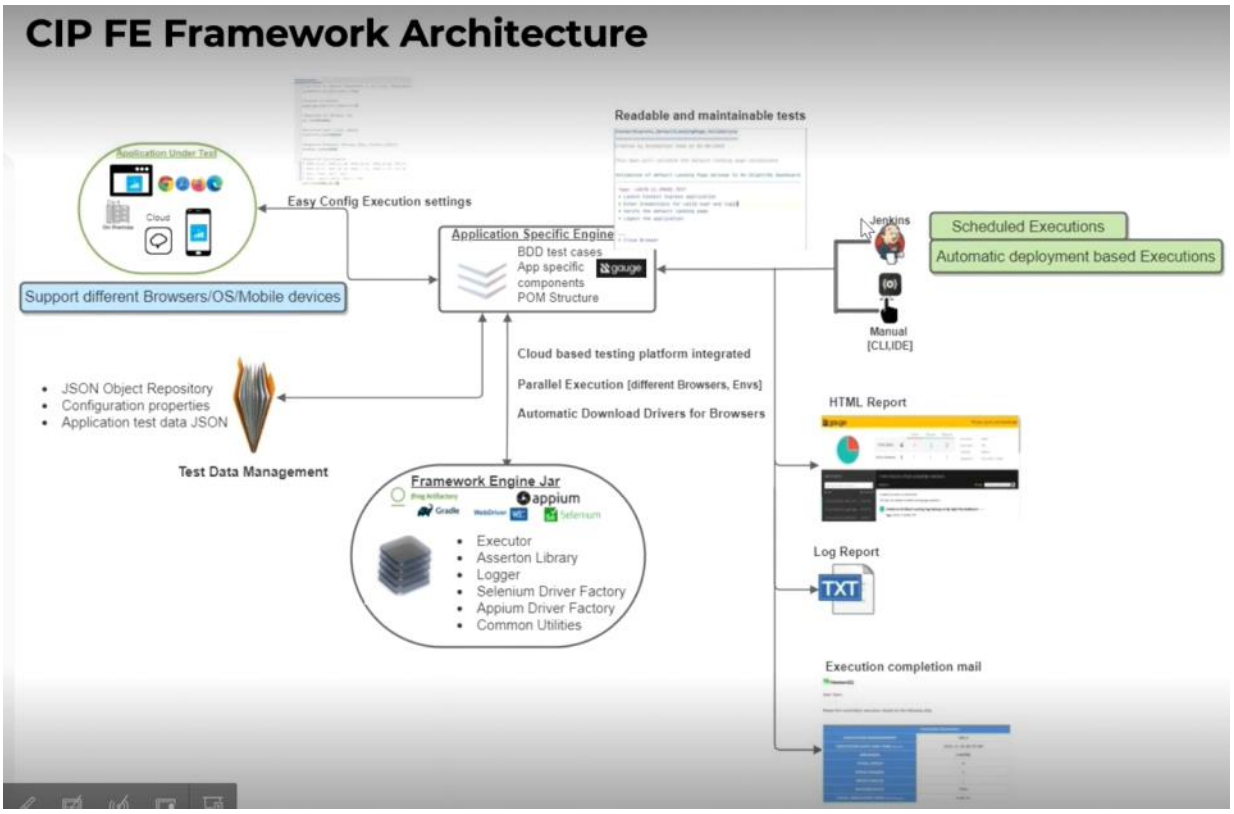
gauge

- The language that we use in gauge is similar to the natural language that we use, it is close to English . So , everyone can have an idea on it . Either technical or non technical.



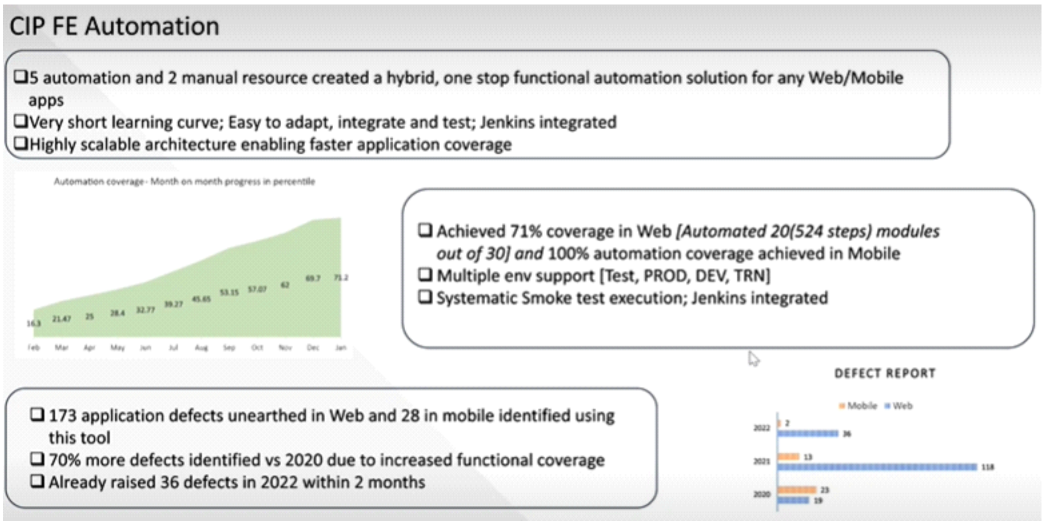
CIP FE Framework

- One stop solution for Functional automation testing any Web and Mobile application
- Easy to adapt, integrate and test Web applications
- Jenkins pipeline for CI/CD integration
- Highly scalable architecture enabling faster application coverage
- Easy Maintenance and failure analysis
- Technical and functional components separated into different Java projects
- Framework segregated and secluded in Artifactory
- All components are tailored into the framework on requirement basis
- Simple understandable tests, assisting in faster failure analysis and detailed HTML Reporting
- Facility to send HTML reports automatically to given DLs post execution
- Wrapper layer between framework and all the external libraries – selenium, log4j, testng etc



CIP FE Framework Architecture

- Executions started at Jenkins or CI or IDE level and then all the BDD testcases and app specific test cases will be executed and interact with the test data management for the test cases and for utilizing the frameworks for testing it searches the jar file and deploy them on the different OS/android versions or mobiles , to see the performance and give back the results in format of HTML report , log report and through automatic mails like execution completion mails.



