

Software Testing



Software Testing



Works on the principle of
“test first,code later”

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.

Software Testing



Software testing refers to the process of validating and verifying the artifacts and behavior of the software under test. Software testing involves:

- ▶ **Verify and validate** that the product is **bug-free**.
- ▶ Determines if the product complies with the technical specifications, according to the design and development.
- ▶ Ensure the product **meets the user's requirements** efficiently and effectively.
- ▶ Measure the product's functionality and **performance**.
- ▶ Find ways to **improve** software efficiency, accuracy, and usability.

Software Testing



- ▶ In Software Testing, there is the difference between **Errors**, **Defects**, and **Bugs** that we should distinguish clearly to avoid misunderstanding problem.
- ▶ “A mistake in coding is called **Error**, **Error** found by the tester is called **Defect**, **Defect** accepted by development team then it is called Bug, build does not meet the requirements then it is a **Failure**.”

Importance of testing



- ▶ **Cost-Effective:** It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.
- ▶ **Security:** It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.
- ▶ **Product quality:** It is an essential requirement of any software product. Testing ensures a quality product is delivered to customers.
- ▶ **Customer Satisfaction:** The main aim of any product is to give satisfaction to their customers. UI/UX Testing ensures the best user experience.

Importance of testing



- ▶ Address issues such as:
 - ▶ Architectural flaws
 - ▶ Poor design decisions
 - ▶ Invalid or incorrect functionality
 - ▶ Security vulnerabilities
 - ▶ Scalability issues

Types of software testing

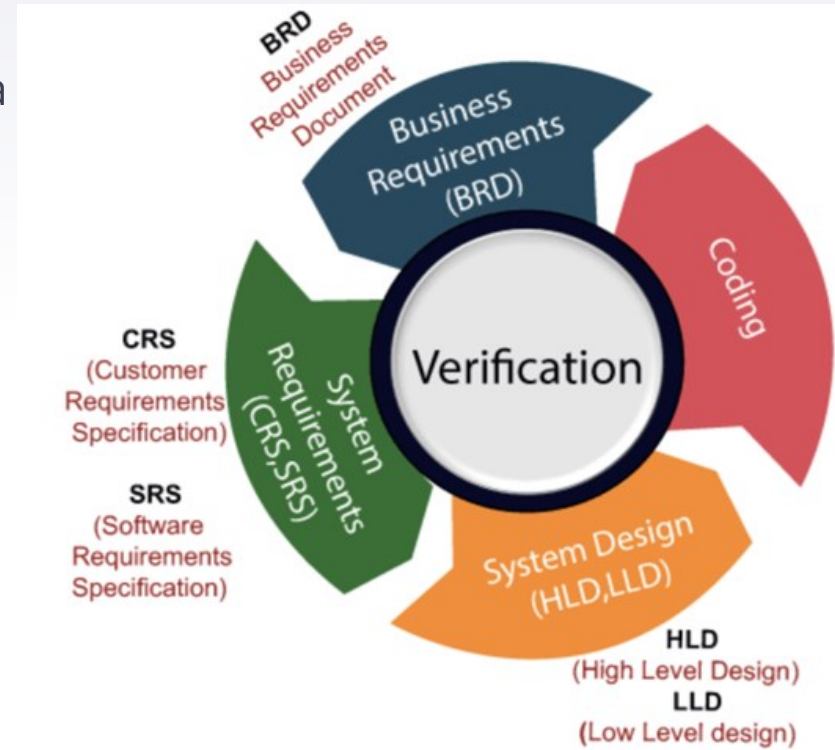


- ▶ **Unit testing:** Validating that each software unit performs as expected. A unit is the smallest testable component of an application.
- ▶ **Integration testing:** Ensuring that software components or functions operate together.
- ▶ **Functional testing:** Checking functions by emulating business scenarios, based on functional requirements. Black-box testing is a common way to verify functions.
- ▶ **Performance testing:** Testing how the software performs under different workloads. Load testing, for example, is used to evaluate performance under real-life load conditions.
- ▶ **Regression testing:** Checking whether new features break or degrade functionality. Sanity testing can be used to verify menus, functions and commands at the surface level, when there is no time for a full regression test.
- ▶ **Acceptance testing:** Verifying whether the whole system works as intended.
- ▶ **Usability testing:** Validating how well a customer can use a system or web application to complete a task.

