



We are on a mission to address the digital skills gap for 10 Million+ young professionals, train and empower them to forge a career path into future tech

Selenium IDE



Overview

Table of Contents

- **Testing**
- **Manual Testing**
- **Automation Testing**
- **Understand Script**
- **Choosing an Automation Tool**
- **What is Selenium**
- **Selenium Components**
- **IDE Installation and Execution**
- **Locators IDE**
- **MCQ**

Overview

Testing

Testing :

Testing is the process to check whether the actual software product matches expected requirements and to ensure that software product is defect free.

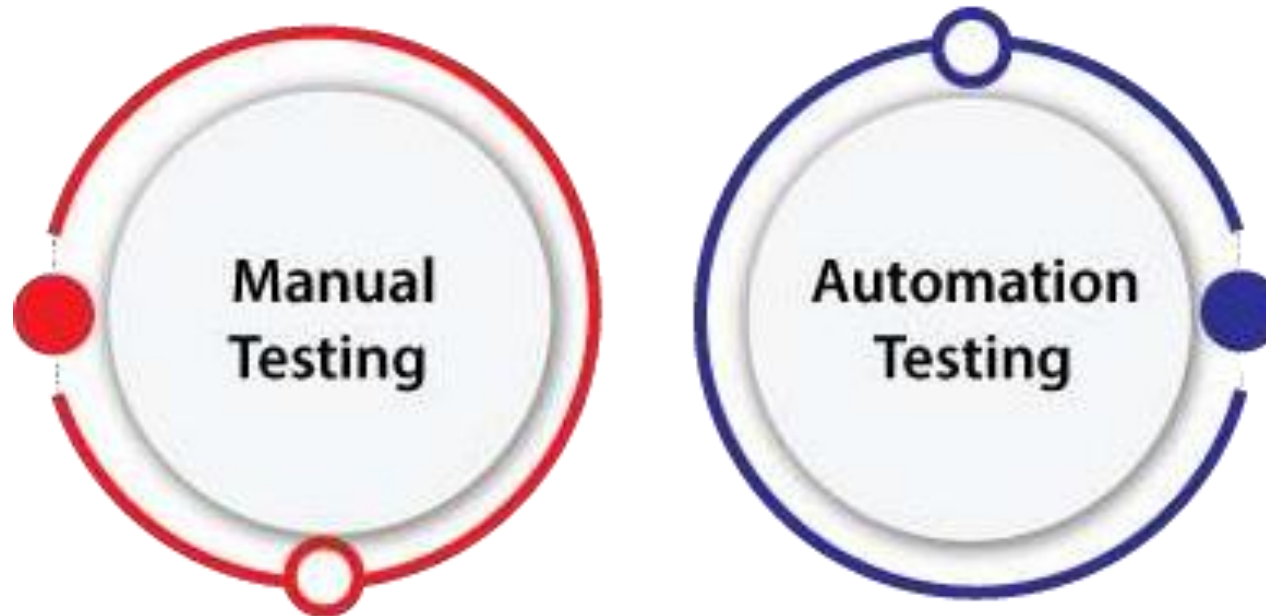
- Actual product
- Expected requirement
- Defect free

Overview

Types of Testing

Software Testing :

Types of Software Testing



Overview

Manual Testing

- Testing the software or application repeatedly or again and again manually in order to find defects in the software according to customer requirements is called as Manual Testing.
- Example : Gmail.



overview

Manual Testing

➤ **Manual Testing :**

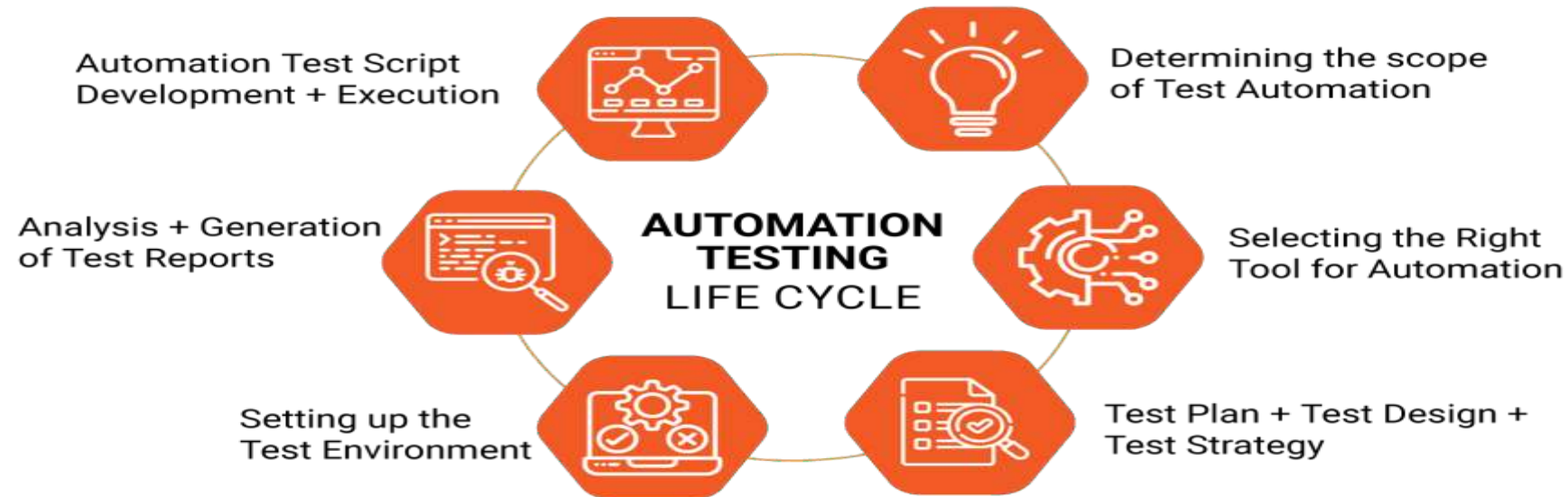
- When it's a short term project
- We build small or simple application
- When the requirements are changing

Overview

Automation Testing

Automation Testing :

Testing the functionality of the application using a software is Test Automation.