CIP FE Automation

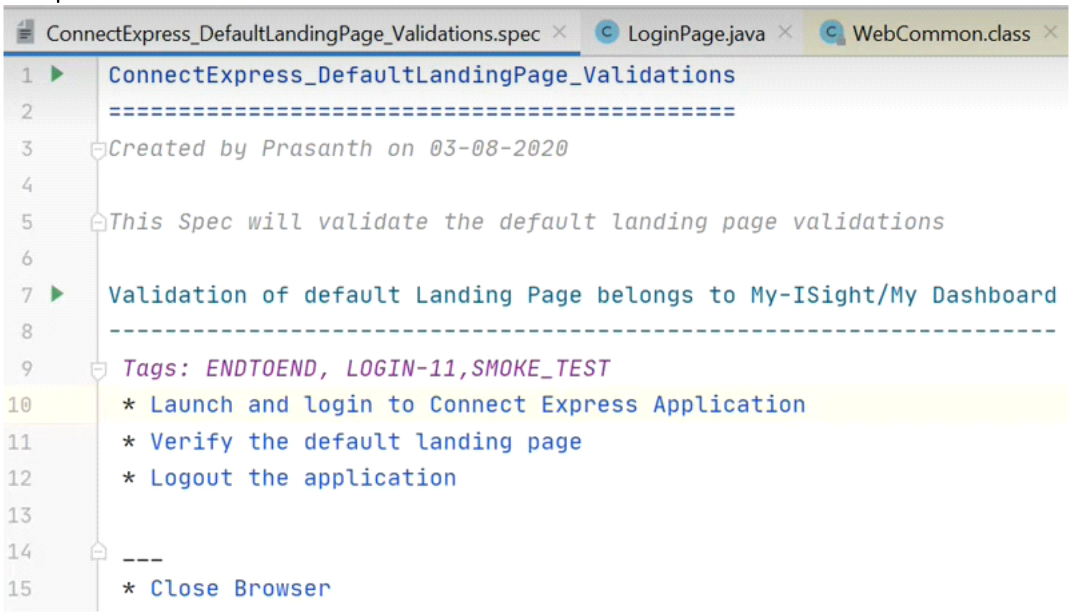
- ❑5 automation and 2 manual resource created a hybrid, one stop functional automation solution for any Web/Mobile apps
- ❑Very short learning curve; Easy to adapt, integrate and test; Jenkins integrated
- ❑Highly scalable architecture enabling faster application coverage

Automation coverage- Month on month progress in percentile
16.3
21.47
25
28.4
32.77
39.27
45.65
53.15
57.07
62
69.7
71.2
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan

- ❑ Achieved 71% coverage in Web *[Automated 20(524 steps) modules out of 30] and* 100% automation coverage achieved in Mobile
- ❑ Multiple env support [Test, PROD, DEV, TRN]
- ❑ Systematic Smoke test execution; Jenkins integrated

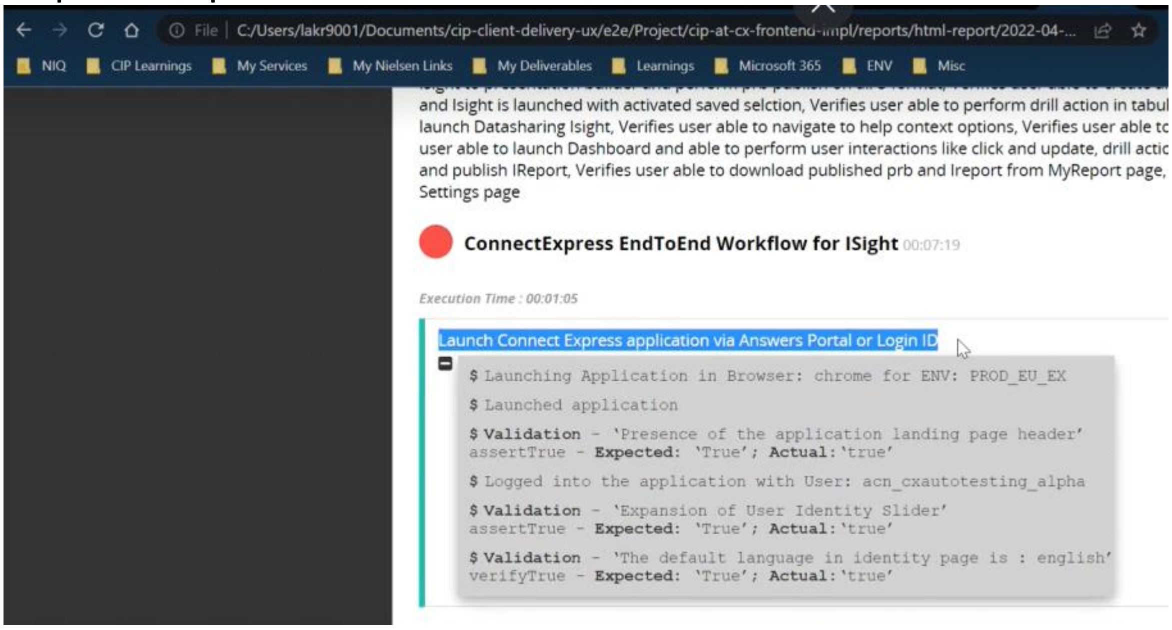
- ❑ 173 application defects unearthed in Web and 28 in mobile identified using this tool
- ❑ 70% more defects identified vs 2020 due to increased functional coverage
- ❑ Already raised 36 defects in 2022 within 2 months

Sample BDD



```
ConnectExpress_DefaultLandingPage_Validations
=============================================
Created by Prasanth on 03-08-2020

This Spec will validate the default landing page validations

Validation of default Landing Page belongs to My-ISight/My Dashboard
--------------------------------------------------------------------
 Tags: ENDTOEND, LOGIN-11,SMOKE_TEST
 * Launch and login to Connect Express Application
 * Verify the default landing page
 * Logout the application

 ___
 * Close Browser
```

Sample HTML Reports



File
C:/Users/lakr9001/Documents/cip-client-delivery-ux/e2e/Project/cip-at-cx-frontend-impl/reports/html-report/2022-04-...
NIQ
CIP Learnings
My Services
My Nielsen Links
My Deliverables
Learnings
Microsoft 365
ENV
Misc

and Isight is launched with activated saved selction, Verifies user able to perform drill action in tabul
launch Datasharing Isight, Verifies user able to navigate to help context options, Verifies user able tc
user able to launch Dashboard and able to perform user interactions like click and update, drill actic
and publish IReport, Verifies user able to download published prb and Ireport from MyReport page,
Settings page

