**Most commonly facing challenges in selenium automation**

The articulation is mainly about discussing the challenges that we come across at our Rota. When we say challenges, though its an impediment still it gives the opprurtunity to explore and that way we get introduced to many things

Basis the experience, I would like to half split this into two categories.

**1.       Anticipated challenges**

**2.       Un-anticipated challenges**

**Anticipated Challenges: -** I always reckon that the things that we are discussing here are not really challenges or some impediment that blocks us from proceeding. These should be considered as tasks those have to be performed by adding some additional coding actions or by integrating an additional concept to the selenium scripts .

1.       Improper waits

2.       Weak locators selection

3.       File uploads

4.       Send keys operation in IE

5.       Browser zoom settings in IE

6.       Zone mismatch setting in IE

7.       Windows authentication pop-up during URL launch

8.       Seldom windows interactions

9.       Sweet alert handles

10.   Dynamic data expectation by application

11. Taking snap during alert presence

12. Launching brower with options

Let’s see one after another to know, what each one is about

1**.       Improper waits: -** Basically when we are working with a web application through a browser we tend to see lot of synchronization problems, I mean there would be a significant problem in rendering the webpage that perhaps due to the poor and sluggish network provisions or at times the design of the webpage itself or sluggish server response i.e. not all the elements may not load at the same instance.

But by nature selenium’s swift is far high and try to interact with the element that it was instructed to do so and fails, because the element isn’t loaded properly so far. How do you handle this risk?

The above one is potentially a technical problem, but we have better concepts to achieve that. Based on need and the suitability with proper understanding any of the below wait concept is used

* Implicit wait – one mention at the start of script would help the wait applicable for all the subsequent actions
* Explicit wait – This is recommended only when we know that a particular element is expected to be delayed for presence. This wait is applicable to that one particular action, so that way we can ensure we are not waiting on other elements because of this
* Fluent wait- This is an advanced explicit wait, it features the recurrence with which the wait has to be executed with the help of polling time.

**2.       Improper choice Locators: -** The only possible way to interact with web elements is by identifying the locators in DOM. If the selection of those is improper that leads to a great mess. Careful study of when to go for what locator is much needed. Here below are some examples of weak locators.

**Improper locators:-**

* Do not use Id attribute if at all it has change of numbers/ if its generating dynamically.
* Do not use class name attribute if that has spaces in it.
* While writing Xpath matchers rather than going with start-with, end-with functions its always advisable to go with contains as its generic

**3.       File Uploads: -** On a few use cases, we might need to deal with uploading the file that’s available some other location also downloading the one from browser page. Upon understood the task, right choice of decision whether or not a third party tool integration is needed to perform, or do we have any libraries that performs tasks for us. Talking to the right set of people or seeking suggestions from experienced peers would help us achieving the task, rather than just shouting out as a blocker

**Possible solutions:-**

* Observe the DOM attributes for the file path text box in the webpage, if at all it is defined with a type parameter as “File”. We can directly send keys the file location (no need to click) that way we can upload file
* On the other hand if the possibility one doesn’t work, we will have to integrate the AutoIT / VNC viewer script to upload the file
* Using Robot class presented in java we can achieve the action as that helps impersonating the mouse and keyboard actions

**4.       IE problems: -** internet explorer being an inbuilt browser on windows machine is certainly not as much friendly as the other native browsers such as Chrome, Firefox..Etc. the most commonly seen challenges and the way to fix are

**         Send keys operation: -** This is a tedious task, unless we knew that we have a work around to use 32 bit version of InternetExplorerDriver.exe file, otherwise it is relatively slow with IE.

**         Zoom Settings: -** if at all zoom in is set to some higher value (perhaps 100). Element can’t be clicked as they are not inclined within the clickable window, just taking care of the setting would help us

**         Security Zones: -** The IE browser might not get launched if at all the settings were not set to the same protected mode.