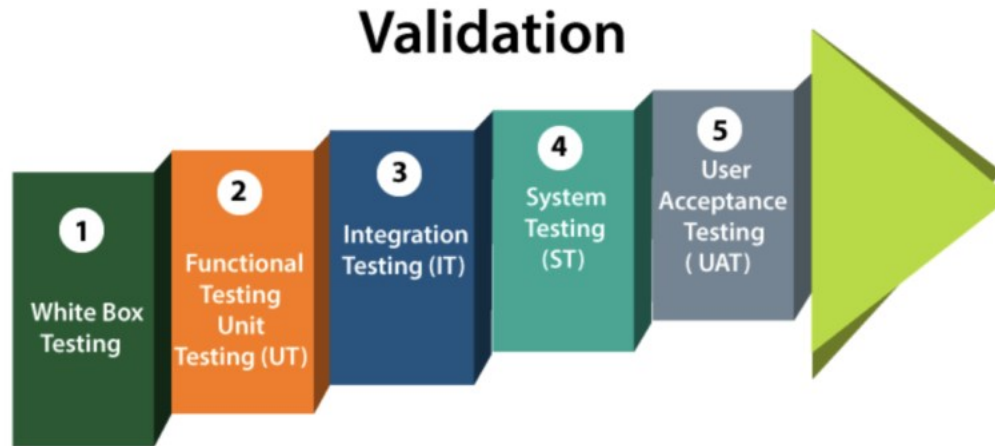
Verification and Validation



- ▶ **Verification** is the process of checking that a software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfills the requirements that we have. Verification is static testing.
- ▶ Verification means **Are we building the product right?**



- ▶ **Validation** is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product i.e. it checks what we are developing is the right product. it is validation of actual and expected product. Validation is the dynamic testing.
- ▶ Validation means **Are we building the right product?**



QA, QC, and Testing



- ▶ **Quality Assurance (QA)** is process oriented. It is all about preventing defects by ensuring the processes used to manage and create deliverables works. Not only does it work, but is consistently followed by the team. Moreover, QA is about engineering processes that assure quality is achieved in an effective and efficient way.
- ▶ For instance, if a defect is found and fixed, there is no guaranteeing it won't pop back up. The role of QA is to identify the process that allowed the error to occur and re-engineer the system so that these defects won't appear for the second time.
- ▶ Examples of QA include process definition and implementation, training, audits and selection of tools.

QA, QC, and Testing



- ▶ **Quality control**, alternatively, is product oriented. It is the function of software quality that determines the ending result is what was expected. Whereas QA is proactive, QC is reactive. QC detects bugs by inspecting and testing the product. This involves checking the product against a predetermined set of requirements and validating that the product meets those requirements.
- ▶ Examples of QC include technical reviews, software testing and code inspections.

QA, QC, and Testing



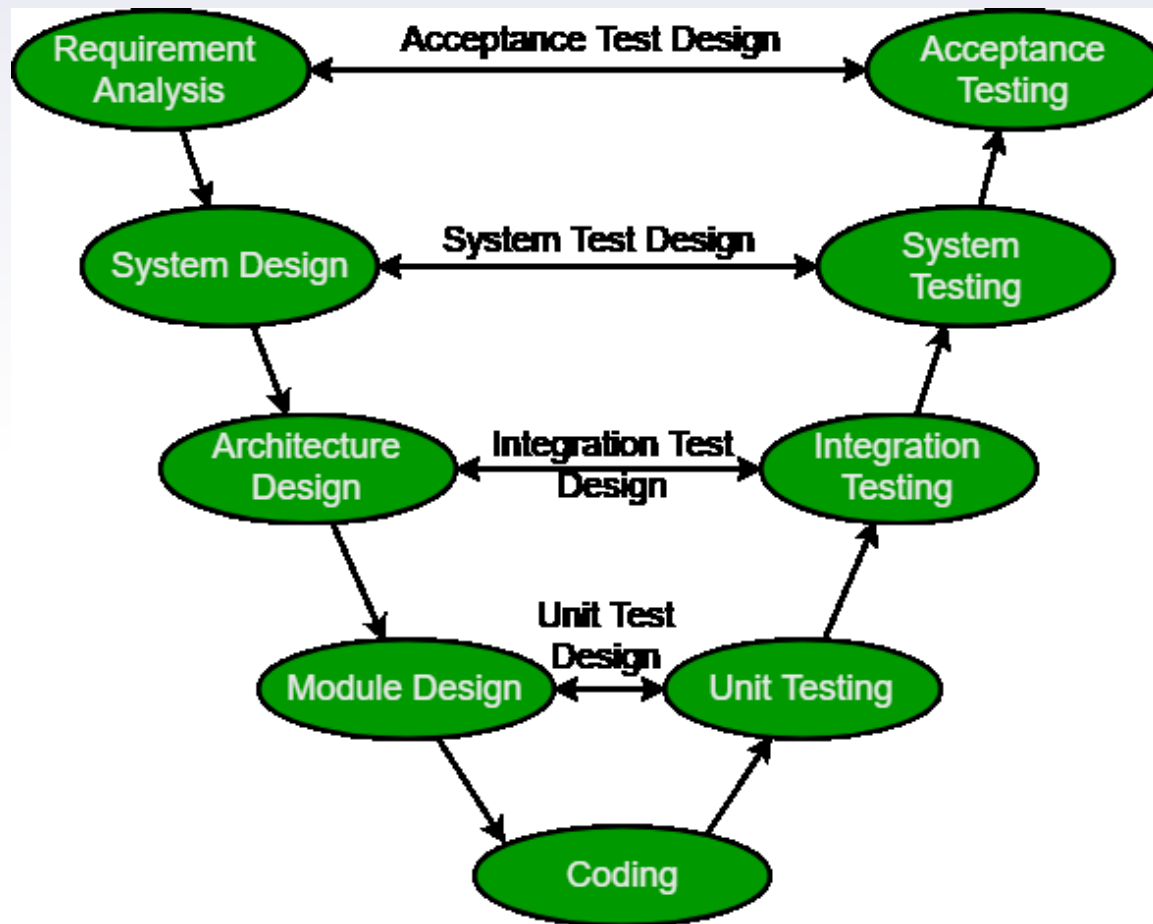
- ▶ **Testing** is a subset of QC. It is the process of executing a system in order to detect bugs in the product so that they get fixed. Testing is an integral part of QC as it helps demonstrate that the product runs the way it is expected and designed for.
- ▶ To summarize, think of everything as an assembly line. QA can be thought of as the process to ensure the assembly line actually works, while QC is when the products coming off the assembly line are checked to verify they meet the required specifications.
- ▶ Ultimately, both QA and QC are required for ensuring a successful product.