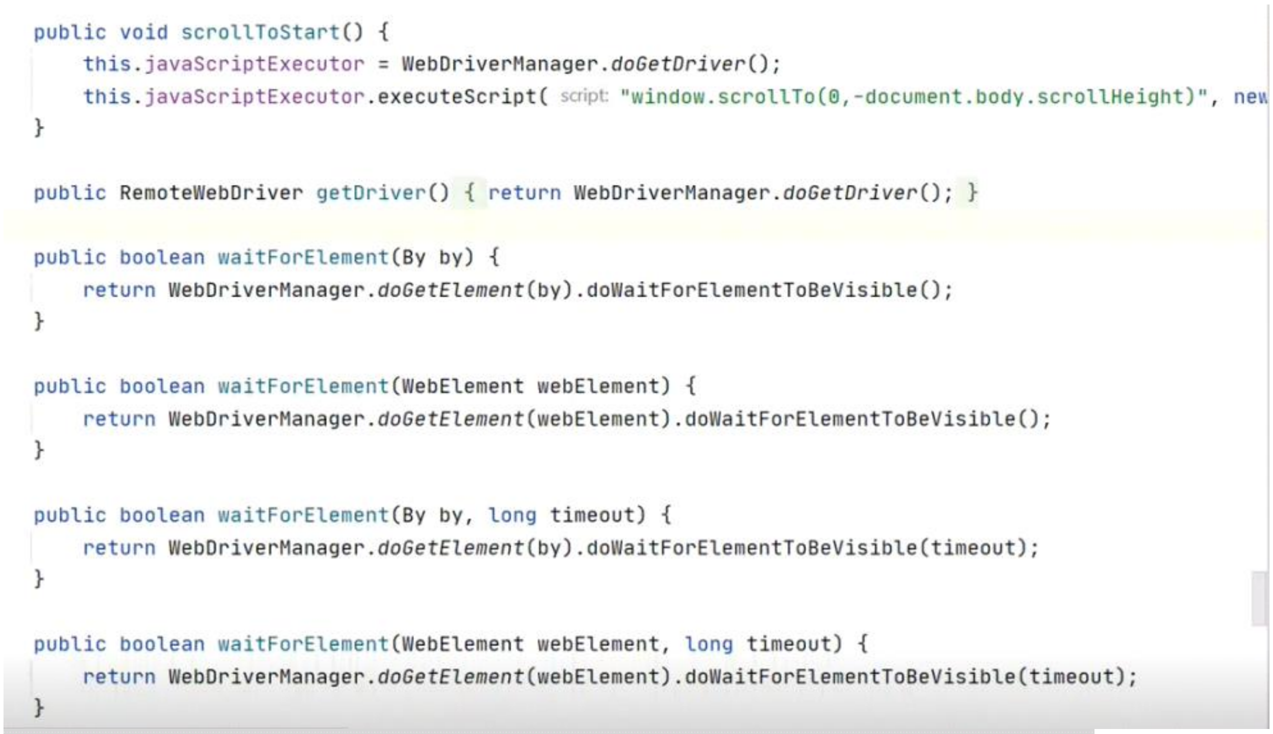
ConnectExpress EndToEnd Workflow for ISight 00:07:19

Execution Time : 00:01:05

Launch Connect Express application via Answers Portal or Login ID

```
$ Launching Application in Browser: chrome for ENV: PROD_EU_EX
$ Launched application
$ Validation - 'Presence of the application landing page header'
assertTrue - Expected: 'True'; Actual:'true'
$ Logged into the application with User: acn_cxautotesting_alpha
$ Validation - 'Expansion of User Identity Slider'
assertTrue - Expected: 'True'; Actual:'true'
$ Validation - 'The default language in identity page is : english'
verifyTrue - Expected: 'True'; Actual:'true'
```

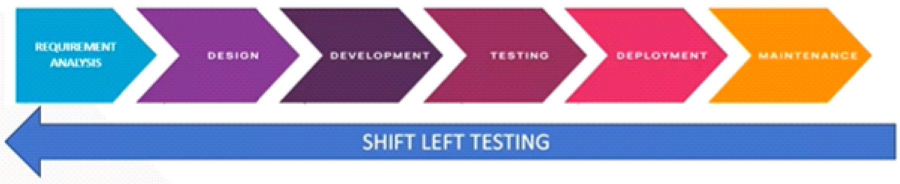
- All the logic is written in java codes

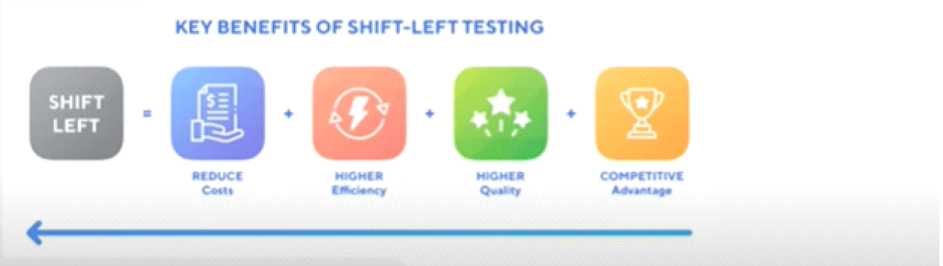


```
public void scrollToStart() {
    this.javaScriptExecutor = WebDriverManager.doGetDriver();
    this.javaScriptExecutor.executeScript( script: "window.scrollTo(0,-document.body.scrollHeight)", new
}

public RemoteWebDriver getDriver() { return WebDriverManager.doGetDriver(); }

public boolean waitForElement(By by) {
    return WebDriverManager.doGetElement(by).doWaitForElementToBeVisible();
}

public boolean waitForElement(WebElement webElement) {
    return WebDriverManager.doGetElement(webElement).doWaitForElementToBeVisible();
}

public boolean waitForElement(By by, long timeout) {
    return WebDriverManager.doGetElement(by).doWaitForElementToBeVisible(timeout);
}

public boolean waitForElement(WebElement webElement, long timeout) {
    return WebDriverManager.doGetElement(webElement).doWaitForElementToBeVisible(timeout);
}
```