**5.       Windows Interaction: -** On a few occasions windows interaction is unavoidable, however it’s a not very straight forwarded action still it’s mandatory to perform the action as it’s a business need. Considering them as challenge shout out is not desired, this again generates space for an individual to explore and integrate the other components. Following or some actions which are expected to be encountering

         Accessing the file that was downloaded

         Impersonating the mouse and key board actions

         Run a batch file that’s available in a specific path

**Solution:-** The possible solutions for any of the above solutions would be to use the Robot class present in java AWT package or to integrate the selenium code with anyother external tools such as AutoIT/ VNC viewer or coded UI

**6. Scrolling actions:-** At times the element that we are looking for may not be visible unless we scroll down to the webpage.

**Solution:** the possible solution would be through javaScriptExecutor which is an interface, help us getting the executeScript operation done through driver.

The fact that browser has bit of implementation of java script these actions were preferably done through java script. We can pass the below arguments to the executeScript method to scroll in browser window

* Scroll by method- we can actually specify the location of the elements in terms of co-ordinates, then pass those parameters so that we will have window scrolled down until that place
* ScrollTo- This is another offered method to scroll to a specific elements, we locate the element to which we wanted to scroll to then pass that locator as parameter
* ScrollHeight- Despite the element location, if we want to scroll down to the bottom of the page, this method can be used

**7. Windows Authentication pop-up:-** There are some applications which throws windows authentication popup after navigated to the URL, the pop up that comes is neither an alert nor blong the application to locate them in DOM.

**Solutions:-** There are two possible solutions for this problem,

* Pass the credentials in URL as <http://username:password@abc.com>
* On the other hand using of java script executor would help, we can do a sendkeys operation within the executeScript method to achieve this

8**.       Sweet Alerts: -** Alert handling is one of the popular mechanism in selenium, I feel many out there wouldn’t even come across these kinds of alerts (Sweet). These would basically be displayed for a few seconds and disappear. The real challenge with these elements is about how to identify them in DOM.

**Solution: -** In order to find them following procedure should be followed

         Mouse hover on the sweet alert to make that pop out.

         Freeze the DOM (shift+F8) when the element is present on UI, then the related attributes of the elements will be displayed to capture.

**9.       Dynamic data expectation by application: -** web applications day on day becoming more advanced, dynamic  in order to enhance the user experience. The data that present on one row today may not present at the same place tomorrow also many fields do not accept the duplicate data. Given this advanced behavior the static test data may not help. Generating the random numbers/ alphabets  and appending to the actual value would help avoiding such problems.

**10. Capturing snapshot during alert present:-**  Though we have takeScreenShotAs method also know of a procedure to capture the snap during each action, it is quite not possible with the case when we have an alert appeared on the window, in order to achieve this we should use the below coding snippet.

System.*setProperty*("webdriver.chrome.driver", "./drivers/chromedriver.exe");

ChromeDriver driver = **new** ChromeDriver();

driver.manage().window().maximize();

driver.get("https://www.irctc.co.in/eticketing/forgotPassword.jsf");

driver.manage().timeouts().implicitlyWait(30, TimeUnit.***SECONDS***);

driver.findElementById("forgot\_passwrd:checkDetails1").click();

Thread.*sleep*(1000);

// take snap

BufferedImage image = **new** Robot().createScreenCapture

(**new** Rectangle(Toolkit.*getDefaultToolkit*().getScreenSize()));

ImageIO.*write*(image, "png", **new** File("./data/Alert.png"));

}

11. Challenges during browser launch:- when we launch a browser to navigate to application URL, we might see the below problems

* Notification or info bars, that makes the script to fail
* Cache memory not cleared due to which can’t navigate to application home screen
* Browser is not opening in maximized mode by default
* At times we will have a requirement to open the browser to incognito tab

Solution:- For the above potential problems we have a solutions by adding chrome options . we can also set the desiredCapabilities for any browser such as Mozilla, IE< safri..etc