Overview

Automate

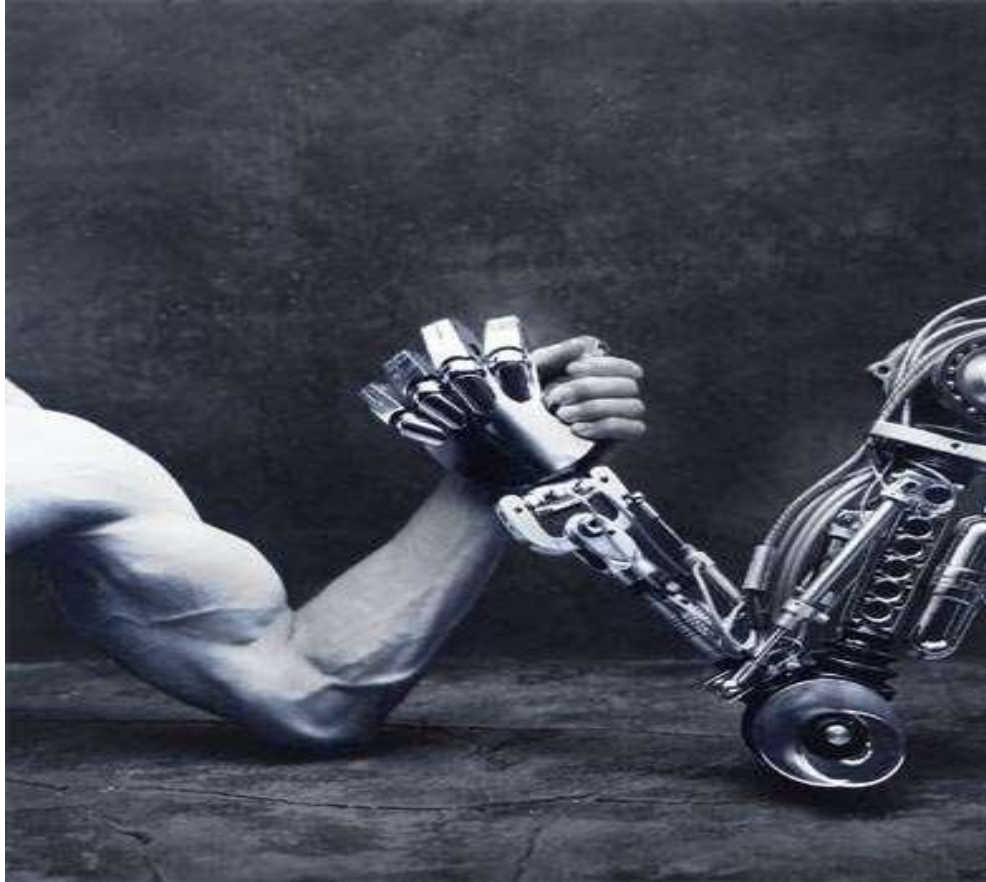
➤ **Automate:**

- When there is need for regression
- Cross platform testing
- Lots of data with one time execution
- To test for performance and security
- The software has multiple versions and releases.
- It is cost effective in long run.
- Optimization in costs occurred due to manual errors.
- Atleast 1 build has to be released

Overview

Testing

Testing : Manual vs Automation

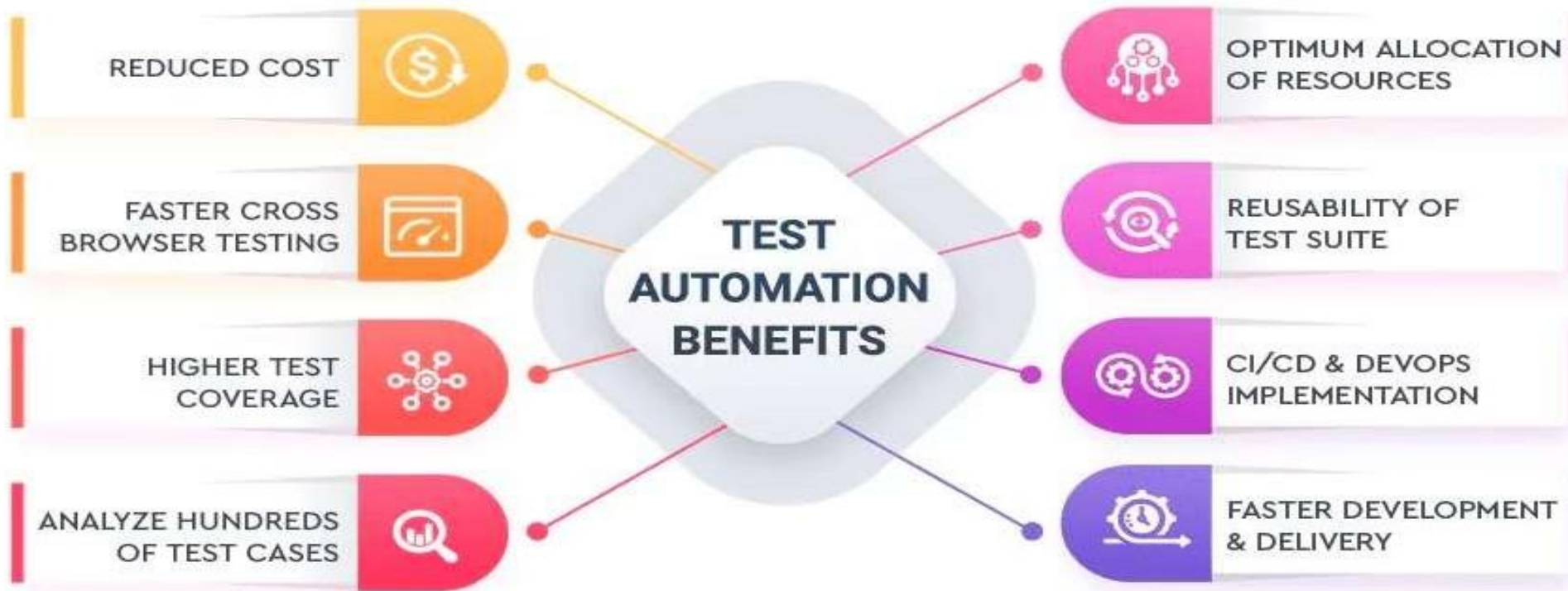


Pros of Manual Testing

- If the test case only runs twice a coding milestone, it most likely should be a manual test. Less cost than automating it.
- Ad-hoc testing (random testing): It allows the tester to perform more ad-hoc (random testing). In my experiences, more bugs are found via ad-hoc than via automation. And, the more time a tester spends playing with the feature, the greater the odds of finding real user bugs.
- Coding knowledge not required