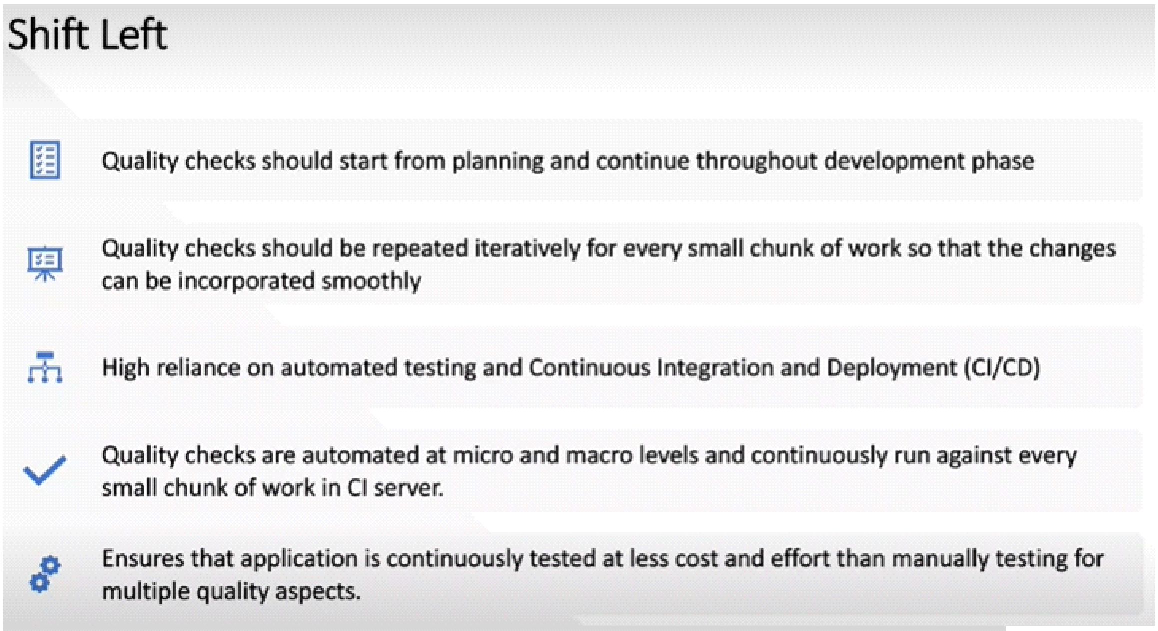
Shift Left

- Shift – left Testing - Shifting testing activities to beginning of cycle resulting in high-quality results





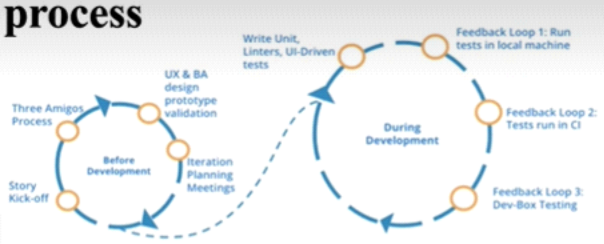
- In Shift left testing the shifting is done at every phase of development , so process of development and the risk of the malware decreases. This separates us from our competitors as the development is fast and risk of malware is low.
- We try to automate everything in such testing to save the resources.



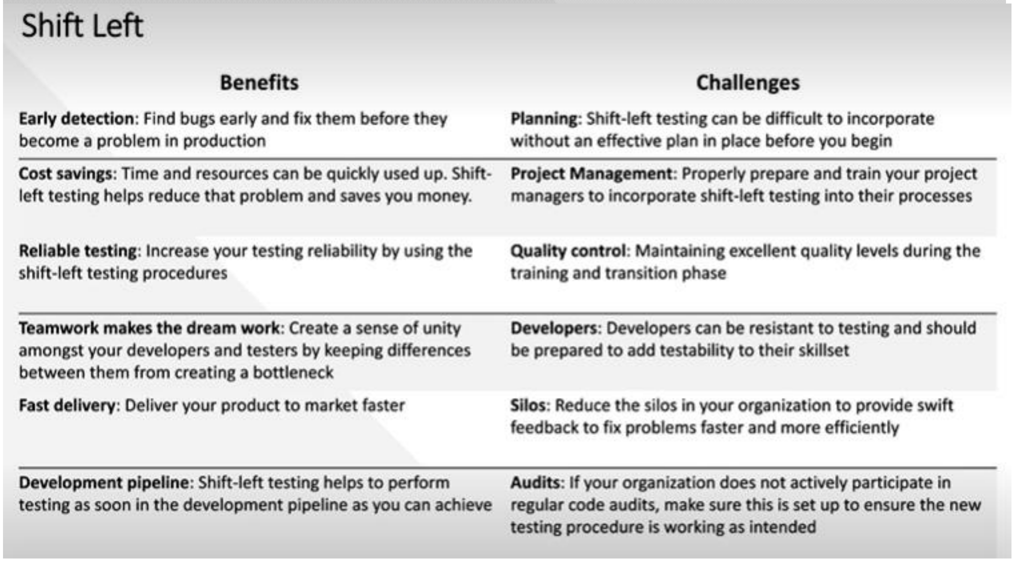
Shift Left

- Quality checks should start from planning and continue throughout development phase
- Quality checks should be repeated iteratively for every small chunk of work so that the changes can be incorporated smoothly
- High reliance on automated testing and Continuous Integration and Deployment (CI/CD)
- Quality checks are automated at micro and macro levels and continuously run against every small chunk of work in CI server.
- Ensures that application is continuously tested at less cost and effort than manually testing for multiple quality aspects.