ChromeOptions op = **new** ChromeOptions();

// op.setHeadless(true);

op.addArguments("--disable-notifications");

// To run in private mode

op.addArguments("--incognito");

// To start browser in full screen

op.addArguments("--start-fullscreen");

// To disable the yellow info bar

op.addArguments("--disable-infobars");

ChromeDriver driver = **new** ChromeDriver(op);

driver.get("https://www.google.co.in/");

System.***out***.println(driver.getTitle());

**Un-anticipated challenges: -** These are a kind of un-checked exceptions which are truly due to the poor implementation. These can’t be expected due to the fact that we may not realize what it can cause until we come across, ideally these should have been avoided with due diligence and thorough review of the project, but there will be instance where people miss to do that. Given below were some examples of this category.

1.       improper coding standards

2.       least concerned about following framework standards

3.       improper reporting mechanism

4.       Improper feasibility analysis

5.       No proper exception handling

6.       Improper VCS system existence

**1.       Improper coding standards: -** If the coding standards are not maintained properly within the framework it can lead to great problems. Below are a few things which could eat our cat.

         Not using the best suitable data structure to perform data operations

         Hard coding the data in script

         Not writing generic methods

The above problems were some high level ones, which can increase the maintenance effort to our framework as redundant code needs to added, also it could cause significant delay to the script execution.

**2.       Least concerned about following framework standards: -** The purpose of framework is to build a structure to the project as where to write what? and how? With a proper template structure built, all team members have common understanding.

**A couple of problems seen are,**

* Not defining the folder structure
* Writing the common methods w/o following the interface document given
* Adding the dependencies to the build path directly rather fetching from the POM.xml/ build.gradle file

**3. Improper reporting mechanism:-** Reports are essential component to any test automation framework, based on reports the test case is mainly judged whether or not it’s a failed or passed one. It is mandatory to include the report test steps right after every validation to ensure we are capturing the action result to show it to the business stakeholders. Some common problems we may see interms of report generation on a few frameworks are

* No mention of the report step after validation, ideally these steps are more advised to add it to the common functionalities written
* Not mentioning the understandable description in the report, the problem with it is non-technical people may not decipher the results
* Not calling the report generation method under right testing annotation rather than writing them at the end of whole test execution as that results in generation of multiple reports
* No meaning full name to the report, that gets generated.

**4. Improper feasibility analysis:-** it is fact that 100% automation is impossible, below should be taken into consideration whilst taking a decision

* A careful study of the use case with the given scope is required to decide whether or not I can be automated.
* Understanding the value that it adds by automating the test case is must. Ideally we should be able to identify which way can it be validated quickly

**5. No proper exception handling: -** Exception handling is one important mechanism to understand the cause for the failure. If we fail to catch every exception that may be thrown by the test case during execution that leads to a problem of flakiness, test will execute till that step and hard stops then when we verify the report we will have all steps passed, but unfortunately not the whole test case is being run. This is one big problem at a business level, so documenting every exception with the help of try with multiple catch blocks is needed. It is equally important to add a understandable description in the report fail step for easy triaging

**6. No VCS exists in project: -** when we work as team, having a VCS system (GIT, Bit bucket, SVN)in place is must. Lack of this facilitation will cause problems as can’t properly merge the changes done by the different team members. It’s not ideal to just copy the changes from one system to another system when we have a system which does this for us very accurately. This will also help us to keep everybody in the team with the latest code base details.

**7. No Documentation:-** seldom times we come across certain issues, work around would have found after a thorough follow up with different teams. We must realize that the articulating those learning would help the team to a great extent. Continuous retrospection and noting the work around will service back the team and no need to spend efforts again when they re-occur. This is also acts as a pure knowledge base to the team.

**Eg**: any certificate installation to the java keystores /trust store to hit the application, verifying the hostname , any inputs from infra team during automation set up installation.