Pros of Automation Testing

- **Pros of Automation Testing :**



Cons of Manual

- **Cons of Manual :**
- Running tests manually can be very time consuming
- Each time there is a new build, the tester must rerun all required tests - which after a while would become very tiresome work
- Resource utilization is more
- No consistency in testing

Cons of Automation

➤ Cons of Automation

- High investment in initial stages of training
- Documentation is essential
- No adhoc testing
- Man power requirement for test preparations
- Many test scenarios might be left uncovered

Automation Tool

➤ Choosing an Automation Tool :

- Here's a quick comparison of QTP, Selenium and Open Script.
- I had done before choosing Selenium. I hope this will be useful to you as well.

Feature	Open Script	Selenium	QTP
Browser Support	IE & Firefox	IE, Firefox, Safari, CHROME	IE & Firefox
OS Support	Windows & Unix based	Windows, Mac, Unix based	Windows only
Scripting Language	Java only	HTML, Java, c#, perl, PHP, Python, ruby	VBScript
License	Oracle owned product	Free and open source	Not free. PHP proprietary product

Automation Tool

➤ Choosing an Automation Tool :

Simulation method	Simulates browser actions, hence browser can be run in background	Simulates browser actions, hence browser can be run in background	Simulates end user actions, hence the browser must be visible and running in foreground.
Validation/checkpoint support	Text matching, Object matching, Server response matching, Database query match, Table match checkpoints supported with wildcard and regex based checks.	Provides 'verify' and 'assertion' checkpoint for the accessors various properties of an identified HTML element These accessors have to made use of through the scripting language to create custom checkpoints. Hence text matching, object match, table matching and database can be implemented	Standard Text, Text Area, Image, Xml, Database checkpoints available by default. Apart from these, Bitmap checkpoint is also supported, which is not supported by Openscript, selenium
Types of code views	Tree view and code view present	Tree view is only present if HTML scripting is used. Otherwise only code view is present.	Keyword view and expert view(code view) are present.

Automation Tool

➤ Choosing an Automation Tool :

Screenshots and HTML capture	Automatically captures screenshots and HTML during recording & playback	Does not capture by default. The command <code>captureEntirePageScreenshot()</code> must be used wherever screenshot is required.	Automatically captures screenshots and HTML during recording & playback
HTML report	Automatically generates HTML reports for each step group	Does not generate HTML report. But can generate XML,JSON log files which can be parsed and converted to HTML reports.	Automatically generates HTML reports for each step
Object library/repository	Provides inbuilt object library in the form of *.properties files. Also provides a tool to merge repositories.	Does not provide in built object library.	Provides object repository and tools to compare and merge repositories.

Automation Testing Tools

➤ Automation Testing Tool :



What is Selenium ?

➤ **Selenium :**

- Selenium is a free open source automation testing suite
- Selenium Software is not just a single tool but a suite of software
- Primarily Selenium was created by Jason Huggings in 2004 and later improvised by many others

What is Selenium ?

➤ Selenium :

- **Why is it named Selenium?**

Popular automated testing framework made by Mercury Interactive and since Selenium is an antidote for mercury poisoning Jasson suggested this name.

- **Selenium supports**

- Firefox, Chrome, Safari, IE
- Windows, Linux, Mac
- Python, C#, Ruby, Java, PHP, Pearl, Groovy.

Components of Selenium

➤ Components Selenium :

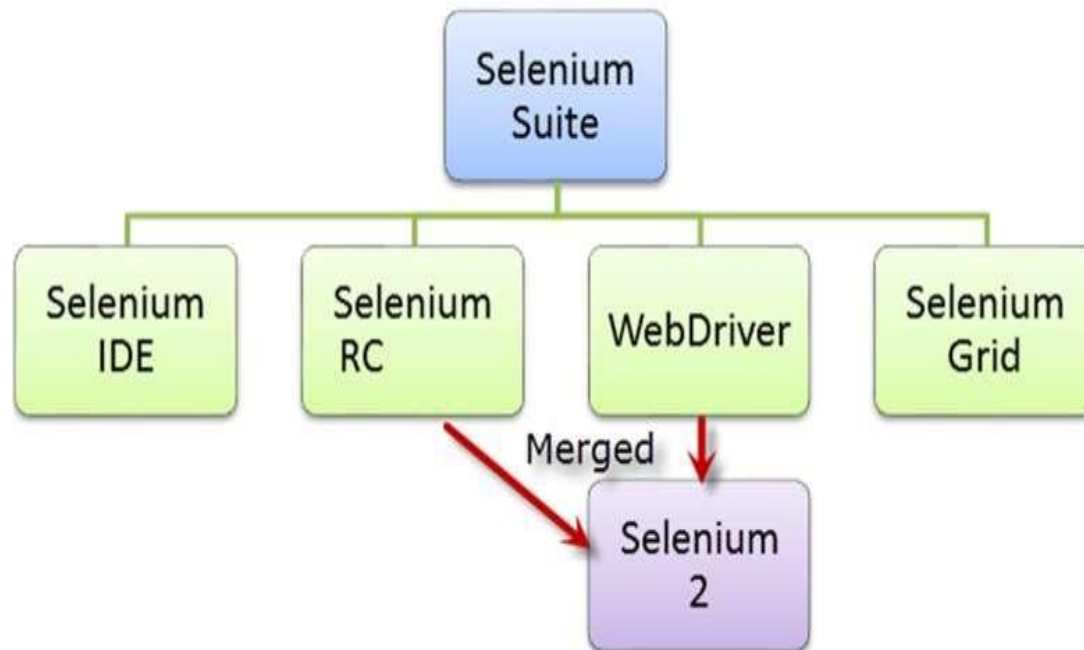
- Integrated Development Environment (IDE)-□ a Firefox and Chrome extension that can automate the browser through a record-and-playback feature
- Selenium RC – Testers have to download whole application under test and web server on local machine. So Paul Hammat, created a RC server that acts in between browser and the code. We start & stop the server before and after execution
- Web Driver□ It was the first cross-platform testing framework that could control the browser from the OS level.