Shift Left process


Three Amigos Process
UX & BA design prototype validation
Iteration Planning Meetings
Story Kick-off
Before Development
Write Unit, Linters, UI-Driven tests
Feedback Loop 1: Run tests in local machine
Feedback Loop 2: Tests run in CI
Feedback Loop 3: Dev-Box Testing
During Development

- Three amigos process - Ceremony conducted in the analysis phase to discuss the feature thoroughly
- Iteration planning meeting (IPM) - conducted at beginning of iteration/sprint to discuss the user stories in detail
- Story kickoff - A minified version of three amigos with more focus on specific user story's requirements and edge cases
- Test creation - Unit test are written; Post development, UI-driven functional tests written and integrated with CI
- Execution – Before commit developers run, post every commit tests (unit, service, UI, etc.) run automatically
- Dev-box testing - Final level, quick round of manual exploratory testing on the newly developed functionality

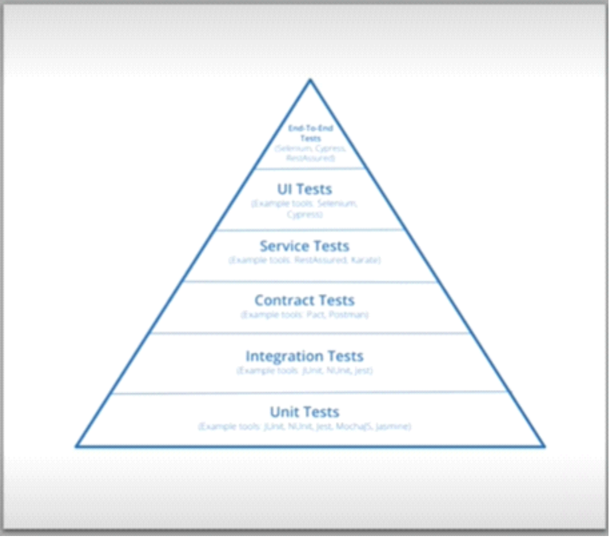


Shift Left

Benefits	Challenges
Early detection: Find bugs early and fix them before they become a problem in production	**Planning**: Shift-left testing can be difficult to incorporate without an effective plan in place before you begin
Cost savings: Time and resources can be quickly used up. Shift-left testing helps reduce that problem and saves you money.	**Project Management**: Properly prepare and train your project managers to incorporate shift-left testing into their processes
Reliable testing: Increase your testing reliability by using the shift-left testing procedures	**Quality control**: Maintaining excellent quality levels during the training and transition phase
Teamwork makes the dream work: Create a sense of unity amongst your developers and testers by keeping differences between them from creating a bottleneck	**Developers**: Developers can be resistant to testing and should be prepared to add testability to their skillset
Fast delivery: Deliver your product to market faster	**Silos**: Reduce the silos in your organization to provide swift feedback to fix problems faster and more efficiently
Development pipeline: Shift-left testing helps to perform testing as soon in the development pipeline as you can achieve	**Audits**: If your organization does not actively participate in regular code audits, make sure this is set up to ensure the new testing procedure is working as intended

Testing Pyramid

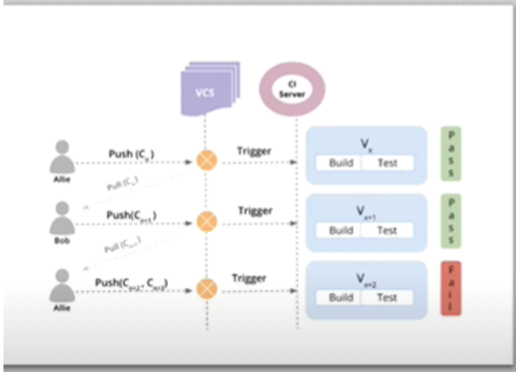
- Test Pyramid strategy when applied to automated testing helps with tests that validate right scope of the functionality in the right layers of the application yielding fastest feedback to the team
- This recommends having a broad bucket of micro-level tests and gradually reducing the macro-level tests as its scope increases
- As the scope of the tests increases, they take more time to run and cost more to write and maintain.

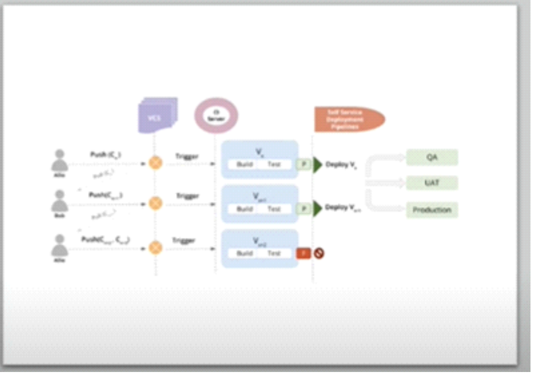

End-To-End Tests
(Selenium, Cypress, RestAssured)
UI Tests
(Example tools: Selenium, Cypress)
Service Tests
(Example tools: RestAssured, Karate)
Contract Tests
(Example tools: Pact, Postman)
Integration Tests
(Example tools: JUnit, NUnit, Jest)
Unit Tests
(Example tools: JUnit, NUnit, Jest, MochaJS, Jasmine)

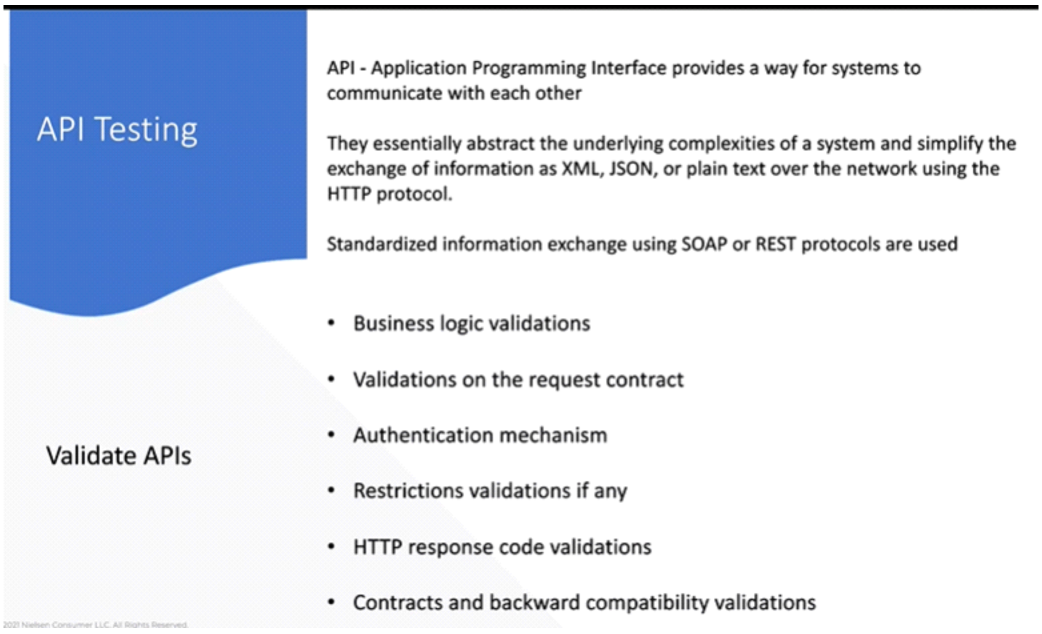
- In top layers of testing pyramid we have End to end and UI testing where those take more time to test because they have to go around the application and test it. Each testcase may take up to 10 mins , so at top layers , we must not have more number of test cases as they are time consuming , so only important test cases must be considered limitedly. If we have 100 Unit Test cases we must only have 10 End to end test cases. Otherwise we loose the time balance in the product development