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

Introduction to STLC and V Model



- ▶ The V-model is a type of SDLC model where process executes in a sequential manner in V-shape. It is also known as Verification and Validation model. It is based on the association of a testing phase for each corresponding development stage. Development of each step directly associated with the testing phase. The next phase starts only after completion of the previous phase i.e. for each development activity, there is a testing activity corresponding to it.



Design Phase



- ▶ **Requirement Analysis:** This phase contains detailed communication with the customer to understand their requirements and expectations. This stage is known as Requirement Gathering.
- ▶ **System Design:** This phase contains the system design and the complete hardware and communication setup for developing product.
- ▶ **Architectural Design:** System design is broken down further into modules taking up different functionalities. The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood.
- ▶ **Module Design:** In this phase the system breaks down into small modules. The detailed design of modules is specified, also known as Low-Level Design (LLD).

Testing Phases



- ▶ **Unit Testing:** Unit Test Plans are developed during module design phase. These Unit Test Plans are executed to eliminate bugs at code or unit level.
- ▶ **Integration testing:** After completion of unit testing Integration testing is performed. In integration testing, the modules are integrated and the system is tested. Integration testing is performed on the Architecture design phase. This test verifies the communication of modules among themselves.
- ▶ **System Testing:** System testing test the complete application with its functionality, inter dependency, and communication. It tests the functional and non-functional requirements of the developed application.
- ▶ **User Acceptance Testing (UAT):** UAT is performed in a user environment that resembles the production environment. UAT verifies that the delivered system meets user's requirement and system is ready for use in real world.

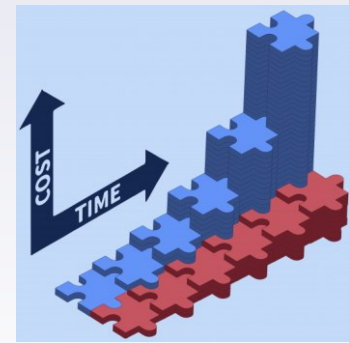
Principles of Software Testing



- ▶ As a whole, software testing principles describe how software testers or testing engineers should create bug-free, clear, and maintainable code. Test principles will help you draft error-catching test cases and create an effective Test Strategy. Software testing is governed by the following seven principles:
- ▶ Testing shows the presence of defects
- ▶ Exhaustive testing is not possible
- ▶ Early testing
- ▶ Defect clustering
- ▶ Pesticide paradox
- ▶ Testing is context dependent
- ▶ Absence of errors fallacy

- ▶ **Testing Shows the Presence of Defects** – The testing principle states that, “Testing talks about the presence of defects and doesn’t talk about the absence of defects”. In software testing, we look for bugs to be fixed before we deploy systems to live environments – this gives us confidence that our systems will work correctly when goes live to users. Despite this, the testing process does not guarantee that software is 100% error-free. It is true that testing greatly reduces the number of defects buried in software, however discovering and repairing these problems does not guarantee a bug-free product or system.
- ▶ **Exhaustive Testing is Impossible** – Exhaustive testing usually tests and verifies all functionality of a software application while using both valid and invalid inputs and pre-conditions. No matter how hard you try, testing EVERYTHING is pretty much impossible. The inputs and outputs alone have an infinite number of combinations, so it is 100% not possible to test an application from every angle.