GRID - It was capable of capturing browser screenshots during significant stages, and also of sending out Selenium commands to different machines simultaneously.

- **Web Driver Latest version – 3.141.0 (Nov 01 2018)**

Components of Selenium

➤ Components Selenium :



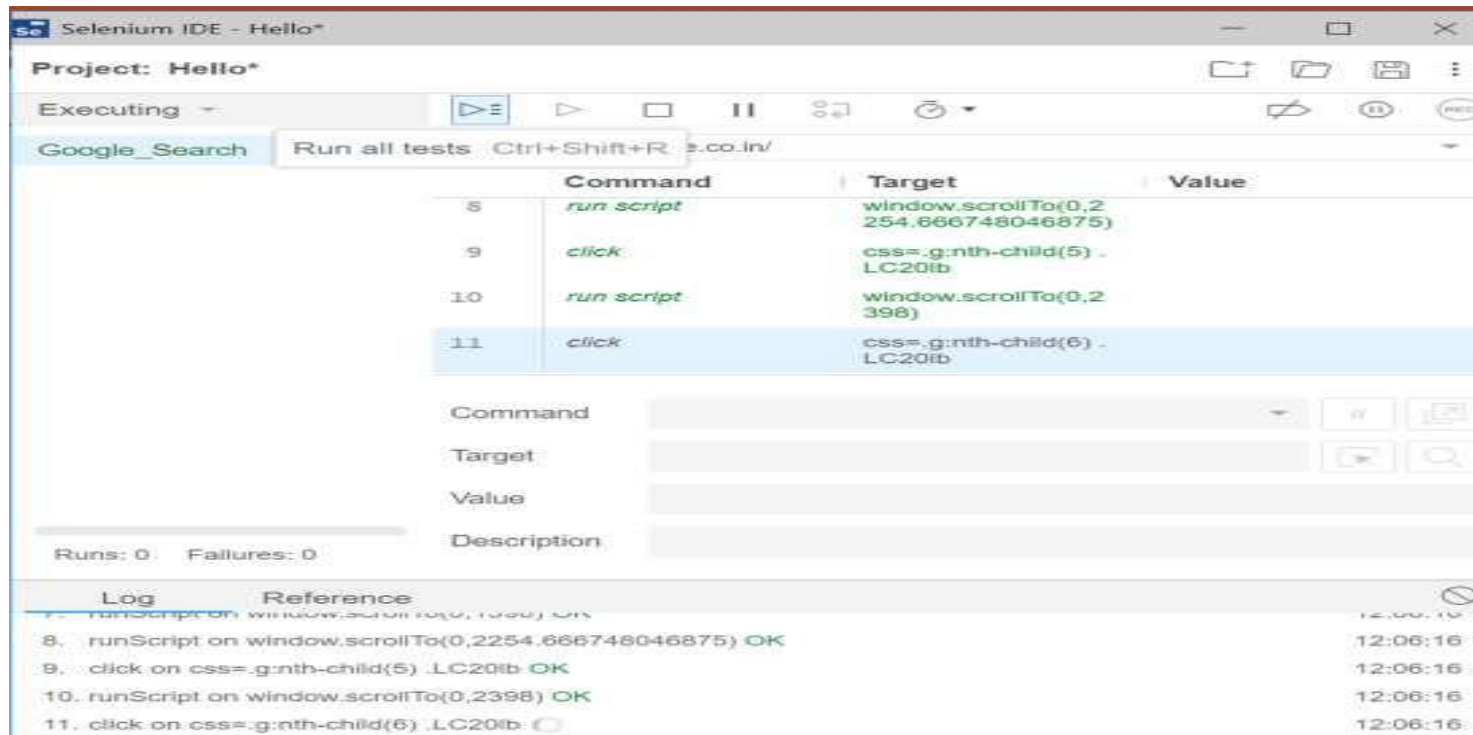
Selenium IDE - Installation

➤ Selenium IDE - Installation :

- **Download the Selenium IDE from the <http://seleniumhq.org/download/>**
- Install the Selenium IDE add-on in the Firefox browser
- Click on the 'Install Now' in the add-on window pop-up
- Restart the Firefox browser
- Go to the Tools menu and check it the Selenium IDE
- Click on the Selenium IDE, it opens the Selenium IDE tool.

Selenium IDE - Introduction

➤ Selenium IDE - Introduction :



Selenium IDE - Introduction

➤ **Selenium IDE - Introduction :**

- ✓ Selenium IDE is a browser based extension to record and play back.
- ✓ No programming experience is required, though knowledge of HTML and DOM are required.
- ✓ Designed for Firefox, Chrome, IE
- ✓ Recorded generate changed to Java, C#, PHP, Python, Ruby language

Selenium IDE – Features

➤ **Selenium IDE - Introduction :**

- ✓ Selenium IDE is a browser based extension to record and play back.
- ✓ No programming experience is required, though knowledge of HTML and DOM are required.
- ✓ Designed for Firefox, Chrome, IE
- ✓ Recorded generate changed to Java, C#, PHP, Python, Ruby language

Selenium IDE – Features

➤ Menu Bar

- Create Test case and Test Suite
- Save/Open the Test case
- Copy, Paste and Delete the command
- Record and Play, Clipboard Format

➤ Action Bar :



Speed Control: controls how fast your test case runs.

Run All: Runs the entire test suite

Selenium IDE – Features

Action Bar



Runs the Currently Selected test.



