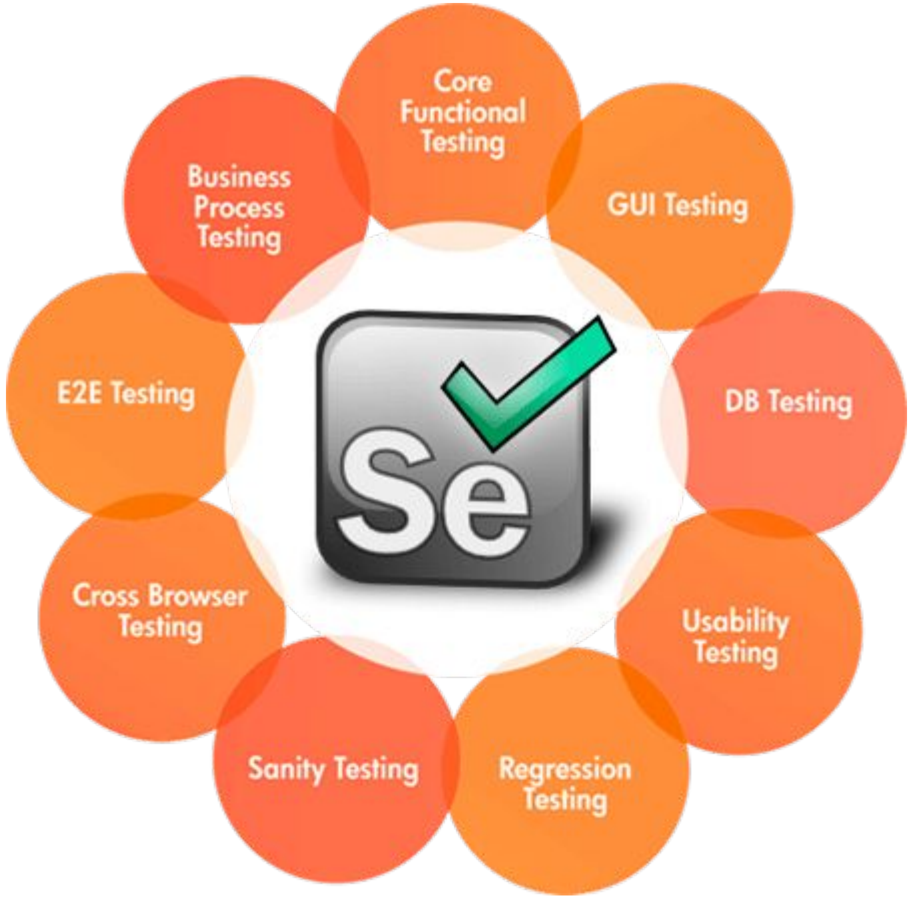
SDJ INTERNATIONAL COLLEGE

Introduction
Unit 1

– By Tushar Sir



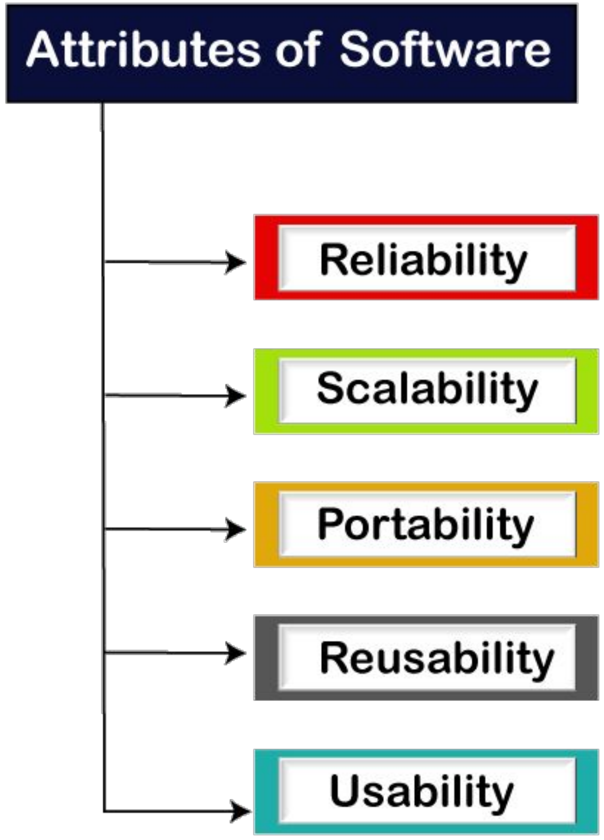
– By Tushar Sir

Concept of Software testing

- ***Software testing*** is the process of evaluating and verifying that a software product or application does what it is supposed to do.
- The benefits of testing include preventing bugs, reducing development costs and improving performance.
- Software testing is a process of identifying the correctness of software by considering its all attributes **(Reliability, Scalability, Portability, Re-usability, Usability)** and evaluating the execution of software components to find the software bugs or errors or defects.
- Software testing can be stated as the process of verifying and validating whether a software or application is bug-free, meets the technical requirements as guided by its design and development, and meets the user requirements effectively and efficiently by handling all the exceptional and boundary cases.

– By Tushar Sir

Concept of Software testing



– By Tushar Sir

Concept of Software testing

- Software testing provides an independent view and objective of the software and gives surety of fitness of the software.
- It involves testing of all components under the required services to confirm that whether it is satisfying the specified requirements or not.
- The process is also providing the client with information about the quality of the software.
- Testing is mandatory because it will be a dangerous situation if the software fails any of time due to lack of testing. So, without testing software cannot be deployed to the end user.

– By Tushar Sir

Concept of Software testing

Software testing can be divided into two steps:

1. **Verification:** it refers to the set of tasks that ensure that the software correctly implements a specific function.
2. **Validation:** it refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements.

- ***Verification:*** "Are we building the product right?"
- ***Validation:*** "Are we building the right product?"

– By Tushar Sir

What is testing?

- ***Testing*** is a group of techniques to determine the correctness of the application under the predefined script but, testing cannot find all the defect of application.
- The main intent of testing is to detect failures of the application so that failures can be discovered and corrected.
- It does not demonstrate that a product functions properly under all conditions but only that it is not working in some specific conditions.
- Testing furnishes comparison that compares the behavior and state of software against mechanisms because the problem can be recognized by the mechanism.
- The mechanism may include past versions of the same specified product, comparable products, and interfaces of expected purpose, relevant standards, or other criteria but not limited up to these.

– By Tushar Sir

What is testing?

- Testing includes an examination of code and also the execution of code in various environments, conditions as well as all the examining aspects of the code.
- In the current scenario of software development, a testing team may be separate from the development team so that Information derived from testing can be used to correct the process of software development.
- The success of software depends upon acceptance of its targeted audience, easy graphical user interface, strong functionality load test, etc.
- **For example,** the audience of banking is totally different from the audience of a video game. Therefore, when an organization develops a software product, it can assess whether the software product will be beneficial to its purchasers and other audience.

– By Tushar Sir

History of Software testing

- Software testing arrived alongside the development of software, which had its beginnings just after the second world war.
- Computer scientist Tom Kilburn is credited with writing the first piece of software, which debuted on June 21, 1948, at the University of Manchester in England. It performed mathematical calculations using machine code instructions.
- Debugging was the main testing method at the time and remained so for the next two decades.
- By the 1980s, development teams looked beyond isolating and fixing software bugs to testing applications in real-world settings.
- It set the stage for a broader view of testing, which encompassed a quality assurance process that was part of the software development life cycle.

– By Tushar Sir

History of Software testing

- **1940s-1950s:** Early computers were manually tested by programmers for errors due to limited technology.
- **1960s:** Formal testing methodologies like "waterfall" emerged, focusing on verification and validation.
- **1970s:** Debugging tools and techniques gained popularity to identify and fix coding errors.
- **1980s:** White-box and black-box testing methods gained traction, promoting structured testing approaches.
- **1990s:** Agile methodologies emerged, emphasizing iterative development and continuous testing.