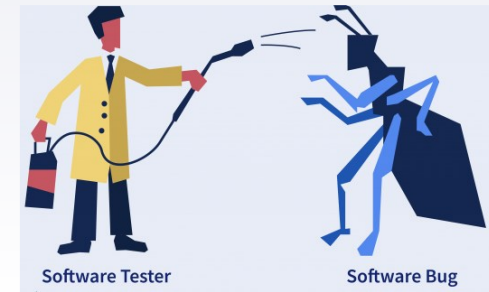
- ▶ **Early Testing** - In software development, early testing means incorporating testing as early as possible in the development process. It plays a critical role in the software development lifecycle (SDLC). The cost to fix a bug increases exponentially with time as the development life cycle progresses.



- ▶ **Defect Clustering** - In software testing, defect clustering refers to a small module or feature that has the most bugs or operation issues. This is because defects are not evenly distributed within a system but are clustered. It could be due to multiple factors, such as the modules might be complicated or the coding related to such modules might be complex.
- ▶ Pareto Principle (80-20 Rule) states that 80% of issues originate from 20% of modules, while the remaining 20% originate from the remaining 80% of modules. Thus, we prioritize testing on 20% of modules where we experience 80% of bugs.



- ▶ **Pesticide Paradox** – generally refers to the practice of repeating the exact same test cases over and over again. As time passes, these test cases will cease to find new bugs. Developers will create tests which are passing so they can forget about negative or edge cases. This is based on the theory that when you repeatedly spray the same pesticide on crops in order to eradicate insects, the insects eventually develop an immunity, making the pesticide ineffective. The same is true for software testing.



- ▶ **Testing is Context-Dependent** – Each type of software system is tested differently. According to this principle, testing depends on the context of the software developed, and this is entirely true. The reality is that every application has its own unique set of requirements, so we can't put testing in a box. Of course, every application goes through a defined testing process, however, the testing approach may vary based on the application type.

► **Absence of Error – Fallacy** - The software which we built not only must be 99% bug-free software but also it must fulfill the business, as well as user requirements otherwise it will become unusable software. Even bug-free software may still be unusable if incorrect requirements are incorporated into the software, or if the software fails to meet the business needs.



Manual vs Automation testing



- ▶ **Manual testing** is the process in which QA analysts execute tests one-by-one in an individual manner. The purpose of manual testing is to catch bugs and feature issues before a software application goes live.
- ▶ When manually testing, the tester validates the key features of a software application. Analysts execute test cases and develop summary error reports without specialized automation tools.
- ▶ **Automation testing** is the process in which testers utilize tools and scripts to automate testing efforts.
- ▶ Automation testing helps testers execute more test cases and improve test coverage. When comparing manual vs. automation testing, manual takes longer. Automated testing is more efficient.

How Manual Testing Works

- ▶ Manual testing is very hands-on. It requires analysts and QA engineers to be highly involved in everything from test case creation to actual test execution.

How Automated Testing Works

- ▶ Automation testing involves testers writing test scripts that automate test execution. (A test script is a set of instructions to be performed on target platforms to validate a feature or expected outcome.)

What Are the Strengths and Weaknesses of Automated Testing vs. Manual Testing?

- ▶ Both have their strengths and weaknesses. Manual testing is slow and tedious. But its strength is that it better handles complex scenarios. Automated testing requires coding and test maintenance. But on the plus side, it is much faster and covers many more permutations.

Manual Testing

Manual testing is not accurate at all times due to human error, hence it is less reliable.

Manual testing is time-consuming, taking up human resources.

Investment is required for human resources.

Manual testing is only practical when the test cases are run once or twice, and frequent repetition is not required.

Manual testing allows for human observation, which may be more useful if the goal is user-friendliness or improved customer experience.

Automated Testing

Automated testing is more reliable, as it is performed by tools and/or scripts.

Automated testing is executed by software tools, so it is significantly faster than a manual approach.

Investment is required for testing tools.

Automated testing is a practical option when the test cases are run repeatedly over a long time period.