Allows stopping and re-starting of a running test case

Allows you to “step” through a test case

Records the user’s browser actions

Selenium IDE – Features - CONTD

➤ **Test case pane**

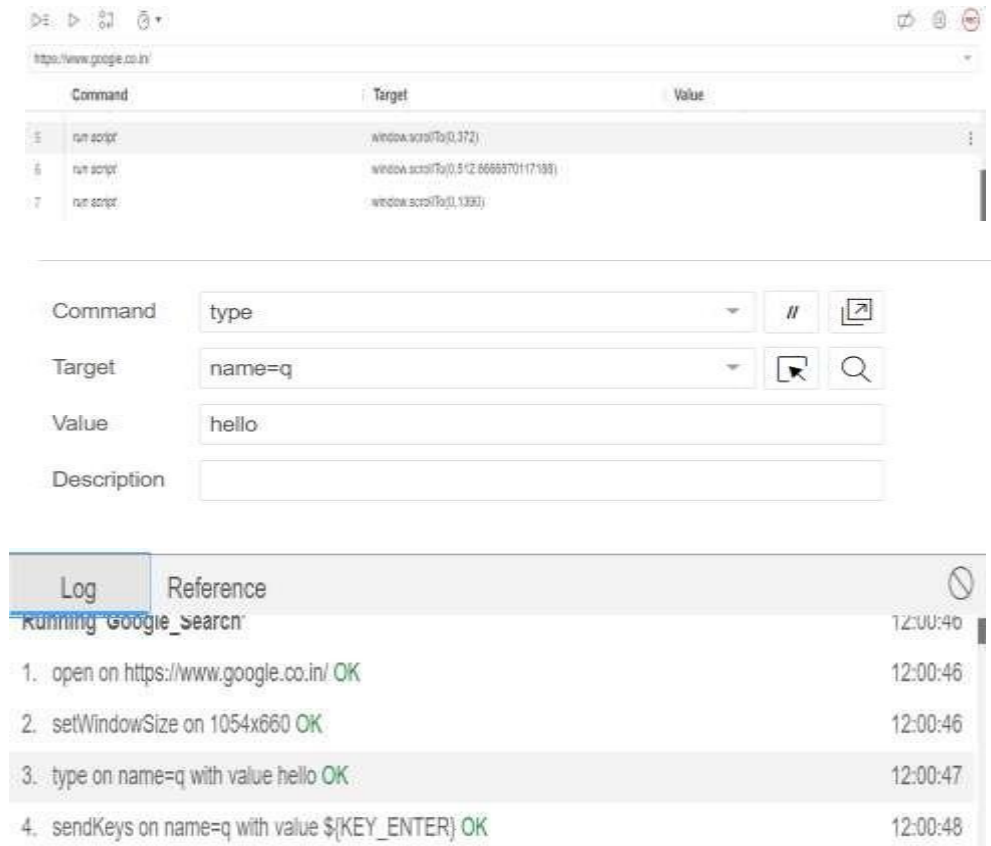
- The Command, Target, and Value entry fields display the currently selected command along with its parameters

➤ **Log/Reference**

- Log -Test case, error messages and information messages are displayed, useful for debugging
- Reference - Reference pane will display documentation on the selenese command

Selenium IDE – Features - CONTD

Features :



The screenshot displays the Selenium IDE interface. At the top, a browser address bar shows 'https://www.google.co.in/'. Below it is a table of commands:

Command	Target	Value
run script	window.scrollTo(0,372)	
run script	window.scrollTo(0.512,666670117198)	
run script	window.scrollTo(0,1393)	

Below the table is a command editor with the following fields:

- Command: type
- Target: name=q
- Value: hello
- Description: (empty)

At the bottom is a log window with two tabs: 'Log' (selected) and 'Reference'. The log shows the following entries:

Log	Reference	Time
Running Google_Search		12:00:40
1. open on https://www.google.co.in/ OK		12:00:46
2. setWindowSize on 1054x660 OK		12:00:46
3. type on name=q with value hello OK		12:00:47
4. sendKeys on name=q with value \${KEY_ENTER} OK		12:00:48

Selenium IDE – Features - Exporting

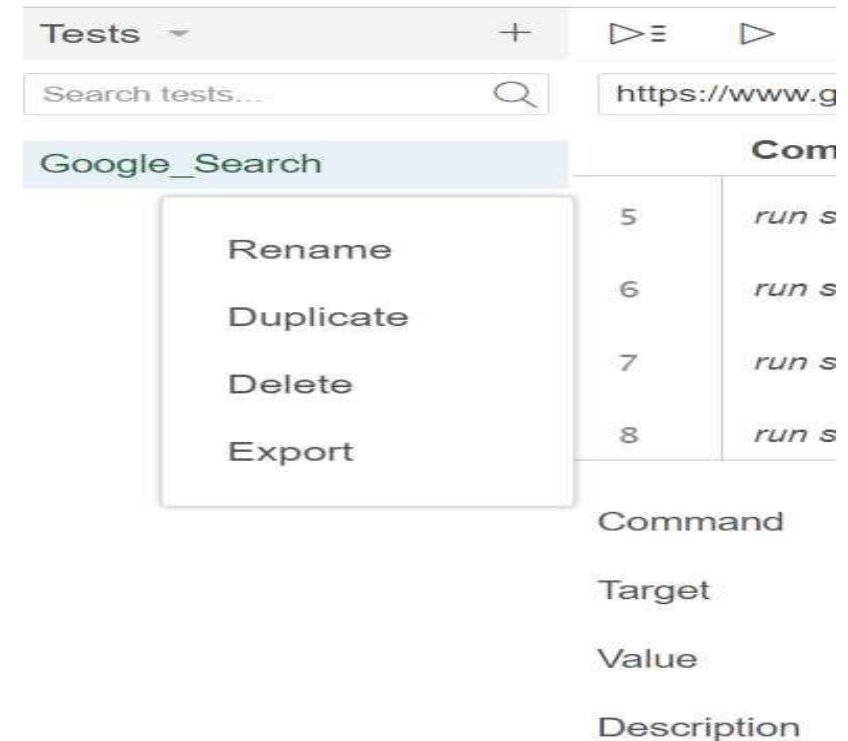
Exporting :

A most interesting feature of Selenium IDE: export to Selenium RC.

Selenium IDE is able to export test cases in the following formats:

- **C# NUnit**
- **C# xUnit**
- **Java JUnit**
- **JavaScript Mocha**
- **Python pytest**
- **Ruby Rspec**

You can export test-cases or the entire test-suite.