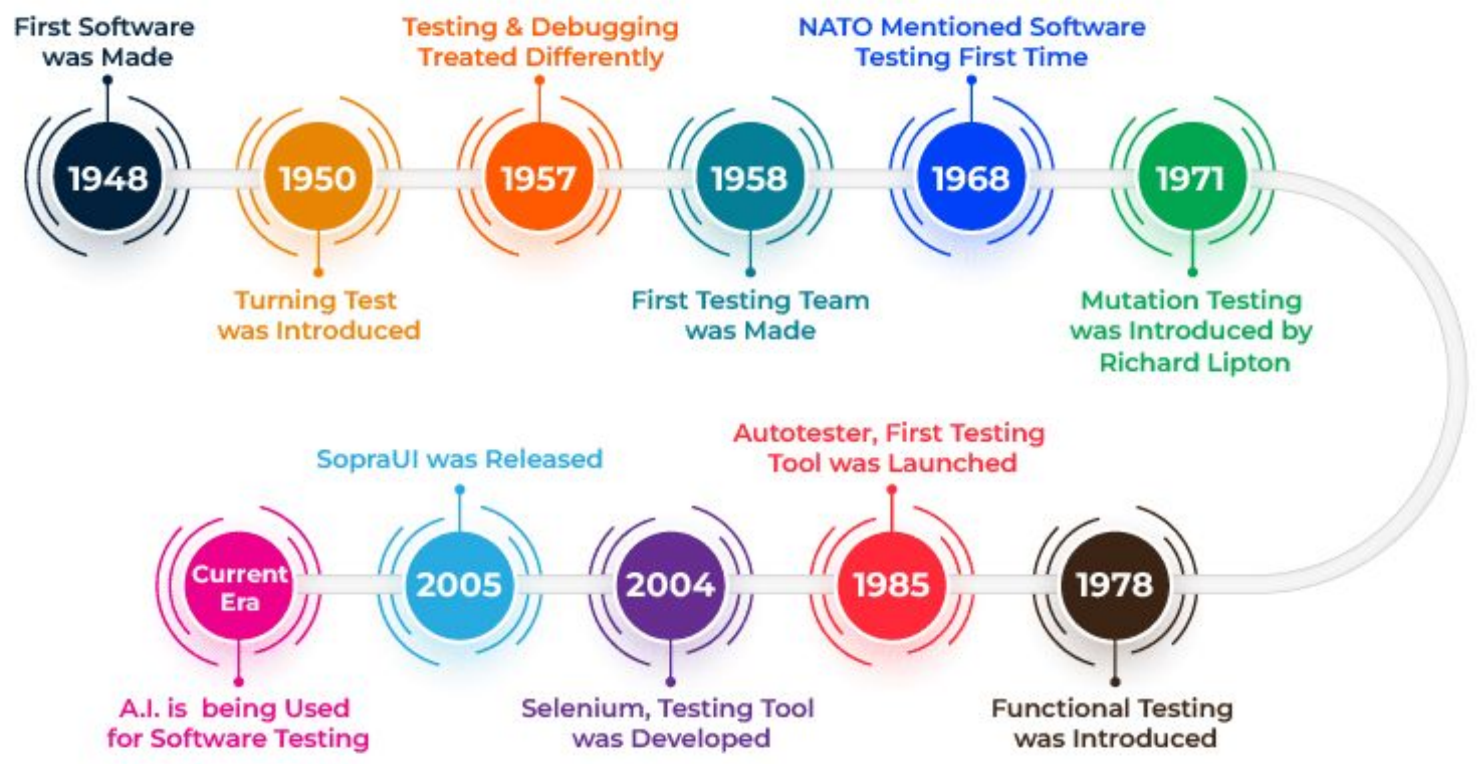
– By Tushar Sir

History of Software testing

- **2000s:** Web application testing and compatibility testing became significant with the growth of the internet.
- **2010s:** Test automation tools expanded, integrating testing earlier in the development process.
- **2010-2020s:** Agile practices and DevOps culture accelerated continuous testing and integration. Performance, security, and user experience testing gained prominence.
- **Present:** AI-driven testing and shift-right testing approaches evolved, enhancing testing efficiency.

– By Tushar Sir

History of Software testing


First Software was Made
1948
1950
Turning Test was Introduced
Testing & Debugging Treated Differently
1957
1958
First Testing Team was Made
NATO Mentioned Software Testing First Time
1968
1971
Mutation Testing was Introduced by Richard Lipton
1978
Functional Testing was Introduced
Autotester, First Testing Tool was Launched
1985
2004
Selenium, Testing Tool was Developed
SopraUI was Released
2005
Current Era
A.I. is being Used for Software Testing

– By Tushar Sir

Why Software testing is important?

Software testing is crucial for several reasons:

- **Quality Assurance:** Testing helps identify defects and errors in software, ensuring that the final product meets quality standards and performs as expected.
- **Bug Detection:** Testing uncovers bugs and glitches in the software, preventing them from reaching end-users and causing operational issues.
- **User Satisfaction:** Thorough testing enhances user experience by delivering a reliable and functional software product that meets user expectations.
- **Risk Mitigation:** Testing reduces the risk of software failures and potential financial losses by addressing issues before deployment.
- **Security:** Testing identifies vulnerabilities and security loopholes, safeguarding sensitive data and ensuring the software is resistant to cyber threats.
- **Compliance:** Many industries require software to meet specific regulatory standards. Testing ensures that the software aligns with these regulations.

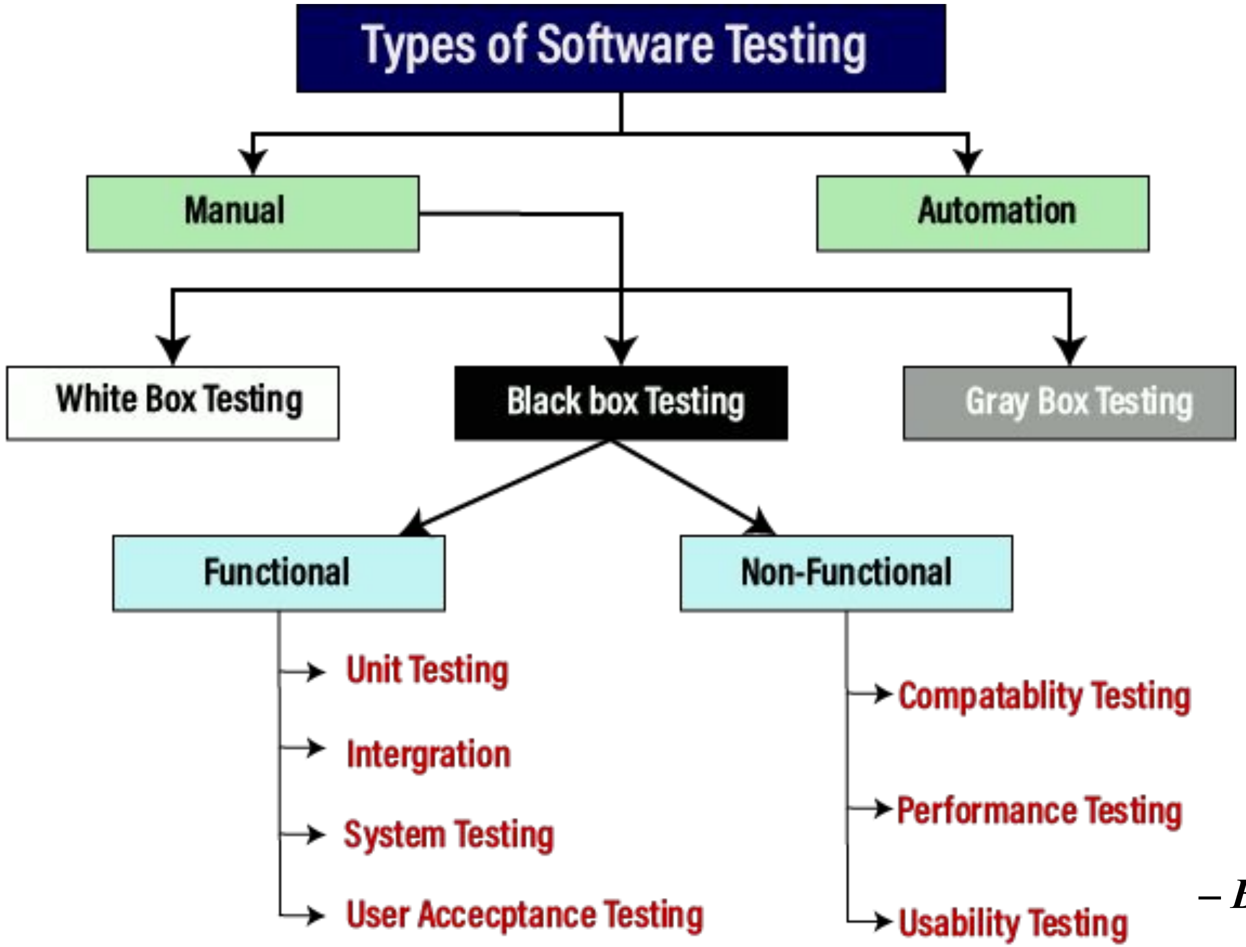
– By Tushar Sir

Why Software testing is important?

- **Cost Savings:** Detecting and fixing issues early in the development cycle is more cost-effective than addressing them after deployment.
- **Reputation:** High-quality software builds a positive reputation for the developer or organization, fostering trust among users.
- **Continuous Improvement:** Feedback from testing helps developers refine and enhance the software, leading to iterative improvements.
- **Confidence:** Rigorous testing provides confidence to stakeholders, including developers, managers, and users, that the software is reliable and functional.
- **Compatibility:** Testing ensures the software works correctly on different platforms, devices, and environments.
- **Performance:** Performance testing assesses the software's speed, responsiveness, and scalability under various conditions.