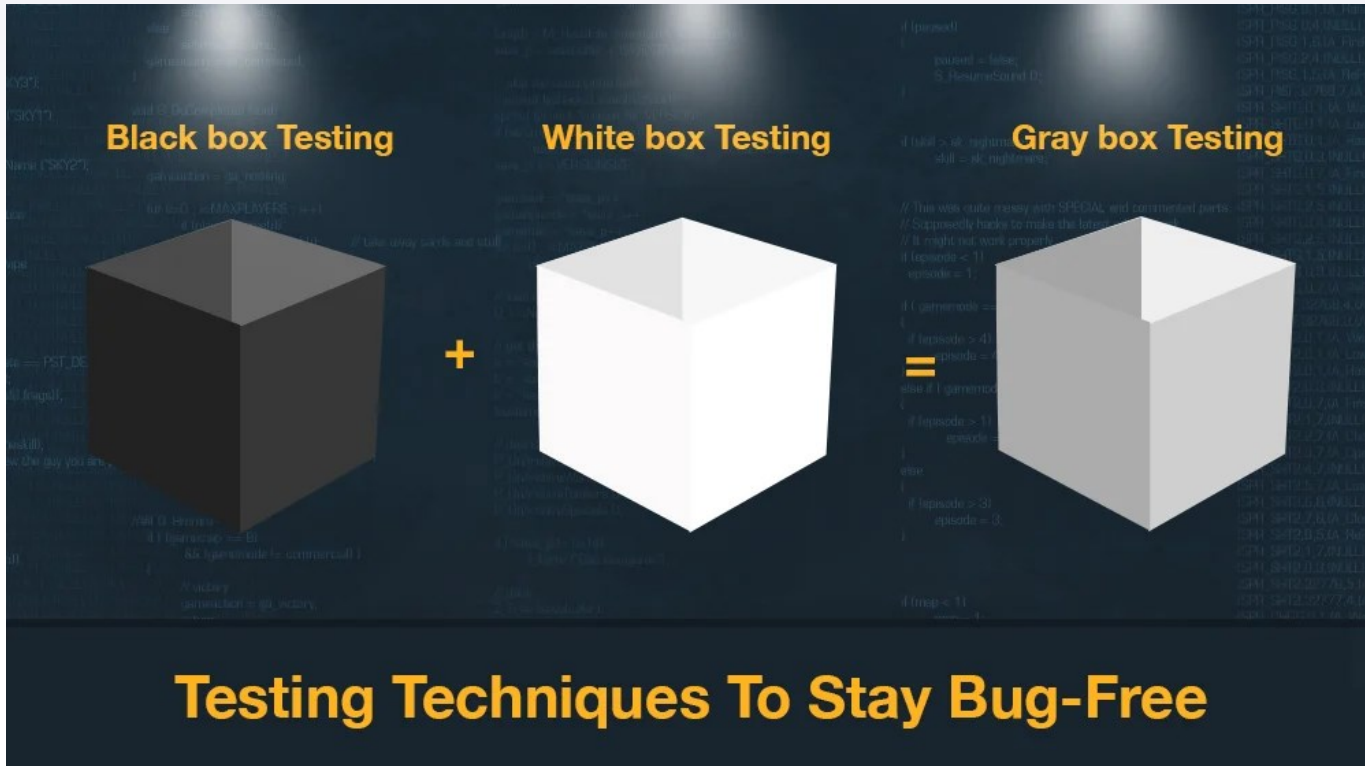
Automated testing does not entail human observation and cannot guarantee user-friendliness or positive customer experience.

Automation Testing Tools



- ▶ Selenium - <http://www.seleniumhq.org/>
- ▶ Appium - <http://appium.io/>
- ▶ Katalon Studio - <https://www.katalon.com/>
- ▶ Cucumber - <https://cucumber.io/>
- ▶ SoapUI - <https://www.soapui.org/>
- ▶ TestComplete - <https://smartbear.com/product/testcomplete/overview/>
- ▶ IBM Rational Functional Tester (RFT)
- ▶

Testing methods



Testing methods



- ▶ **Black Box Testing** is a software testing type that has nothing to do with the internal structure or workings of the codes. The software tester doesn't need to know the system architecture or source code. Black box testing is also known as closed-box testing, data driven testing or functional testing. A functionality of an application can be tested by interacting with its user interface. A tester provides inputs, records outputs and compares it with the standard output he expects. Unexpected results and deviations are noted and brought to the notice of developers and designers.
- ▶ It has limited coverage. Only a select number of scenarios can be created. Its test cases are difficult to design and you cannot depend only on Black box testing to ensure your application is stable and of quality.