Summary

➤ Summary :

- **Need for Automation Testing**
- **Steps involved in choosing the automation framework.**
- **How to start with Selenium IDE.**

Summary

➤ **Summary :**

- **Need for Automation Testing**
- **Steps involved in choosing the automation framework.**
- **How to start with Selenium IDE.**

Recording Test Cases



Table of Content

➤ Table of Content :

- Selenium IDE- Creating Test Cases
- Insert Commands
- Executing the Test Script
- Practices

Selenium IDE – Creating Test Cases

➤ Test Cases :

- We will create test cases by inserting selenium commands instead of recording option.
- For this test, we will search a text operation on any publically available search engine (say "Google"). Subsequently, we will create a Login test case in the same test suite.

Table of Content

➤ Table of Content :

- Selenium IDE- Creating Test Cases
- Insert Commands
- Executing the Test Script
- Practices

Insert Commands

➤ Insert Commands :

- Launch Firefox browser.
- Click on the Selenium icon present on the top right corner on your browser.
- It will launch the default interface of Selenium IDE.
- Enter the project name as "Manual Test".
- Enter the test case name as "Search Test".
- Click on the command text box present on the Test Script Editor Box.

Insert Commands

➤ Insert Commands :

- Modify the properties of First command as:
- Command : open
- Target : <https://www.google.co.in>
- During execution of the test case, this command will load the Google search engine web page on your Firefox browser.

Insert Commands

➤ Insert Commands :

Playback base URL ▼

Command	Target	Value

Command
 Target
 Value
 Comment

↓

add selection
 answer on next prompt
 assert alert
 assert checked
 assert not checked
 assert confirmation

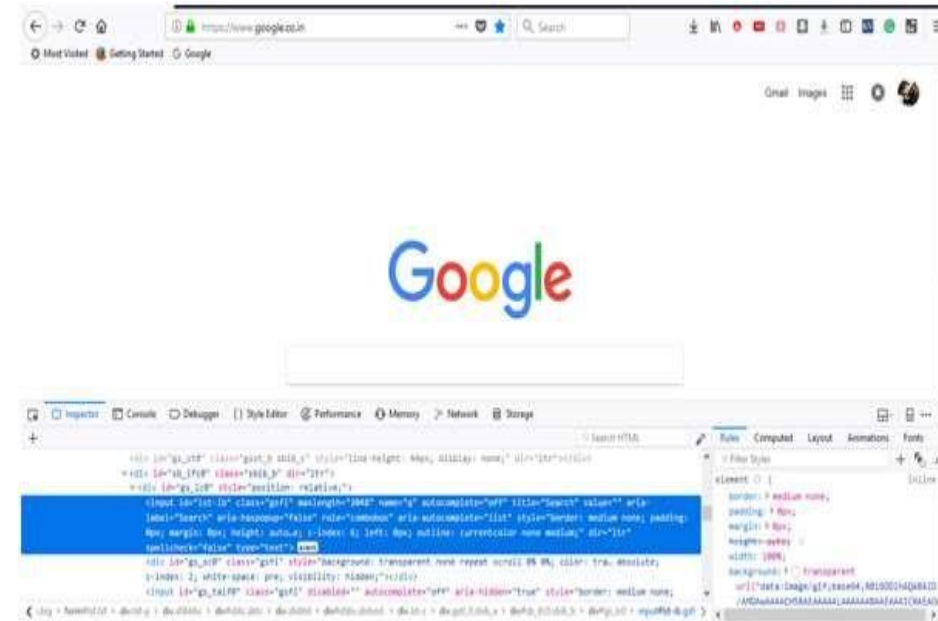
Insert Commands

➤ Insert Commands :

- Now, we have to add a command that will click on the Google search engine text box. For this, we need a unique identification element of the text box which will help the IDE to identify the target location.
- The method for finding a unique identification element involves inspection of HTML codes.
- Open URL: <https://www.google.co.in> your Firefox browser.
- Right click on the Google search text box and select Inspect Element.

Insert Commands

➤ **Insert Commands :**



Insert Commands

- **Second command as:**
 - Command :click at
 - Target : name=q
 - During execution of the test case, this command will click on the search text box present on the Google search engine web page.

Insert Commands

- **Third command as:**
 - Command : type
 - Target : name=q
 - Value: javaTpointJavaFX tutorial
 - During execution of the test case, this command will type the specified text on the Google search text box.

Insert Commands

➤ **Third command as:**

- Now add a command which will generate a button click event on our web page. For this event to be generated, we need a unique identification element for the Google search button.
- Right click on the Google search button and select Inspect Element.

Insert Commands

➤ Fourth command as:

- Command : click at
- Target : name=btnK
- During execution of the test case, this command will click on the search button present on the Google
search engine web page.