– By Tushar Sir

Types of Software testing



– By Tushar Sir

Manual testing

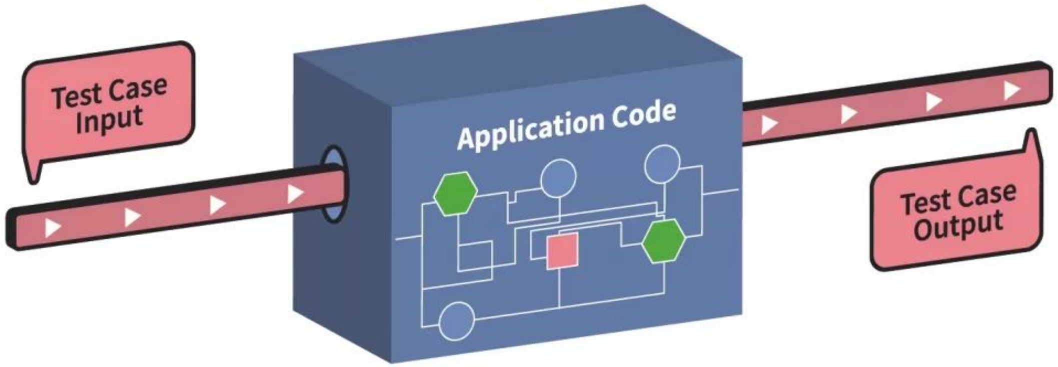
- The process of checking the functionality of an application as per the customer needs without taking any help of automation tools is known as manual testing.
- While performing the manual testing on any application, we do not need any specific knowledge of any testing tool, rather than have a proper understanding of the product so we can easily prepare the test document.
- In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug. There are different stages for manual testing such as unit testing, integration testing, system testing, and user acceptance testing.
- Testers use test plans, test cases, or test scenarios to test software to ensure the completeness of testing. Manual testing also includes exploratory testing, as testers explore the software to identify errors in it.
- Manual testing can be further divided into three types of testing, which are as follows:
  - **White box testing**
  - **Black box testing**
  - **Gray box testing**

– By Tushar Sir

Manual testing

White-box Testing:

- Also known as **"clear box"** or **"structural testing,"** white-box testing involves examining the internal code and logic of the software.
- Testers have access to the source code and design specifications. This type of testing focuses on verifying code paths, logic flow, and making sure that all possible scenarios and conditions are tested.
- It's particularly effective for finding coding errors, security vulnerabilities, and ensuring code coverage.

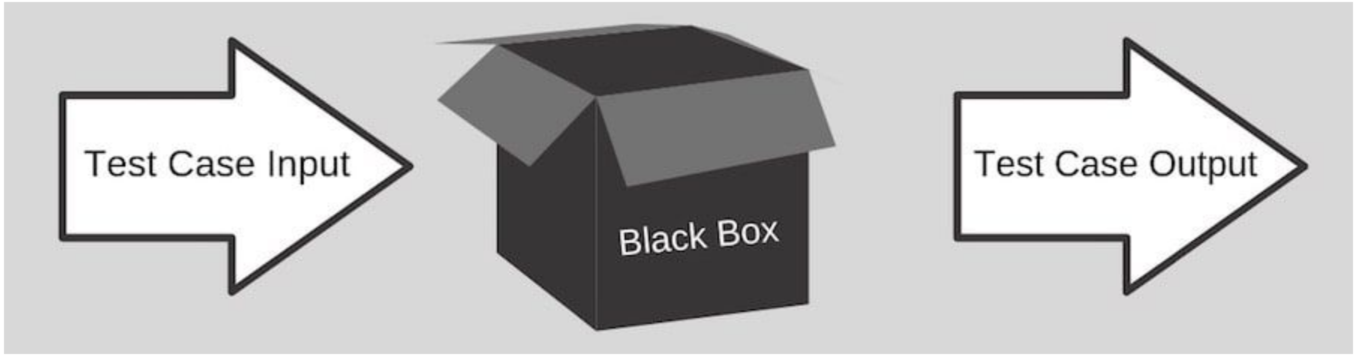


– By Tushar Sir

Manual testing

Black-box Testing:

- Black-box testing treats the software as a "black box" where testers don't have access to the internal code or design details.
- Instead, testing is based on the software's specified requirements and functionality. Testers create test cases to validate inputs and outputs, uncover defects, and ensure that the software meets its intended functionality.
- It's helpful for assessing the software from a user's perspective and identifying external issues that users might encounter.



– By Tushar Sir

Manual testing

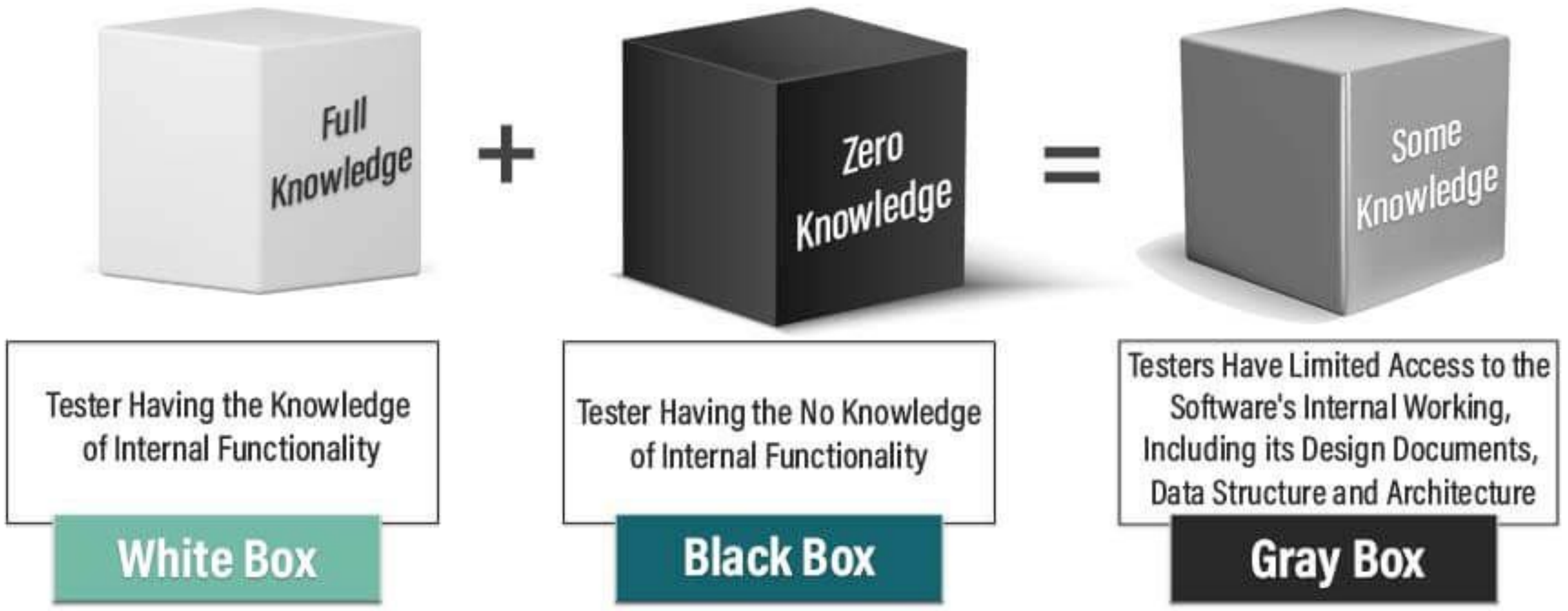
Gray-box Testing:

- Gray-box testing combines elements of both white-box and black-box testing.
- Testers have partial knowledge of the internal workings of the software, such as high-level design or system architecture.
- This approach aims to leverage both the external perspective of black-box testing and the focused analysis of white-box testing.
- Gray-box testing is particularly useful for scenarios where having some knowledge of the internal structure can enhance the effectiveness of the testing process.

Black Box Testing
+
White Box Testing
=
Gray Box Testing

– By Tushar Sir

Manual testing



Tester Having the Knowledge of Internal Functionality

White Box

Tester Having the No Knowledge of Internal Functionality

Black Box

Testers Have Limited Access to the Software's Internal Working, Including its Design Documents, Data Structure and Architecture

Gray Box

– By Tushar Sir

Manual testing

Pros:

- **Exploratory Testing:** Manual testing is well-suited for exploratory testing, where testers can use their creativity and domain knowledge to uncover unexpected issues.
- **Early Testing:** It can be initiated early in the development process, even before a graphical user interface (GUI) is available, making it useful for initial testing.
- **User Experience Evaluation:** Manual testing allows testers to evaluate the user interface, user experience, and usability aspects of the software.
- **Adaptability:** Testers can adapt quickly to changes in requirements or the application's behavior, making it flexible in agile development environments.
- **Complex Test Cases:** For highly complex scenarios and non-functional testing like usability, accessibility, and localization, manual testing is often more effective.

