Automation and its best practices

As we all know that the importance of automation in this industry revolution is pinnacles. If we just hold breathe and asks ourselves a question as why? Due to its high accuracy of executing the redundant things, reporting results and it needs no or less human intervention hence the cost that a company spends on a resource is relatively reduced in a long run.

Whilst it’s true that the automation is so much beneficial in terms of bringing accuracy in and cost savings to the company, it is also true that if at all the planning and implementation is not taken utmost care it equally gives the negative results.

So, in this blog we are going to talk about the automation best practices, there aren’t any standard defined things as such, rather the below discussed things are the learning from various previous assignments. A proper retrospection after completion of each sprint and with the help of discovery learning we have come up with a set of best practices.

**Understand the application technology then identify tool**

Before we kick start with automation, there has to be a lot of ground work to be done from the application perspective. Thorough understanding of the application development infrastructure such as, what technologies are being used to develop front end, business logics and backend. This will help us identifying the tool that best fit for the application which in turn helpful for hassle free test automation implementation.

Eg:- if at all the application front end is developed with angular js then protractor is preferred to selenium, because it’s not possible for selenium to deal with some of the locators

Determine the types of automation tests

We need to identify what types of testing to be performed on the application then do a feasibility study to proceed with automation. When we say types of test we are dealing with two things

**Levels of testing**

This is a conventional test approach in almost all the projects, that first unit testing then integration testing followed by system testing. In this testing pyramid as we reach from the top to the base we intensify the test process to make sure there are no leaks, in the same fashion we need to ensure that each level of these tests are automated as well.

**Unit tests**

This is anyways done by the development team, these tests are automated most likely in the process of building and deployment process. Automated unit tests can give us some confidence that all components are running properly and can further be tested

**Integration tests**

It’s well known that Integration tests are performed after unit tests. In the recent agile model in order to conduct parallel testing alongside development activities, we need to consider automating the integration tests as well

These tests come under the classification API and involve with verifying the communication between specified micro services or controllers. UI might not have been implemented by the team we perform this.

**System testing**

System tests are designed when the system is available, in this level we deal with writing more number of UI tests which would deal with the core functionalities of the application. It adds great value when we consider these tests for automation.

**Types of tests that can be automated**

**UI layer tests**

User interface the most important piece of the application because that’s what exposed to the customer, more often than not business focuses on automating the UI tests. Alongside doing in sprint automation leveraging the regression test suit gives a great benefit.

**API layer tests**

We should be relying on UI tests alone, given their time consuming factor. API tests are par quicker than UI tests, automating tests in this layer will help us confirm the application stability without UI. This way we can conduct some early testing on the application to understand any major bugs earlier.

**Database tests**

Database test automation adds a great value to the team when there is a need to test new schema or during data migration or to test any quartz job that run in the DB table.

**Conduct feasibility study diligently to deem the ROI**

While brining automation in to the testing process is essential, it is also very much needed to do an analysis to understand the return on investment factor. Jumping on to automate everything is not a good approach. A proper understanding of the application then taking a decision on the tool and framework selection is recommended.

Failing to conduct feasibility study will lead to problems like, weak scripts with flakiness, more maintenance effort.

**Choose the best framework**

Choosing a suitable framework is part of feasibility study. Based on the agreed test approach such as TDD- test driven development or BDD- behavior driven development, framework selection and implementation should be taken into consideration

It’s recommended to go with BDD framework approach considering the current agile methodologies. As the spec file which can be understood by any non-technical professional drives the test case, its quite easy to follow and define the scope of any test case in the application.

**Separate tests and application pages code**

In test automation while designing the tests, it’s essential to write test cases and the application pages method. This kind of page factory implementation helps us to get rid of unnecessary maintenance. It’s expected that the application screens likely be changing over cycles, if at all the tests are not separated from pages we need to edit all the test cases even for a single change.

**Showcase meaningful reports**

The main purpose of test execution is to observe the results, be it manual testing or automated testing. Given the context, we need to integrate a reporting mechanism that should clearly state each test step with the corresponding results. When we share the reports to any business stakeholders they should find these meaningful and seem to capture all the correct information.

We have many third party reports being integrated to our test suite such as extent reports, allure reports and testing reports. The choice of reporting tool should be based on the framework implementation. The dashboards should be very clear. Again BDD frameworks such as cucumber , gauge would produce better html reports and these are in built.

