**Automation Standards**

# Framework should use best practices and java coding standards

Framework follows best practices and coding standards of java at all levels.

Example: Class level, method level, variables, constants etc.

Complete code is scanned through SonarQube and the suggestions are implemented in the framework.

# Framework should have advanced reporting

Framework uses advanced reporting and logging using Extent reports.

Additional code has be used to make this report look more effective and user-friendly.

It contains high-level data, graphs which is most useful for the management.

Detailed level data will be present which is useful for technical people.

On failure, screenshots are embedded into the reports.

# Framework should have the capability to run on multiple instances of browser in parallel and effectively.

Framework uses Threadlocal concept in java, which helps TestNG to perform parallel execution of test cases on multiple browser instances.

TestNG supports up to 10 parallel threads.

Framework allows test cases to run parallel at suite level or class level or methods level based on configuration provided in testNg.xml.

# Framework should be have configuration for which tests to run and this configuration should be in user-friendly manner.

Framework has the capability for configure which test cases / data set to run at below places:

* Excel file
* TestNG.xml
* Annotation in the code (ex: grouping test cases).

# Framework should be have configuration where we can specify on which browser tests to run and this configuration should be in user-friendly manner.

We can specify the browser name that we want the tests to run in config.properties file.

# No hard coding should be used in the framework

All the inputs are configurable.

Framework level parameters are configured at config.properties. Ex: Browser, UAT URL etc.

Test case level parameters are configured at Constants.java. Ex: Locator information etc.

Test case data is supplied from an excel file.

# Automation tests used in the framework should be resistant to changes in the UI as much as possible.

The generic automation tests that we planned to be implemented in the framework are designed in such a way that locator information does not change upon change in UI for most of the locators.

All the locator information is stored in Constants file.

# Framework should have the ability to be integrated to Browserstack, SauceLabs, and Appium.

Hooks are created as part of framework and can be integrated to Appium, browserstack, and Saucelabs.