– By Tushar Sir

Manual testing

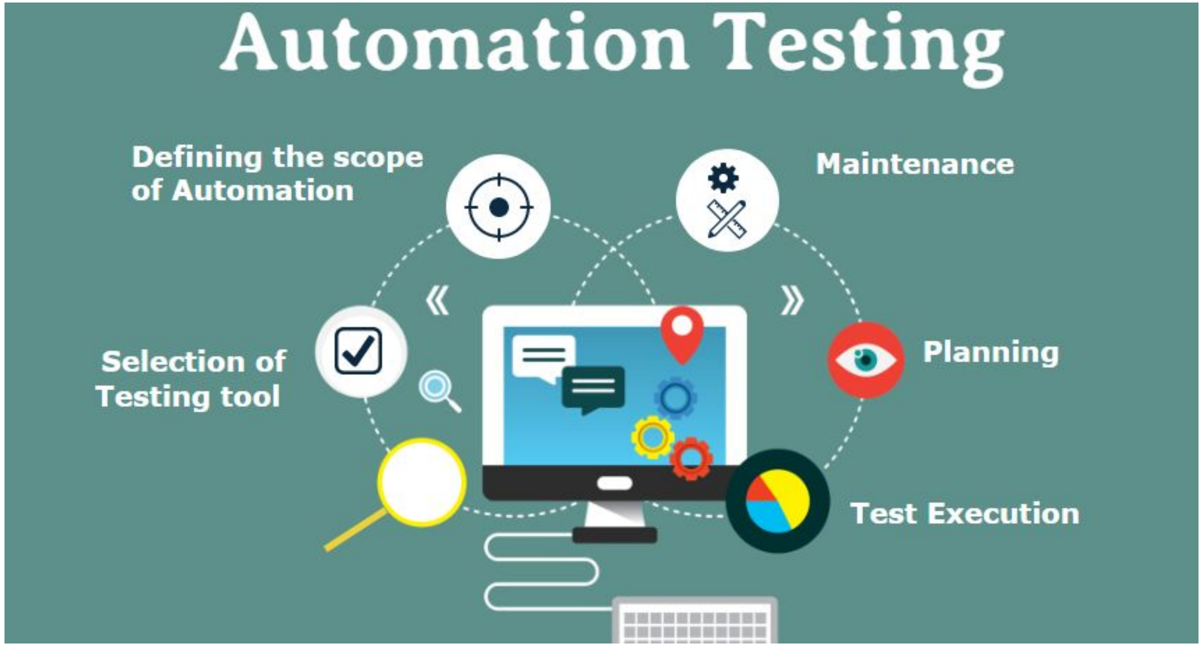
Cons:

- **Resource-Intensive:** Manual testing requires a team of skilled testers, which can be expensive and time-consuming for large or complex applications.
- **Human Error:** Testers may introduce errors during test case execution, and there is a risk of missing defects due to human oversight.
- **Repetitive Tasks:** Regression testing and repetitive test cases can be monotonous and prone to errors.
- **Limited Test Coverage:** It may not be feasible to perform exhaustive testing due to time and resource constraints, potentially missing some defects.
- **Documentation Overhead:** Manual test cases require detailed documentation, which can be time-consuming and may not be up to date.

– By Tushar Sir

Automation testing

- **Automated Testing** is a technique where the Tester writes scripts on their own and uses suitable Software or Automation Tool to test the software.



– By Tushar Sir

Automation testing

- It is an Automation Process of a Manual Process. It allows for executing repetitive tasks without the intervention of a Manual Tester.
- It is used to automate the testing tasks that are difficult to perform manually.
- Automation tests can be run at any time of the day as they use scripted sequences to examine the software.
- Automation tests can also enter test data can compare the expected result with the actual result and generate detailed test reports.
- The goal of automation tests is to reduce the number of test cases to be executed manually but not to eliminate manual testing.
- It is possible to record the test suit and replay it when required.

– By Tushar Sir

Why Transform From Manual to Automated Testing?

- In the year 1994, An aircraft completing its Routine flight crashed just before landing. This was due to some bug or defect in the Software.
- The Testers didn’t even care about the final testing and hence this accident happened. So in order to replace for few of the Manual Tests (mandatory), there is a need for Automation Testing.

Below are some of the reasons for using automation testing:

- **Quality Assurance:** Manual testing is a tedious task that can be boring and at the same time error-prone. Thus, using automation testing improves the quality of the software under test as more test coverage can be achieved.

– By Tushar Sir

Why Transform From Manual to Automated Testing?

- **Error or Bug-free Software:** Automation testing is more efficient for detecting bugs in comparison to manual testing.
- **No Human Intervention:** Manual testing requires huge manpower in comparison to automation testing which requires no human intervention and the test cases can be executed unattended.
- **Increased test coverage:** Automation testing ensures more test coverage in comparison to manual testing where it is not possible to achieve 100% test coverage.
- **Testing can be done frequently:** Automation testing means that the testing can be done frequently thus improving the overall quality of the software under test.

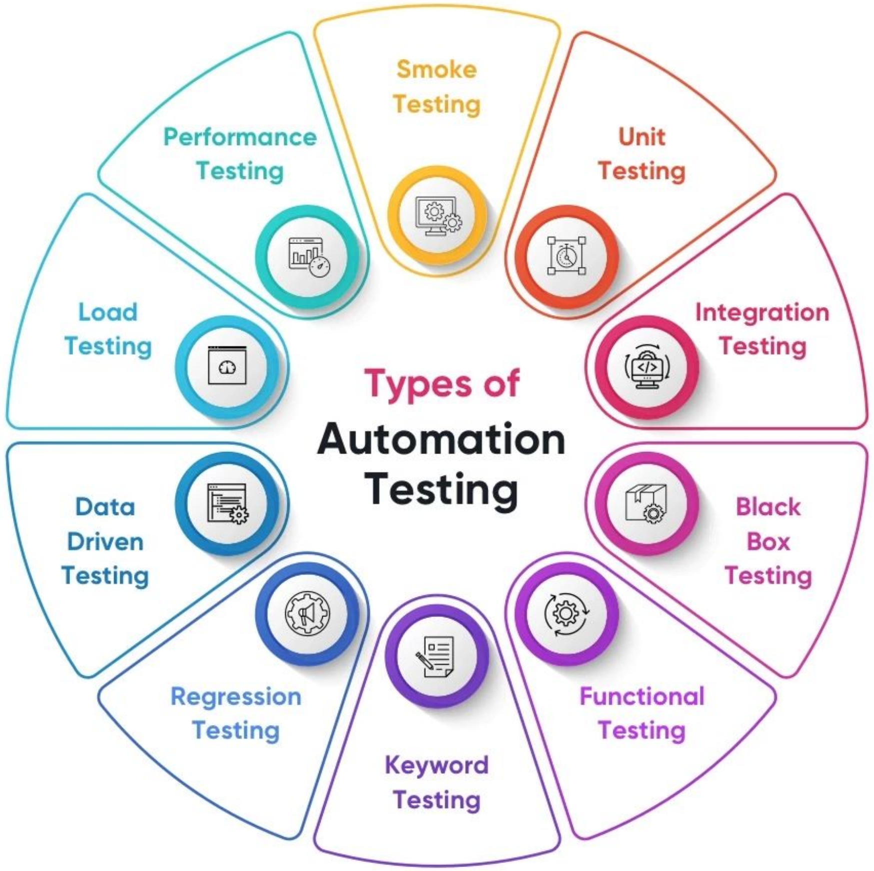
– By Tushar Sir

Types of Automation Testing

- **Unit testing:** Unit testing is a phase in software testing to test the smallest piece of code known as a unit that can be logically isolated from the code. It is carried out during the development of the application.
- **Integration testing:** Integration testing is a phase in software testing in which individual software components are combined together and tested as a group. It is carried out to check the compatibility of the component with the specified functional requirements.
- **Smoke testing:** Smoke testing is a type of software testing that determines whether the built software is stable or not. It is the preliminary check of the software before its release in the market.
- **Performance testing:** Performance testing is a type of software testing that is carried out to determine how the system performs in terms of stability and responsiveness under a particular load.

– By Tushar Sir

Types of Automation Testing



– By Tushar Sir

Types of Automation Testing

- **Regression testing:** Regression testing is a type of software testing that confirms that previously developed software still works fine after the change and that the change has not adversely affected existing features.
- **Security testing:** Security testing is a type of software testing that uncovers the risks, and vulnerabilities in the security mechanism of the software application. It helps an organization to identify the loopholes in the security mechanism and take corrective measures to rectify the gaps in security.
- **Acceptance testing:** Acceptance testing is the last phase of software testing that is performed after the system testing. It helps to determine to what degree the application meets end users' approval.
- **API testing:** API testing is a type of software testing that validates the Application Programming Interface(API) and checks the functionality, security, and reliability of the programming interface.
- **UI Testing:** UI testing is a type of software testing that helps testers ensure that all the fields, buttons, and other items on the screen function as desired.