Insert Commands

▶ ≡
▶
🔄
🕒 ▼
✂
⏸
REC

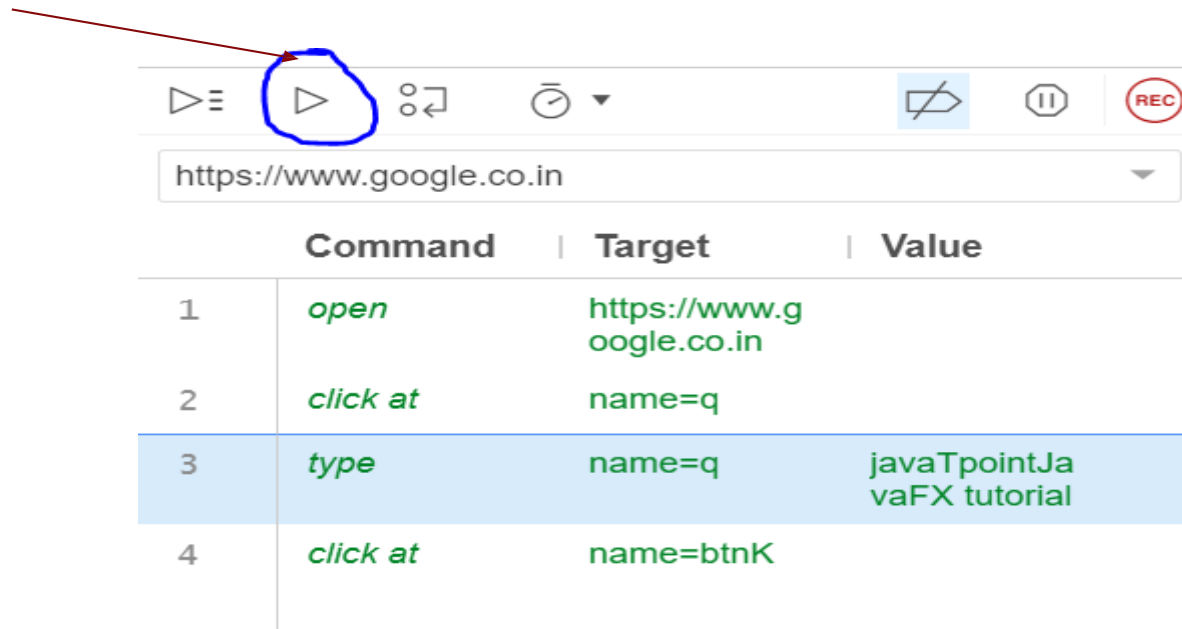
<https://www.google.co.in>

	Command	Target	Value
1	<i>open</i>	https://www.g oogle.co.in	
2	<i>click at</i>	name=q	
3	<i>type</i>	name=q	javaTpointJa vaFX tutorial
4	<i>click at</i>	name=btnK	

Test Script

Executing the Test Script :

- Click on the "Run Current Test" button present on the tool bar menu of the IDE.



Test Script

Executing the Test Script :

The Log pane displays the overall summary of the executed test scripts.

Runs: 1 Failures: 0		Description
Log	Reference	
1. open on https://www.google.co.in	OK	14:58:19
2. clickAt on name=q	OK	14:58:19
3. type on name=q with value javaTpointJavaFX tutorial	OK	14:58:23
4. clickAt on name=btnK	OK	14:58:23
'Test_Hotmail' completed successfully		14:58:23

Introduction to Selenium Commands - Selense



Introduction to Selenese Command

➤ **Selenese Command :**

- Selenese – The language of Selenium or Commands of Selenium
- Selenese consists of Actions, Accessors, Element Locators and Variables

➤ **Actions – Combination of Command, Target and Value**

- **Command : Scripts perform a particular action on a web application**

- **Target** : Typically take element locator
- **Value** : Value for any input

Example:

Open - Open a URL

Click – Click button, link, etc...

Type – Type text in text field

Introduction to Selenese Command

- There are only about 6 commands you need to know to be able to automate tests. Same as a manual test case really (open, click, select, type, assert (check), wait).
- **Selenium commands are basically classified in three categories:**
 - **Actions**
 - Actions are the selenium commands that interact directly with the application
 - **Accessors**
 - Accessors are the selenium commands that examine the state of the application and store the results in variables. They are also used to automatically generate Assertions. Allows user to store certain value to user defined variable
 - **Assertions**
 - Assertions are the commands that enable testers to verify the state of the **application with the expected state**

Actions

➤ Actions:

- Actions are the selenium commands that generally manipulate the state of the application

Command/Syntax	Description
open (url)	It launches the desired URL in the specified browser and it accepts both relative and absolute URLs.
type (locator,value)	It sets the value of an input field, similar to user typing action.
typeKeys (locator,value)	This command simulates keystroke events on the specified element.
click (locator)	This command enables clicks on a link, button, checkbox or radio button.
clickAt (locator,coordString)	This command enables clicks on an element with the help of locator and co-ordinates
doubleClick (locator)	This command enables double clicks on a webelement based on the specified element.
focus (locator)	It moves the focus to the specified element
highlight (locator)	It changes the background color of the specified element to yellow to highlight is useful for debugging purposes.

Actions

➤ Actions:

<code>close()</code>	This command simulates the user clicking the "close" button in the title bar of a popup window or tab.
<code>store</code> <code>(expression,variableName)</code>	This command specifies the name of a variable in which the result is to be stored and expression is the value to store
<code>waitForCondition</code> <code>(script,timeout)</code>	This command executes the specified JavaScript snippet repeatedly until it evaluates to "true".

ACCESSORS

➤ Accessors:

Command/Syntax	Description
storeTitle (variableName)	This command gets the title of the current page.
storeText (locator, variableName)	This command gets the text of an element..
storeValue (locator,variableName)	This command gets the (whitespace-trimmed) value of an input field.
storeTable (tableCellAddress, variableName)	This command gets the text from a cell of a table.
storeLocation (variableName)	This command gets the absolute URL of the current page.
storeElementIndex (locator, variableName)	This command gets the relative index of an element to its parent (starting from 0).
storeBodyText (variableName)	This command gets the entire text of the page.
storeAllButtons (variableName)	It returns the IDs of all buttons on the page.
storeAllFields (variableName)	It returns the IDs of all input fields on the page.
storeAllLinks (variableName)	It returns the IDs of all links on the page.

ACCESSORS

➤ Accessors:

Command/Syntax	Description
storeTitle (variableName)	This command gets the title of the current page.
storeText (locator, variableName)	This command gets the text of an element..
storeValue (locator,variableName)	This command gets the (whitespace-trimmed) value of an input field.
storeTable (tableCellAddress, variableName)	This command gets the text from a cell of a table.
storeLocation (variableName)	This command gets the absolute URL of the current page.
storeElementIndex (locator, variableName)	This command gets the relative index of an element to its parent (starting from 0).
storeBodyText (variableName)	This command gets the entire text of the page.
storeAllButtons (variableName)	It returns the IDs of all buttons on the page.
storeAllFields (variableName)	It returns the IDs of all input fields on the page.
storeAllLinks (variableName)	It returns the IDs of all links on the page.

Practice Program - 1

Open the Page: <https://demo.opencart.com/>

- Click on the Phones & PDAs
- Click on HTC Touch HD
- Extract the Text “HTC Touch HD” from the loaded page and store it in the variable.