Continuous Testing

- The process of validating the application quality for every incremental change and alerting the team when the result changes/deviates from the intended outcomes It relies heavily on *Continuous Integration* (CI) practice to perform automated testing against every change.







API Testing

API - Application Programming Interface provides a way for systems to communicate with each other

They essentially abstract the underlying complexities of a system and simplify the exchange of information as XML, JSON, or plain text over the network using the HTTP protocol.

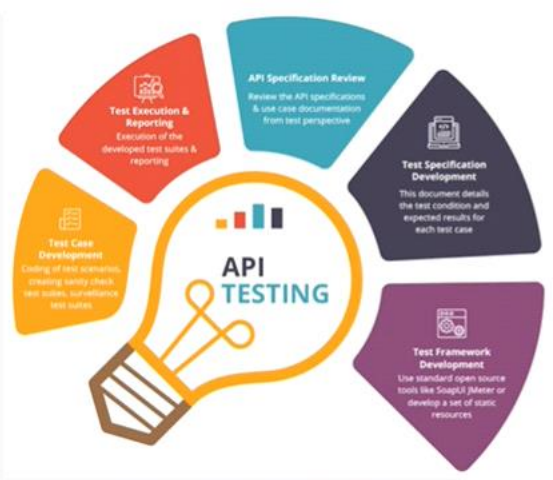
Standardized information exchange using SOAP or REST protocols are used

Validate APIs

- Business logic validations
- Validations on the request contract
- Authentication mechanism
- Restrictions validations if any
- HTTP response code validations
- Contracts and backward compatibility validations

The following tools are used for validation of API

API Testing process


API Specification Review
Review the API specifications & use case documentation from test perspective
Test Specification Development
This document details the test condition and expected results for each test case
Test Framework Development
Use standard open source tools like SoapUI JMeter or develop a set of static resources
Test Case Development
Coding of test scenarios, creating sanity check test suites, surveillance test suites
Test Execution & Reporting
Execution of the developed test suites & reporting
API
TESTING

API Testing tools

Katalon Studio

ACCELQ

POSTMAN

REST-Assured

SMARTBEAR ReadyAPI

SWAGGER SMARTBEAR

Mobile Testing

Mobile device testing is the process by which mobile apps are tested
for functionality, usability, and consistency. Testing app on mobile devices can be done manually or with automation.

Type of mobile apps:
Native - Native apps are developed mainly to work on a single mobile OS such as Android or iOS. They provide excellent performance, access to all OS features and APIs, can work offline, and can deliver the look and feel to perfection

Mobile Web - Mobile web apps are websites that are accessed using mobile web browsers. They have the advantages of being
OS independent, free of installation procedures, and no local storage space for installation needed

Hybrid - best from both. A hybrid app is developed using standard web development frameworks such as HTML, JS, or CSS and is then wrapped with mobile dev frameworks such as Cordova

Progressive Web - Advanced version of mobile web, can be installed on the phone using URL and take very little storage space. It can provide push notifications, can work offline, and has access to OS features, making the experience like native app.

Mobile testing types

Manual Exploratory Testing - Exploratory testing becomes extremely crucial in the mobile landscape due to the myriad of the device, app, and network combinations as discussed earlier

Functional Automated Testing - Functional behavior of app and interactions can be automated using unit and UI-driven functional end-to-end tests. Just like web functional tests they need to be plugged to the CI for getting continuous feedback.

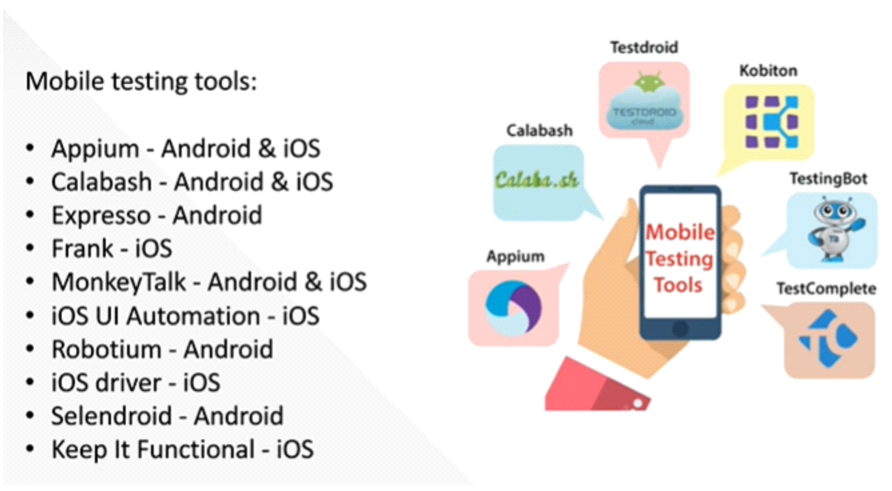
Data Testing – With mobile apps, data can be stored in various layers for different purposes. like local mobile DB, a common database, and the local device storage etc. It is essential to understand the data flow and include them in testing

Visual Testing - Covered when device is tested manually, just like in Web and can also be automated for different screen sizes using Appium and Applitools etc.

Security - Users sensitive data should be encrypted and stored, strong authentication mechanisms, appropriate permissions to access other apps on the phone etc. should be thought of during testing.

Performance - For mobile UI layer, testing should concentrate on, app not clogging or depleting device's critical resources like CPU, memory, and battery power and responsiveness

Accessibility - App should be perceivable, operable, understandable, and robust, some features to test might include ability to zoom in and out, readability in small screen sizes, color contrast between elements, click space for the buttons, layout consistency etc.