– By Tushar Sir

Test Automation Frameworks

- **Linear framework:** This is the most basic form of framework and is also known as the record and playback framework. In this testers create and execute the test scripts for each test case. It is mostly suitable for small teams that don't have a lot of test automation experience.
- **Modular-Based Framework:** This framework organizes each test case into small individual units knowns as modules each module is independent of the other, having different scenarios but all modules are handled by a single master script. This approach requires a lot of pre-planning and is best suited for testers who have experience with test automation.
- **Library Architecture Framework:** This framework is the expansion of a modular-based framework with few differences. Here, the task is grouped within the test script into functions according to a common objective. These functions are stored in the library so that they can be accessed quickly when needed. This framework allows for greater flexibility and reusability but creating scripts takes a lot of time so testers with experience in automation testing can benefit from this framework.

– By Tushar Sir

Which Tests to Automate?

- **Monotonous test:** Repeatable and monotonous tests can be automated for further use in the future.
- **A test requiring multiple data sets:** Extensive tests that require multiple data sets can be automated.
- **Business critical tests:** High-risk business critical test cases can be automated and can be scheduled to run regularly.
- **Determinant test:** Determinant test cases where it is easy for the computer to decide whether the test is failed or not can be automated.
- **Tedious test:** Test cases that involve repeatedly doing the same action can be automated so that the computer can do the repetitive task as humans are very poor at performing the repetitive task with efficiency, there increase the chances of error.

– By Tushar Sir

Automation Testing Process

- **Test Tool Selection:** There will be some criteria for the Selection of the tool. The majority of the criteria include: Do we have skilled resources to allocate for automation tasks, Budget constraints, and Do the tool satisfies our needs?
- **Define Scope of Automation:** This includes a few basic points such as the Framework should support Automation Scripts, Less Maintenance must be there, High Return on Investment, Not many complex Test Cases
- **Planning, Design, and Development:** For this, we need to Install particular frameworks or libraries, and start designing and developing the test cases such as NUnit, JUnit, QUnit, or required Software Automation Tools
- **Test Execution:** Final Execution of test cases will take place in this phase and it depends on Language to Language for .NET, we'll be using NUnit, for Java, we'll be using JUnit, for JavaScript, we'll be using QUnit or Jasmine, etc.
- **Maintenance:** Creation of Reports generated after Tests and that should be documented so as to refer to that in the future for the next iterations.

– By Tushar Sir

Best Practices for Test Automation

- **Plan self-contained test cases:** It is important to ensure that the test is clearly defined and well-written. The test cases should be self-contained and easy to understand.
- **Plan the order to execute tests:** Planning the test in the manner that the one test creates the state for the second test can be beneficial as it can help to run test cases in order one after another.
- **Use tools with automatic scheduling:** If possible use tools that can schedule testing automatically according to a schedule.
- **Set up an alarm for test failure:** If possible select a tool that can raise an alarm when a test failure occurs. Then a decision needs to be made whether to continue with the test or abort it.
- **Reassess test plans as the app develops and changes:** It is important to continuously reassess the test plan as there is no point in wasting resources in testing the legacy features in the application under test.

– By Tushar Sir

Popular Automation Tools

- **Selenium:** Selenium is an automated testing tool that is used for Regression testing and provides a playback and recording facility. It can be used with frameworks like JUnit and Test NG. It provides a single interface and lets users write test cases in languages like Ruby, Java, Python, etc.
- **QTP:** Quick Test Professional (QTP) is an automated functional testing tool to test both web and desktop applications. It is based on the VB scripting language and it provides functional and regression test automation for software applications.
- **Sikuli:** It is a GUI-based test automation tool that is used for interacting with elements of web pages. It is used to search and automate graphical user interfaces using screenshots.
- **Appium:** Apium is an open-source test automation framework that allows QAs to conduct automated app testing on different platforms like iOS, Android, and Windows SDK.
- **Jmeter:** Apache JMeter is an open-source Java application that is used to load test the functional behavior of the application and measure the performance.

– By Tushar Sir

Automation Testing

Pros:

- **Speed and Efficiency:** Automation tests execute much faster than manual tests, allowing for quicker feedback during development.
- **Repeatability:** Automated tests can be run repeatedly with consistent results, making them ideal for regression testing.
- **Wider Test Coverage:** Automation can cover a large number of test cases and configurations, improving overall test coverage.
- **Non-Functional Testing:** It is well-suited for non-functional testing types like performance, load, and stress testing, where precise measurements are required.
- **Continuous Integration:** Automated tests can be integrated into the CI/CD pipeline, ensuring that tests run automatically after code changes.

– By Tushar Sir

Automation Testing

Cons:

- **Initial Setup:** Automation testing requires significant upfront effort for test script development, tool setup, and maintenance.
- **Limited Exploratory Testing:** It is less effective for exploratory testing, as it relies on predefined test scripts.
- **Maintenance Overhead:** Test scripts require ongoing maintenance to keep up with application changes, making automation more challenging in agile environments.
- **Skill Requirement:** Automated testing demands testers with scripting and programming skills, which may not be readily available.
- **Cost:** The cost of automation tools and resources can be high, particularly for smaller organizations.
- **False Positives/Negatives:** Automated tests may produce false positives (reporting issues that don't exist) or false negatives (missing real issues) due to script limitations or environmental factors.

– By Tushar Sir

Introduction to Selenium

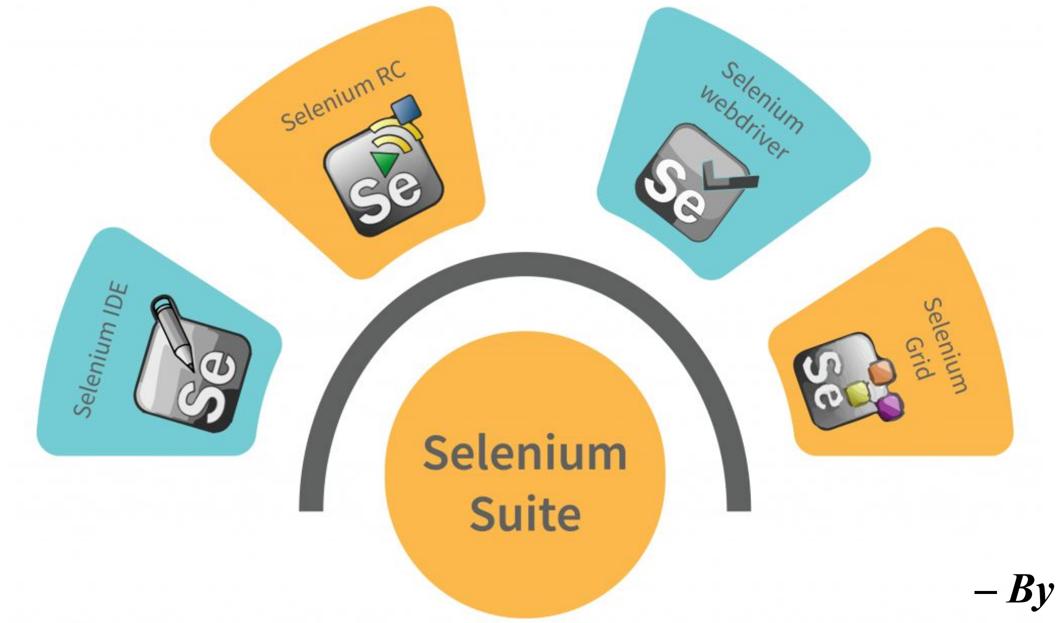
- **Selenium** is a free, open-source tool for automating web applications. It's used to test web applications across different browsers and platforms.
- Selenium is similar to HP QuickTest Pro (QTP).
- Selenium was originally developed by Jason Huggins in 2004 as an internal tool at Thought Works.
- It's primarily built in Java and supports several browsers and programming languages.
- You can use multiple programming languages like Java, C#, and Python to create Selenium Test Scripts.