Testing methods



- ▶ **White Box Testing** is a software testing type in which the detailed investigation of the internal structure and logic of the application is conducted. White Box testing verifies the correctness of the software's statement, conditions, code paths, loops, and data flows. In White Box testing a coder needs to investigate the source code, to determine which unit of code is not performing as per expectations. The coder chooses appropriate input to exercise the code path he wants to test. The white box testing technique is applicable at unit, integration, and system levels of the software testing process.
- ▶ This technique is **time-consuming** and can only be used by skilled coders. It is therefore **expensive**. The programmer must thoroughly know the application he wants to test so that he is then able to create the right kind of test cases. The White Box testing technique is also called **Glass box testing**, design-based testing, clear box testing, or structural testing by some testers.

Testing methods



- ▶ **Gray Box Testing** uses effective combination of both Black Box testing techniques and White Box testing techniques. Grey Box testing techniques increase testing coverage by focusing on all the layers of a complex system by combining all of the Black and White testing techniques. In order to design tests under Gray Box, the tester needs to know all about the algorithm and internal data structure of the software.
- ▶ The techniques can be used by testers, developers and end users. Testers don't need to know the programming language or methods of testing applications which makes the Gray Box Testing Techniques unbiased. The software is tested from the perspective of user rather than the designer. Web applications can be best tested by Gray Box Testing techniques as they don't have any source code or binaries.

Functional and non-functional Testing



Software Testing is broadly categorized into Functional and Non- Functional Testing.

- ▶ **Functional testing** is testing the ‘Functionality’ of a software or an application under test.
- ▶ It tests the behavior of the software under test. Based on the requirement of the client, a document called a software specification or Requirement Specification is used as a guide to test the application.
- ▶ Test data is sculpted based on it and a set of Test Cases are prepared. The software is then tested in a real environment to check whether the actual result is in sync with the expected result.
- ▶ Eg: Unit Testing,, Integration Testing, System Testing, User Acceptance Testing, Recovery testing etc

Functional and non-functional Testing



- ▶ There are some aspects which are complex such as the performance of an application etc and this testing checks the Quality of the software to be tested. Quality majorly depends on time, **accuracy, stability, correctness and durability** of a product under various adverse circumstances.
- ▶ In software terms, when an application works as per the user's expectation, smoothly and efficiently under any condition, then it is stated as a reliable application. Based on these aspects of quality, it is very critical to test under these parameters. This type of testing is called **Non- Functional Testing**.
- ▶ It is not feasible to test this type manually, hence some special automated tools are used to test it.
- ▶ Eg: Performance Testing, Usability Testing, Security Testing etc.



Non-functional Testing Parameters

Security	Availability	Efficiency	Integrity
Reliability	Survivability	Usability	Flexibility
Scalability	Reusability	Interoperability	Portability



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

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

3) Survivability:

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

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

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

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

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

8) Efficiency: The extent to which any software system can handle capacity, quantity and response time.

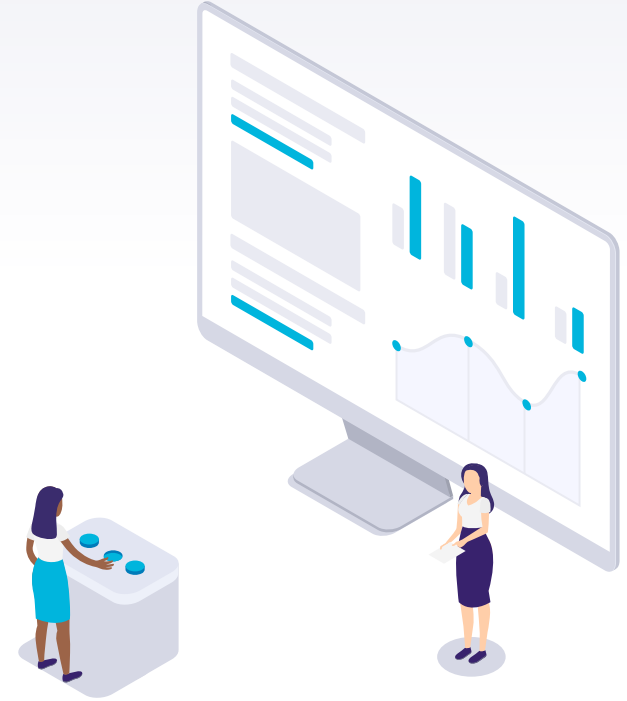
9) Flexibility: The term refers to the ease with which the application can work in different hardware and software configurations. Like minimum RAM, CPU requirements.

10) Portability: The flexibility of software to transfer from its current hardware or software environment.