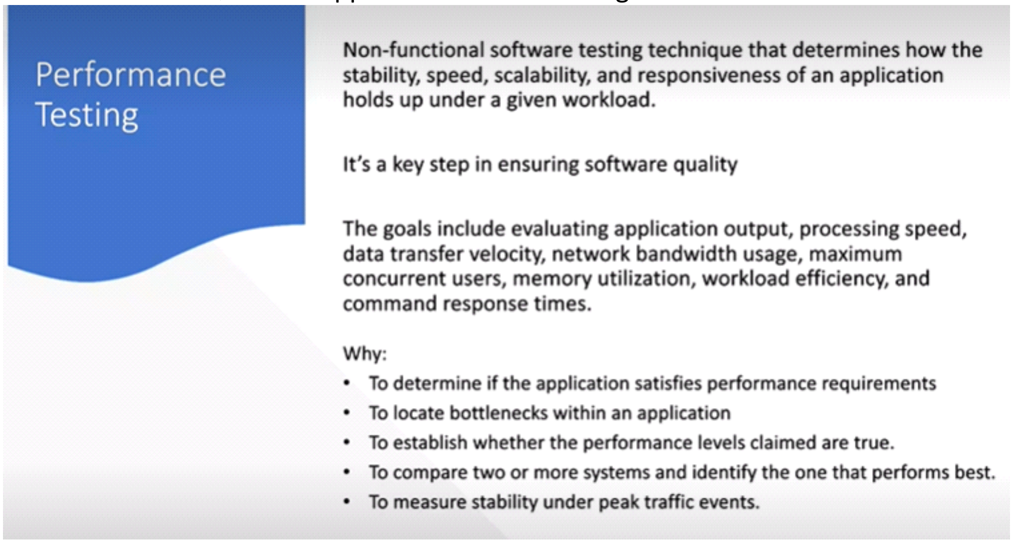


Mobile testing tools:

- Appium - Android & iOS
- Calabash - Android & iOS
- Expresso - Android
- Frank - iOS
- MonkeyTalk - Android & iOS
- iOS UI Automation - iOS
- Robotium - Android
- iOS driver - iOS
- Selendroid - Android
- Keep It Functional - iOS

Testdroid
Kobiton
Calabash
TestingBot
Appium
Mobile Testing Tools
TestComplete

- In NielsenIQ we use Appium for mobile testing.



Performance Testing

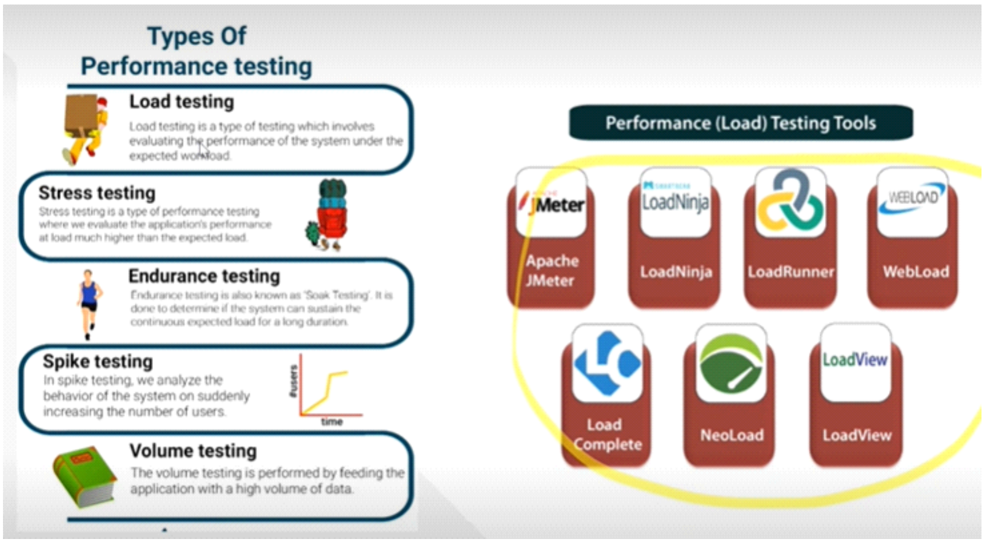
Non-functional software testing technique that determines how the stability, speed, scalability, and responsiveness of an application holds up under a given workload.

It's a key step in ensuring software quality

The goals include evaluating application output, processing speed, data transfer velocity, network bandwidth usage, maximum concurrent users, memory utilization, workload efficiency, and command response times.

Why:

- To determine if the application satisfies performance requirements
- To locate bottlenecks within an application
- To establish whether the performance levels claimed are true.
- To compare two or more systems and identify the one that performs best.
- To measure stability under peak traffic events.



Types Of Performance testing

Load testing
Load testing is a type of testing which involves evaluating the performance of the system under the expected workload.

Stress testing
Stress testing is a type of performance testing where we evaluate the application's performance at load much higher than the expected load.

Endurance testing
Endurance testing is also known as 'Soak Testing'. It is done to determine if the system can sustain the continuous expected load for a long duration.

Spike testing
In spike testing, we analyze the behavior of the system on suddenly increasing the number of users.

Volume testing
The volume testing is performed by feeding the application with a high volume of data.

Performance (Load) Testing Tools

Security Testing

- *Security testing* is the skill to think like a hacker and identify potential vulnerabilities, threats, and risks in the system. Security testers write scripts that does different attacks on the application to expose the vulnerabilities.