– By Tushar Sir

Introduction to Selenium

The Selenium test suite comprises of four tools:

- **Selenium Integrated Development Environment (IDE)**
- **Selenium Remote Control (RC)**
- **Selenium WebDriver**
- **Selenium Grid**



– By Tushar Sir

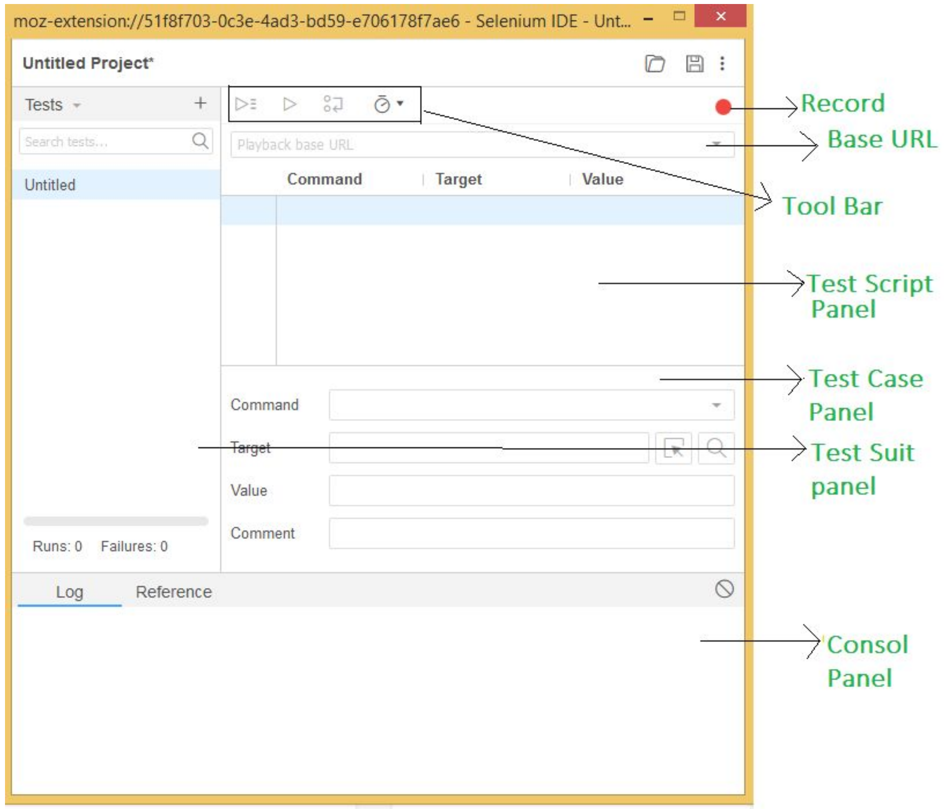
Introduction to Selenium

- **Selenium IDE:** Selenium Integrated Development Environment (IDE) is a Firefox plugin that lets testers to record their actions as they follow the workflow that they need to test.
- **Selenium RC:** Selenium Remote Control (RC) was the flagship testing framework that allowed more than simple browser actions and linear execution. It makes use of the full power of programming languages such as Java, C#, PHP, Python, Ruby and PERL to create more complex tests.
- **Selenium WebDriver:** Selenium WebDriver is the successor to Selenium RC which sends commands directly to the browser and retrieves results.
- **Selenium Grid:** Selenium Grid is a tool used to run parallel tests across different machines and different browsers simultaneously which results in minimized execution time.

– By Tushar Sir

Selenium IDE

- At the beginning Selenium IDE(Integrated Development Environment) was implemented as a Firefox add-on/plugin and now it can be used Selenium IDE on every web browser. It provides record and playback functionality. The figure shows Selenium IDE.



– By Tushar Sir

Selenium IDE

Pros:

- It is an open-source tool.
- Provide base, for extensions.
- It provides multi-browser support.
- No programming language experience is required while using Selenium IDE.
- The user can set breakpoints and debug.
- It provides record and playback functions.
- Easy to Use and install.
- Languages supported by Selenium IDE are Java, Python, C# etc.

– By Tushar Sir

Selenium IDE

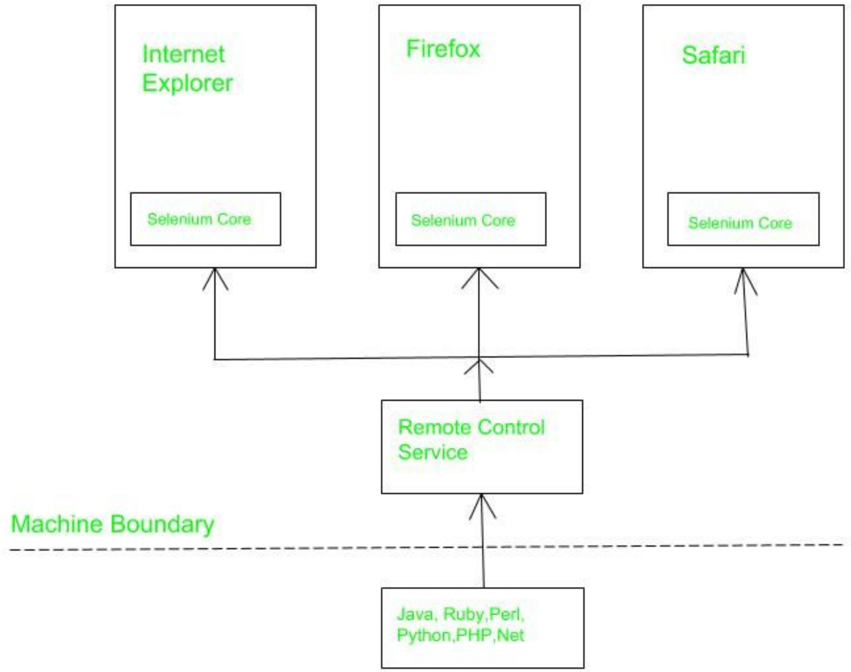
Cons:

- There are no support iteration and conditional operations.
- Execution is slow.
- It does not have any API.
- It does not provide any mechanism for error handling.
- Dynamic web applications are not used to test.

– By Tushar Sir

Selenium RC

- RC stands for Remote Control. It allows the programmers to code in different programming languages like C#, Java, Perl, PHP, Python, Ruby, Scala, Groovy. The figure shows how the Remote Control Server works.



– By Tushar Sir

Selenium RC

Pros:

- It supports all web browsers.
- It can perform iteration and conditional operations.
- Execution is faster as compared to IDE.
- It has built-in test result generators.
- It supports data-driven testing.
- It has a matured and complete API.
- There is also support for cross browser testing.
- There is also support for user preferred languages.

– By Tushar Sir

Selenium RC

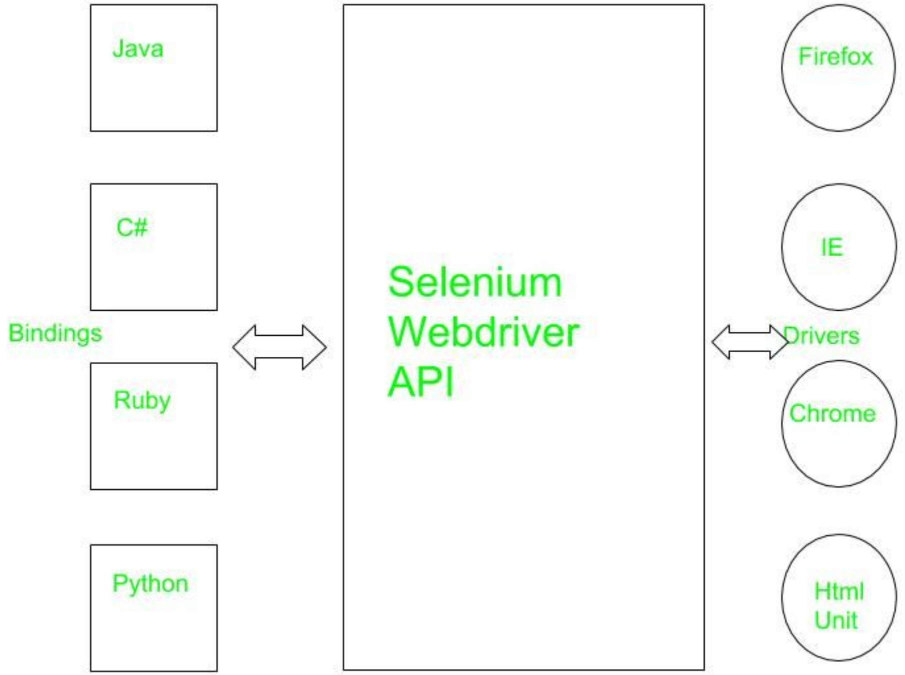
Cons:

- Programming language knowledge is needed.
- It does not support testing for IOS/Android.
- It is a little slower than Selenium Webdriver in terms of execution.
- It does not support record and playback functions.
- Complicated configuration than selenium IDE.
- The API in selenium RC are quite confusing.

– By Tushar Sir

Selenium Webdriver

- Selenium Web Driver automates and controls initiated by the web browser. It does not rely on JavaScript for automation. It controls the browser directly by communicating with it. The figure shows how web driver works as an interface between Drivers and Bindings.



– By Tushar Sir

Selenium Webdriver

Pros:

- It directly communicates with the web browser.
- Execution is faster.
- It supports listeners.
- It supports IOS/Android application testing.
- Installation is simpler than Selenium RC.
- Purely object-oriented.
- No need of any separate Component.