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

1

Introduction to Selenium



Manual Testing - Limitations



Automated Testing



Speed of execution



Accurate results



Lesser investment in human
resources



Time and cost effective



Early time to market



Supports re-testing

Birth of Selenium



Jason Huggins, an engineer at **ThoughtWorks**, Chicago found the repetitious work of manual testing strenuous and monotonous

He developed a JavaScript program to automate the testing of a web application

The program was called **JavaScriptTestRunner**

Initially, the new invention was deployed by the inmates at Thoughtworks. However, in 2004 it was renamed as **Selenium** and was made **open source**

What is Selenium?

Selenium is an automated testing tool used to test web applications across various browsers



Open source



Provides a record/playback tool for authoring tests without learning a test scripting language



Consists of a set of software tools that facilitate testing



Can be coded in many programming languages



Primarily developed in JavaScript



Browser and platform independent





Founded by Simon Stewart in 2006

It is a cross platform testing framework

Programming interface to create and run test cases

Makes provision to perform action on web elements

Does not require a core engine like RC and interacts natively with the browser applications

Supports Java, C#, PHP, Python, Perl, Ruby etc.

JUnit

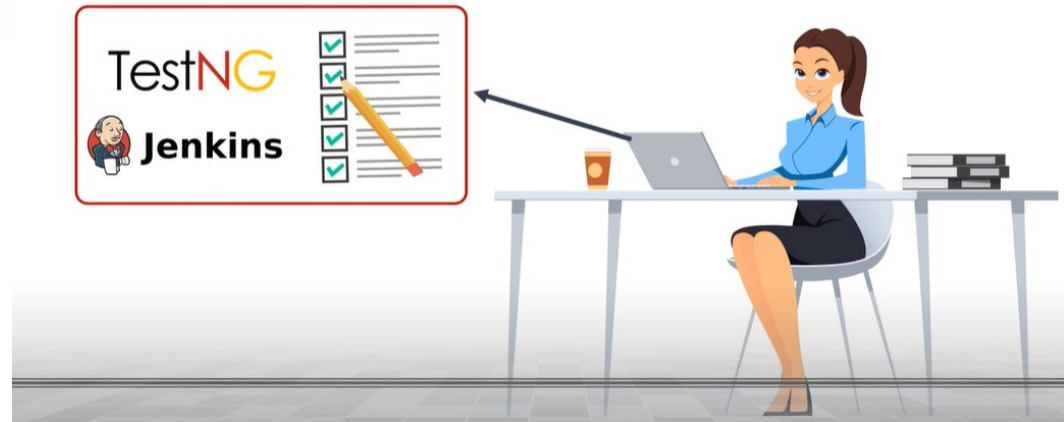
Supports frameworks like TestNG, JUnit, NUnit

designed by freepik.com



- ▶ Selenium provides a set of tools and libraries that enable and support web browser automation testing

She uses TestNG or Jenkins to generate proper test report.



What is Selenium WebDriver?



- ▶ <https://www.seleniumhq.org>



Redundant & confusing APIs

A single function has several commands

?