Man in the middle
Social Engineering
Spoo
Phishing
Shoulder surfing
Tam
CryptoJacking
Intellectual property theft
Repud
Brute force
Ransomware
Inform
Malware
Viruses, trojan horses
Denia
Code Injection
SQL Injection
Botnets
Escal

Security testing types

Static Application Security Testing (SAST) - Analyzing static application source code, byte code and assembled code for known vulnerabilities like scanning code for unencrypted secrets etc

Source Composition Analysis (SCA) - SCA is the technique that identifies the vulnerabilities in the application's third-party dependencies like when open-source libraries, these tools (e.g. OWASP Dependency Check, Snyk Open source)

Image Scanning - Containers have become a widely adopted way to package and deploy applications. Hence testing for vulnerabilities in the container images is critical. Tools like Snyk Container, Anchore, etc., can be used

Dynamic Application Security Testing (DAST) - Black-box testing technique that finds security issues by analyzing how application responds to specially crafted requests that mimic actual attacks. Tools like OWASP ZAP, Burp Suite try to inject malicious scripts into the application to check for code injection vulnerability

Functional Security Tests Automation - Adding automated tests to cover the functional security test cases. 'system admin should not have access to edit the order' can be added as a service test

Manual Exploratory Testing – Derive security-related test cases from threat modeling exercises across all layers

Penetration (Pen) Testing - ethical hacking intentional launching of simulated cyberattacks that seek out exploitable vulnerabilities in computer systems, networks, websites, and applications

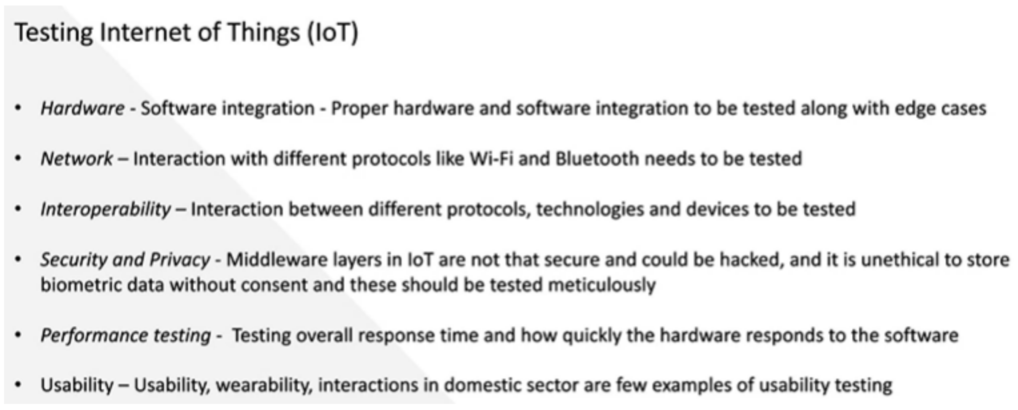
Runtime Application Self Protection (RASP) - For production env, to prevent successful attacks. Monitors application for potential attacks and prevents them from happening. It extends the traditional firewall concept further by working within the application runtime using tools like Twistlock, Aqua Security

Artificial Intelligence and Machine Learning - Testing of AI and ML

- *Validating training data* - When input data comes with different scales cleaning and transformation to a uniform scale should be tested; When input data have null or empty values, they either should be substituted with default values or eliminated during the cleaning stage – this should be validated
- *Validating model quality* - Model's quality is measured in terms of various metrics such as error rates, accuracy, confusion matrix, precision, and recall
- *Validating model bias* - If the input data has a large sample set representing a particular demography, the model will be biased towards that demography. It is critical to test both the input data and the ML model for biases
- Val*idating integrations* - Integrations between the three layers, specifically, the data & model and model & API layers, should be tested using the regular contract and integration testing approaches.

Testing Blockchain

- *Functional Testing* - The functional logic is written in the smart contracts – that needs to be tested
- *API Testing* - Focus on the standard API layer testing like functional testing, integration between modules, contract versioning, error handling, retries etc
- *Security Testing* - Checking for illegitimate transactions, authorization mechanisms, and the cryptography aspects like hashing the blocks which need to be tested
- Performance testing - Performance of a transaction and the functional behavior to handle the delay have to be tested.



Testing Internet of Things (IoT)

- *Hardware* - Software integration - Proper hardware and software integration to be tested along with edge cases
- *Network* – Interaction with different protocols like Wi-Fi and Bluetooth needs to be tested
- *Interoperability* – Interaction between different protocols, technologies and devices to be tested
- *Security and Privacy* - Middleware layers in IoT are not that secure and could be hacked, and it is unethical to store biometric data without consent and these should be tested meticulously
- *Performance testing* - Testing overall response time and how quickly the hardware responds to the software
- Usability – Usability, wearability, interactions in domestic sector are few examples of usability testing

Manual and Automated Testing

Wednesday, April 27, 2022 4:28 PM

Agile is an iterative software development methodology wherein requirements evolve through collaboration between the customer and self-organizing teams. It is not sequential, but rather a continuous process.

In an Agile environment, Agile testing is the only way to ensure continuous delivery of a product, as it provides feedback on an ongoing basis, allowing the product to meet business needs and satisfy customer requirements. Agile testing can be either manual or automated; these testing methods provide great benefits on their own, but when properly combined, can offer the best of both worlds.

What is the benefit in agile?
The Agile methodology focuses on integrating work teams, customers, and users to produce quality software in shorter time frames. Agile testing operates under the same philosophy and considers continuous testing as a crucial part of the development process.

- Clear defined and goals are achievable
- Clients are involved in development process, so there is high chance of customer satisfaction.
- Product quality will be improved , because bug tracking will be in each and every stage (whatever ratio) .
- Faster Software Product Releases (Because testing doesn't takes more time as we following continuous testing process).

Why integration of manual and automated testing is necessary?

- Sometimes we may have certain restrictions and constraints such as a high requirement to develop , less budget to utilize , less timeline to provide the end product , limited expertise to work on and expected high quality of product. In such situation , we must utilize the resources efficiently, we could not use skilled labor for routine testing purpose, which we can automate using computers. So the employee can be used for development purpose or any highly required testing purpose (Quality Analyst).

How integration of manual and automated testing happens?

- For the testcases which doesn't expose high security threat and the testcases which does take high time to check manually , we can automate those tasks . Because automation takes less processing time than manual testing.
- Sometimes , to check the testcases in user perspective. We need to do exploratory testing , and automated testing doesn't supports it, as it does require some expertise to solve we do it in manual testing.
- It does depend on budget of the project or the company, initial investment is higher for automation testing but rate of profit is also better for it than manual testing.
- Automated testing is more reliable than the manual testing because human testing may face human errors.
- We need to automate the tasks only if the product is almost build, because even if there is a small change in the user interface , it changes the entire script of automation. This doesn't any impact in case of manual testing.
- If there is significantly large amount of test cases to execute automation preferable.
- In use case scenario , automation testing is well suited for regression testing , load testing , highly repeatable functional test cases and manual is used for AdHoc testing , exploratory testing and cases where frequent changes are necessary.