– By Tushar Sir

Selenium Webdriver

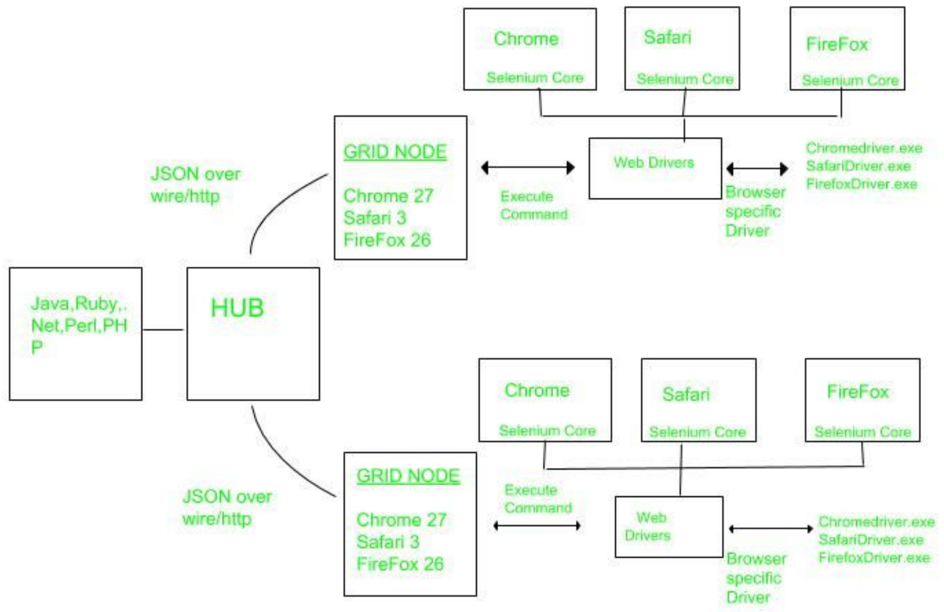
Cons:

- It requires programming knowledge.
- There’s no built-in mechanism for the generation of the test result file.
- Installation process is quite complicated than selenium IDE.
- There is no support for any new browsers.
- There is no detailed test reports.

– By Tushar Sir

Selenium Grid

- Basically, it is a server that allows the test to use a web browser instance running on remote machines.
- It provides the ability to run the test on a remote web browser, which helps to divide a load of testing across multiple machines and it will save enormous time.



– By Tushar Sir

Selenium Grid

- It allows executing parallel tests across different platforms and operating systems.
- Selenium Grid is a network of HUB & nodes.
- Each node registers to the HUB with a certain configuration and HUB is aware of the browsers available on the node.
- When a request comes to the HUB for a specific browser [with desired capabilities object], the HUB, if found a match for the requested web browser, redirects the call to that particular Grid Node and then a session is established bi-directionally and execution starts.
- It makes the easy use for multiple machines to run the tests in parallel.

– By Tushar Sir

Features of Selenium

- **Cross-browser compatibility:** Selenium supports testing on multiple browsers like Chrome, Firefox, Safari, Edge, and Internet Explorer.
- **Language support:** Selenium supports multiple programming languages like Java, Python, C#, Ruby, and JavaScript, making it easy for developers to write automation scripts in their preferred language.
- **Multiple testing frameworks:** Selenium can integrate with multiple testing frameworks like JUnit, TestNG, and NUnit.
- **Record and playback:** Selenium provides the option to record and playback test scripts, which makes it easy for testers to create test cases without having to write code.
- **Parallel execution:** Selenium can execute test cases in parallel across multiple machines, which reduces the overall execution time.

– By Tushar Sir

Features of Selenium

- **Element identification:** Selenium can identify web elements using various locators like ID, Name, XPath, CSS Selector, and Class Name.
- **Handling dynamic elements:** Selenium can handle dynamic web elements like dropdowns, pop-ups, and alerts.
- **Integration with third-party tools:** Selenium can integrate with various third-party tools like Jenkins, Docker, and Appium.
- **Support for mobile testing:** Selenium can also be used for mobile testing using Appium.

– By Tushar Sir

Advantages of Selenium

1. Selenium is pure open source, freeware and portable tool.

2. Selenium supports variety of languages that include Java, Perl, Python, C#, Ruby, Groovy, Java Script, and VB Script. etc.

3. Selenium supports many operating systems like Windows, Macintosh, Linux, Unix etc.

4. Selenium supports many browsers like Internet explorer, Chrome, Firefox, Opera, Safari etc.

5. Selenium can be integrated with ANT or Maven kind of framework for source code compilation.

6. Selenium can be integrated with TestNG testing framework for testing our applications and generating reports.

– By Tushar Sir

Advantages of Selenium

7. Selenium can be integrated with Jenkins or Hudson for continuous integration.

8. Selenium can be integrated with other open source tools for supporting other features.

9. Selenium can be used for Android, IPhone, Blackberry etc. based application testing.

10. Selenium supports very less CPU and RAM consumption for script execution.

– By Tushar Sir

Disadvantages of Selenium

1. Selenium needs very much expertise resources. The resource should also be very well versed in framework architecture.

2. Selenium only supports web based application and does not support windows based application.

3. It is difficult to test Image based application.

4. Selenium need outside support for report generation activity like dependence on TestNG or Jenkins.

5. Selenium does not support built in add-ins support.

6. Selenium user lacks online support for the problems they face.

– By Tushar Sir

Disadvantages of Selenium

7. Selenium does not provide any built in IDE for script generation and it need other IDE like Eclipse for writing scripts.

8. Selenium script creation time is bit high.

9. Selenium does not support file upload facility.

10. Selenium partially supports for Dialog boxes.

– By Tushar Sir

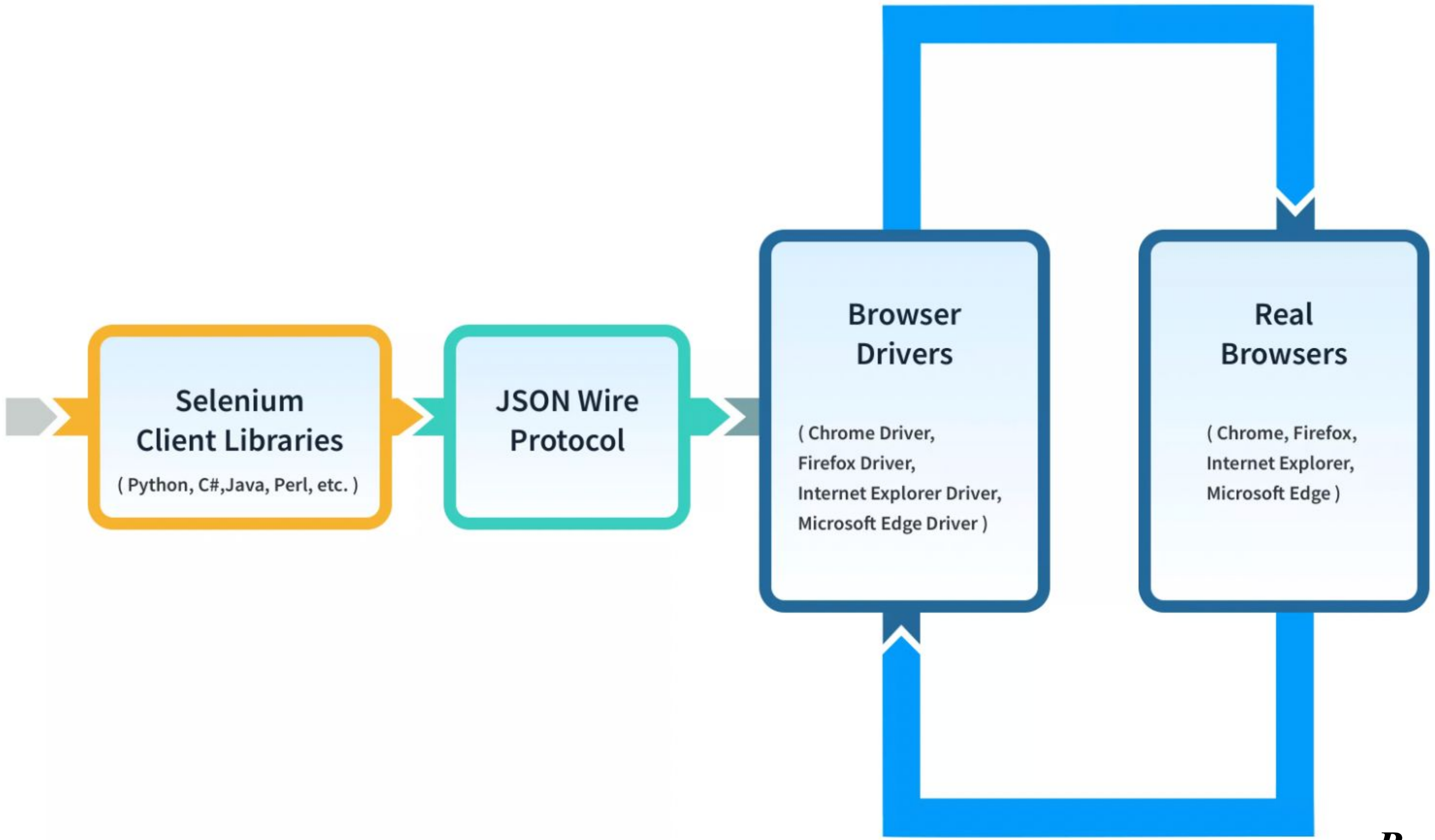
Architecture of Selenium

Selenium WebDriver Architecture is made up of four major components:

- **Selenium Client library:** Selenium provides support to multiple libraries such as Ruby, Python, Java, etc as language bindings
- JSON wire protocol over HTTP: JSON is an acronym for JavaScript Object Notation. It is an open standard that provides a transport mechanism for transferring data between client and server on the web.
- Browser Drivers: Selenium browser drivers are native to each browser, interacting with the browser by establishing a secure connection. Selenium supports different browser drivers such as ChromeDriver, GeckoDriver, Microsoft Edge WebDriver, SafariDriver, and InternetExplorerDriver.
- Browsers: Selenium provides support for multiple browsers like Chrome, Firefox, Safari, Internet Explorer etc.