Different browsers interpret different commands differently

Selenium RC

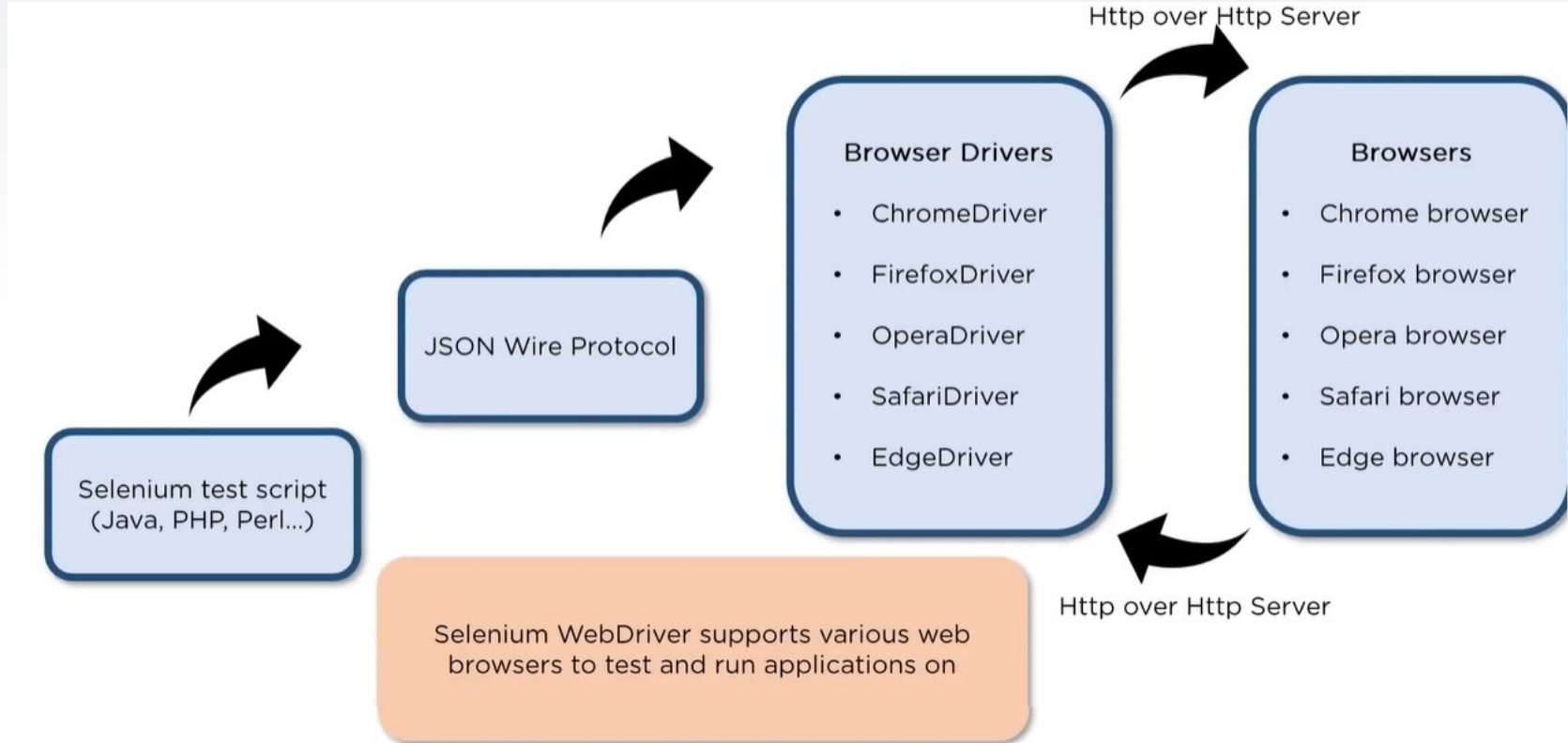
sendKeys()
Click()
scrollBy()

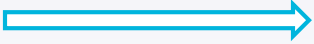
Simple and easy to remember

No Redundant commands

Selenium WebDriver

Architecture



- ▶ <https://www.selenium.dev/downloads/>
- ▶ Download 
- ▶ To automate different web browsers with applications, go to Selenium Web Driver, and download whichever, you wish to test with



Java

Stable: [4.3.0 \(June 23, 2022\)](#)

[Changelog](#)

[API Docs](#)

3 step installation



Java



Eclipse



Selenium

Selenium IDE

Selenium IDE is a Chrome, Firefox and Edge plugin which records and plays back user interactions with the browser. Use this to either create simple scripts or assist in exploratory testing.

Download latest released version for [Chrome](#) or [Firefox](#) or [Edge](#). View the [Release Notes](#).

Download previous IDE [versions](#).

Steps to follow



- ▶ Create a Java Project in Eclipse
- ▶ Convert the project to a Maven Project
- ▶ From <https://mvnrepository.com/> search for selenium java. Copy the dependency
- ▶ From the above site, copy TestNG dependency.
- ▶ In POM.XML, add the dependencies.
- ▶ Create a java class with main function

Browser APIs and their usage

Get Commands

```
driver.get("https://www.google.co.in");
driver.getTitle();
driver.getCurrentUrl();
driver.getPageSource();
driver.getWindowHandle();
driver.getWindowHandles();
```

Navigation Commands

```
driver.navigate().to("https://www.ebay.in/");
driver.navigate().back();
driver.navigate().forward();
driver.navigate().refresh();
```

Locator Commands

```
driver.findElement(By.id("user"));
driver.findElement(By.linkText("click here"));
driver.findElement(By.className("main-container"));
driver.findElement(By.name("admin"));
driver.findElement(By.cssSelector("#primary"));
driver.findElement(By.xpath("//*[@id='primary']"));
```

Browser Commands

```
driver.close()
    Closes the current browser window

driver.quit()
    Quits the entire browser session
```


Demo Usecase



Usecase 1: As a user, I want to visit the eBay site, and search for "JBL Speakers".
Then, I want to check the "Daily deals".

Usecase 2: From eBay, navigate to Amazon site and search for "JBL Speakers". Then
navigate back to eBay.

Usecase 3: At eBay, I want to print the page Title and close the browser

Doubts?