**Follow distinguishable naming convention for test cases**

Not naming the tests properly is the silly mistake we ever do and its costlier. A good name given to the test case itself will speak for the functionality it checks for. This will also make the reports appear better as we are going to showcase the test case name and test case description in the report, it will also help the stakeholders to decipher the reports. Alongside naming the tests a meaningful name, it would be of great help if we can agree to a similar pattern for all test cases in test suit.

**Follow the best coding standards as recommended**

Despite the programming language when we are scripting the test cases its highly important to use the right methods and data structures. It’s not just recommended to find the solution but also to think of best approach that reduces the execution time. Peer review and review by a senior or team lead in the team before it has been pushed to VCS system.

Here are some best practices when we say coding standards

**Use try- catch blocks**

Whenever we are writing any generic method or any test method we must ensure its being surrounded by try with possible catch blocks and as a good practice we must print a useful, relevant message according to the exception that’s being caught.

**Use static variable to set some constant values**

In any automation framework we certainly have some variables which we don’t want to change them at all it’s always a best practice to have them declared as static ones. One good example of this is web driver object is made as static variable.

Considering the scope of variables, we must consider declaring them. Global variables are recommended as we can use them in different methods as the test case executes

**Generate random data and append to test data**

In the recent times applications have become more advanced when it comes to the point of dealing with the data. In a data driven test when we are fetching the data from an excel workbook, since the same test data is going to be used upon multiple runs, during second run if at all application doesn’t accept the duplicate data, we need to supply another set of data. Instead if we can just append a random number or an alphabet by using the java Random class, it serves the purpose

**Write generic methods**

While designing the test automation framework and designing the tests, it’s very important to understand the common functionalities and write them as generic methods. Anyone in the team who are developing a new script they need to make use of these methods as much as possible and any new method that doesn’t exist can be added with the discretion of senior automation developer or team lead. Failing to implement this process will add so much of maintenance to the test code which is tedious.

**Relevant comments to the code**

Comments are much needed to the any code snippet to help with understanding. It need not be a comment a good documentation such as,

What exception is being thrown by the method upon failure?

What parameters it takes?

What it returns back?

Documenting the above things would give crystal clear understanding to any newbie who joins the team and there won’t be any hassles to kick start with script development

**Using of properties files/ xml files in the code base**

Instead of hard coding or supplying a few of the parameters required for test such as browser info, driver exe path, environment URLs, database connection related passwords, hostnames it would be good practice to have them mentioned in a properties file or any xml file, the one advantage is that, these files need not be built every time we trigger the execution. It makes the process a light weight one also it is easy for maintenance.

**Use of proper wait handling**

On many occasions we see some of our tests will exhibit flakiness, its highly uncertain to find them why are they failing. If we ever conduct a proper triage we might end up answering ourselves that it has to do with waits. It’s better to go with implicit wait and wherever it’s required we can go with explicit wait so that we are not adding too much of execution time to the script. Unless there is a specific requirement its not recommended to use Thread.sleep

**VCS system availability**

When we as a team automating the application, its must to have a version controlling system in place, so that we can manage the code changes and merge without any conflicts or re-works if all goes good under proper guidance. Having VCS system is ideally recommended for any project that has automation test service.

**Bring CI/CD flavor to test automation**

DevsecOps is the buzzing thing in the industry in this era, this not just for development to build the code and deploy but also for testing to automate the execution process, having this infrastructure will help set the execution on a timely fashion or with a particular frequency. We can set the goals as what should be pre-execution goals and post execution goals. It can also help the team with sending the reports as an email, only job left is to analyze the results. This is one good thing to execute the regression tests with more frequency to ensure all functionalities work intact

**Capture screenshots upon failure and implement proper logging mechanism**

As the test runs we need to capture the screen shots and have them in a specific directory as evidence. We can take a call whether or not screen shots required for all passed steps or only for failed scenarios based on the memory constraints. The screenshot can further be used share with development team to understand the error when we report that as a bug.

Also a good framework must write log files to a directory, in order to see the steps have been executed. Dumping these into file would also help future reference.

Best practices are the things learnt from experiences, so adopting them would help us developing and maintaining the automation framework easily and that’s how we pave a tussle free path for our daily routines, some part of this discussion can be a debated as well, because in automation world we have many flavors of tools, applications and frameworks so probably there could have been different solutions. The best thing to remember here is, basis the need analyzing and finding solution then make it as a practice is recommended.

Hope there could have been some takeaways while, thanks a ton ☺