– By Tushar Sir

Architecture of Selenium



– By Tushar Sir

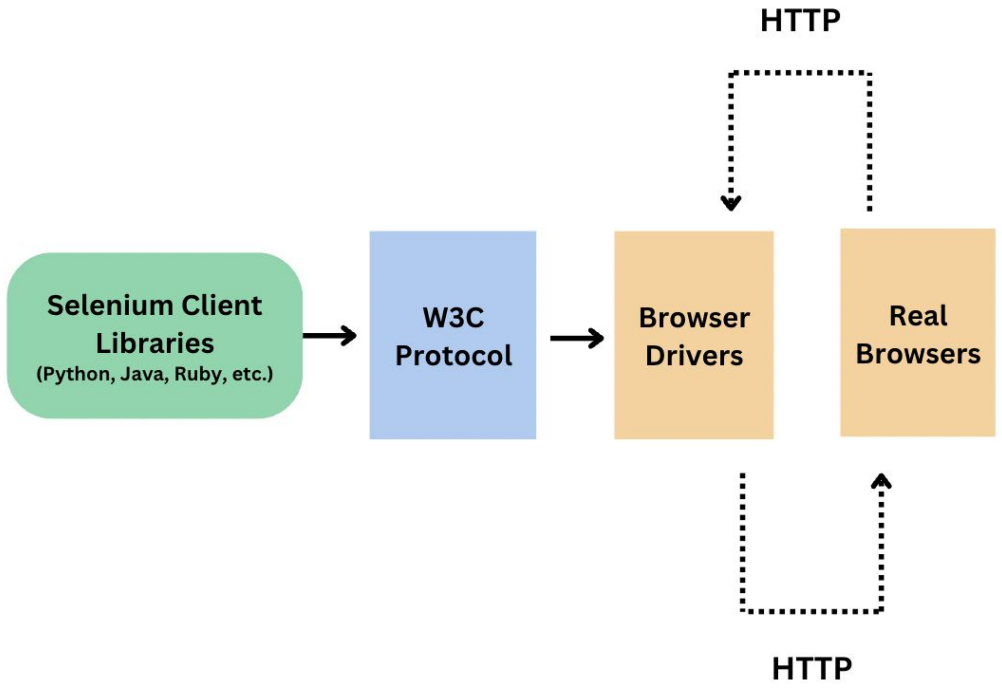
Architecture of Selenium

- In Selenium 3, there is no direct communication between the client libraries (Java, Python, JavaScript, etc) and the browser drivers.
- The server (browser drivers) does not understand language but only the protocols and on the other hand, client libraries does not understand protocols used by browser drivers.
- Therefore, JSON Wire protocol is being used as a mediator between client and server to encode and decode the requests and responses made by client and server respectively.
- This results in limited browser interaction, inefficient communication and lack of standardization which ultimately led to flaky test and slower test execution.

– By Tushar Sir

Architecture of Selenium

- The architecture of Selenium 4 is similar to Selenium 3, however it uses W3C protocol instead of JSON wire protocol for communication between Client Libraries and Browser Drivers.



– By Tushar Sir

Architecture of Selenium

- WebDriver in Selenium 4 is fully W3C compliant!
- W3C stands for the World Wide Web Consortium, an international community that develops and maintains standards and guidelines for the World Wide Web.
- The main aim of the W3C is to ensure the long-term growth and interoperability of the Web.
- It creates open standards and specifications that promote compatibility and consistency across various web technologies and platforms.
- And when we say Selenium 4 is W3C compliant it states that Selenium adheres to the standards and specifications laid by the W3C for web automation.

– By Tushar Sir

Architecture of Selenium

- All the browsers and the browser drivers in Selenium architecture follow W3C, except Selenium 3 WebDriver.
- And hence, JSON Wire Protocol is used to encode and decode the requests and responses.
- Selenium 4 WebDriver was made W3C compliant to make the communication easy and direct between the client libraries and the browser drivers.
- Improved communication led to more stability.
- This has also enhanced browser compatibility, performance and efficiency as there is no overhead of HTTP requests and responses for communication between the WebDriver client and the browser driver.
- Instead, WebDriver now utilises native browser communication channels and protocols.

– By Tushar Sir

Selenium Client Library

- The Selenium client library is a set of functions that run Selenium commands from your own program. It consists of languages like Java, Ruby, Python, and C#.
- The Selenium client library provides a programming interface (API).
- After test cases are triggered, the entire Selenium code is converted to JSON format. JSON stands for Javascript Object Notation. It transfers information from the server to the client.
- Selenium provides support to multiple libraries such as Ruby, Python, and Java. Selenium developers have developed language bindings to provide compatibility for multiple languages. For instance, if you want to use the browser driver in Python, use the Python Bindings.

– By Tushar Sir

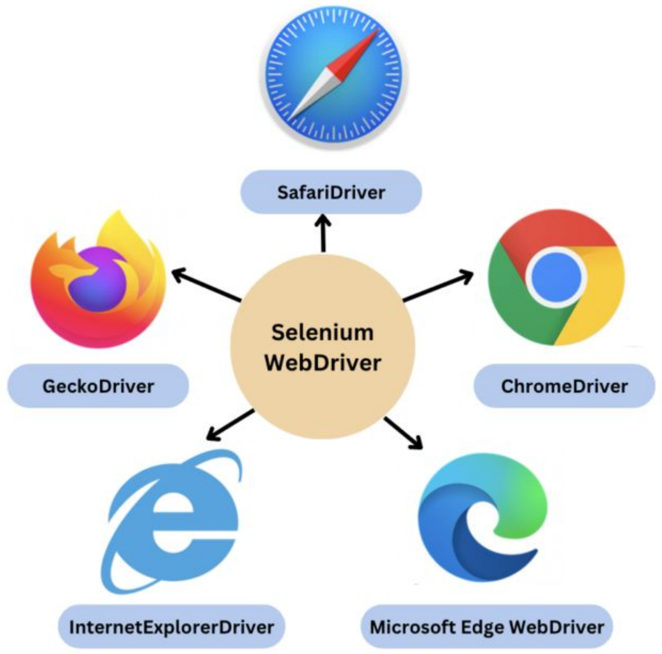
JSON Wire Protocol over HTTP

- **The JSON wire protocol (JSONWP)** is a transport mechanism that uses JSON over HTTP to define a RESTful web service.
- It acts as a mediator between client libraries and Web Drivers. JSONWP is used to test websites automatically using browsers like Firefox, IE, and Chrome.
- JSONWP uses serialization and de-serialization to transfer data between the client and the server. Serialization converts object data to JSON format, and de-serialization converts JSON format to object.
- JSONWP is an abstract specification of how automation behavior is mapped to selenium or appium or HTTP requests and response. It includes a specific set of predefined, standardized endpoints exposed via a RESTful API.

– By Tushar Sir

Concepts of Browser Drivers

- The browser driver is the key to receiving the HTTP requests from the JSON Wire Protocol through its HTTP server and sends the processed requests to the real browser. This is where the interaction with elements takes place in any operation.
- Selenium provides drivers specific to each browser and without revealing the internal logic of browser functionality, the browser driver interacts with the respective browser by establishing a secure connection.
- These browser drivers are also specific to the language which is used for test case automation like C#, Python, Java, etc.



– By Tushar Sir

Concepts of Browser Drivers

When a test script is executed with the help of WebDriver, the following tasks are performed in the background:

- An HTTP request is generated and it is delivered to the browser driver for every Selenium Command
- The HTTP request is received by the driver through an HTTP server
- All the steps/instructions to be executed on the browser is decided by an HTTP server
- The HTTP server then receives the execution status and in turn sends it back to the automation scripts

– By Tushar Sir

Thank You

- **By Tushar Sir**