Solution :

Command	Target	Value
<i>open</i>	https://demo.opencart.com/	
<i>set window size</i>	1382x744	
<i>click</i>	linkText=Phones & PDAs	
<i>click</i>	linkText=HTC Touch HD	
<i>store text</i>	css=h1	x
<i>echo</i>	\${x}	

Practice Program - 2

Open the Page: <https://demo.opencart.com/>

- Store the Title and URL of the Page
- Use Assert method to check the expected URL, Title and Actual Value are same.

Solution :

Command	Target	Value
✓ <i>open</i>	https://demo.opencart.com/	
✓ <i>store title</i>		title
✓ <i>echo</i>	\${title}	
✓ <i>assert title</i>	Your Store	
✓ <i>execute script</i>	return window.location.href	urltext
✓ <i>assert</i>	urltext	https://demo.opencart.com/

Practice Program - 3

- Open the Page: <https://demo.opencart.com/>
- Click on the Phones & PDAs
- Click on HTC Touch HD and click on add to cart.
- Use Assert method to confirm that the product is successfully added to the cart

Solution :

Command	Target	Value
<i>open</i>	/	
<i>click</i>	linkText=Phones & PDAs	
<i>click</i>	linkText=HTC Touch HD	
<i>click</i>	id=button-cart	
<i>assert element present</i>	id=alert	

Practice Program - 4

Open the Page: <https://demo.opencart.com/index.php?route=checkout/voucher&language=en-gb>

Assert the Value of the amount is 1 using contains in xpath.

Verify the text “I understand that gift certificates are non-refundable”.

Solution :

Command	Target	Value
<i>open</i>	/	
<i>click</i>	linkText=Phones & PDAs	
<i>click</i>	linkText=HTC Touch HD	
<i>click</i>	id=button-cart	
<i>assert element present</i>	id=alert	

Selenium ide Java Script Commands

Command	Target	Value	Comment
Do...Repeat If	Javascript to evaluate		Similar to while, the the first "if" check is done at the end of the loop.
gotolf	Javascript to evaluate	Label	(Deprecated) If the expression evaluates to TRUE the execution jumps to LABEL, otherwise continues with the next command. Often used with !statusOK .
gotoLabel	Label		(Deprecated) Jumps to LABEL.
forEach...end	Javascript to evaluate		Loop over the content of an Javascript array.
if-elseif-else-end	Javascript to evaluate		The classic if-then-else conditional. If the expression evaluates to TRUE the "then" section is executed, otherwise the "else" section is executed. Often used with !statusOK
label	Label		(Deprecated) Defines the LABEL position for gotolf and gotoLabel to jump to.
times...end	Number		Loop "Number" times.
while...end	Javascript to evaluate		Executes the section between while...end as often/as long as the conditional statement evaluates to true.

Activate

Selenium ide Java Script Commands

Javascript (executeScript_Sandbox recommended)	Result
<code>return Math.floor(Math.random()*11)</code>	Random number between 0 and 10
<code>return Number (\${i}) + 1;</code>	Increase the value of i by one (very useful inside loops)
<code>var x="abc"; return x.length;</code>	Returns the length of the string (here 3).
<code>x="\${myvar}"; return x.length;</code>	Same as above but with variable as input.
<code>var d=new Date(); return d.getDate()+'-'+((d.getMonth()+1))+'-'+d.getFullYear();</code>	Get todays date
<code>var d= new Date(); var m=((d.getMonth()+1)<10)?'0'+(d.getMonth()+1):(d.getMonth()+1); return d.getFullYear()+"-"+m+"-"+d.getDate();</code>	Get todays date in YYYY-MM-DD format
<code>var d= new Date(); var month = d.toLocaleString('default', { month: 'long' }); return month+"-"+d.getDate() + "-" + d.getFullYear();</code>	Get todays date in MMM-DD-YYYY where MMM is the month's name e. g. May-1-2022
<code>var d= new Date(new Date().getTime() + 24 * 60 * 60 * 1000 * 5); var m=((d.getMonth()+1)<10)?'0'+(d.getMonth()+1):(d.getMonth()+1); m=d.getFullYear()+"-"+m+"-"+d.getDate(); return m</code>	Get yesterday's date (-1), tomorrows date (1) or the date in 5 days (5) => Just change the number 5 in the code snippet.
<code>return parseFloat(\$!runtime))-parseFloat(\${starttime});</code>	Calculate the difference between the current runtime and the start time. Very useful for measuring website performance. <i>!runtime</i> is a built-in variable, and <i>starttime</i> is a regular variable, we can fill it with <i>!runtime</i> e. g. at the macro start.

Practice Program 5

Write selenium ide javascript commands to print value from 1 to 5 using while loop

Solution :

Command	Target	Value
<i>store</i>	1	i
<i>while</i>	$\${i} \leq 5$	
<i>echo</i>	$i = \${i}$	
<i>execute script</i>	return Number ($\${i}$)+1	i
<i>end</i>		

Practice Program 6

Write selenium ide javascript commands to ,

- Verify whether the Availability of the stock of the mac book in <https://demo.opencart.com/>
- Verify the availability by visiting page at least 5 times continuously

Solution :

Command	Target	Value
<i>open</i>	https://demo.opencart.com/	
<i>store</i>	1	i
<i>while</i>	\${i}<=3	
<i>click</i>	linkText=MacBook	
<i>assert text</i>	css=.col-sm li:nth-child(4)	Availability: In Stock
<i>execute script</i>	history.back()	
<i>execute script</i>	return Number (\${i})+1	i
<i>end</i>		
<i>echo</i>	Executed \${i}	

and

Practice Program 7

Write selenium ide javascript commands to ,

- Open <https://demo.opencart.com/>
- Extract, only the price value from price element
- Convert it to number , increment it by 1 and print **it**.

Solution :

Command	Target	Value
<i>open</i>	https://demo.opencart.com/	
<i>store</i>	1	i
<i>while</i>	\${i}<=3	
<i>click</i>	linkText=MacBook	
<i>assert text</i>	css=.col-sm li:nth-child(4)	Availability: In Stock
<i>execute script</i>	history.back()	
<i>execute script</i>	return Number (\${i})+1	i
<i>end</i>		
<i>echo</i>	Executed \${i}	

and

Summary

- Need for Automation Testing
- Steps involved in choosing the automation framework.
- How to start with Selenium IDE.

THANK YOU