Client Name

Project Title

DD Mmm YYYY

Version #.#

Type of Document

DocumenT approval

Synchrony IT

|  |  |  |
| --- | --- | --- |
| Name | Position / Title | Sign off Requirements |
| Gary Channer | VP, QA Center of Excellence, SYF-IT | Review and Sign Off |
| Vinay Gowda | Offshore Automation Stream Lead | Review and Sign Off |

Version History

|  |  |  |  |
| --- | --- | --- | --- |
| Version No. | Release Date | Released By | Description of Changes |
| 0.1 | 08.01.2019 | Ramya Kuthala | Draft Version |

Contact Information

If you have any questions regarding this document, please direct your e-mails to:

Gary Channer

Title: VP, QA Center of Excellence, SYF-IT

Company: Synchrony Financial

Email address: Gary.Channer@syf.com

Vinay Gowda

Title: Automation Stream Lead

Company: Attra Infotech Pvt ltd

Email address: [vinay.gowda@syf.com](mailto:vinay.gowda@syf.com)

Ramya Kuthala

Title: Senior Test Engineer

Company: Attra Infotech Pvt ltd

Email address: [ramya.kuthala@syf.com](mailto:ramya.kuthala@syf.com)

Version History of this template

| Version No. | Version Date | Name | Creator / Reviewer / Approver | Description |
| --- | --- | --- | --- | --- |
| 0.1 | 08.01.2019 | Draft Version | Ramya Kuthala | Initial Draft |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Table of Content

[1. INTRODUCTION 5](#_Toc15574372)

[1.1. Objectives 5](#_Toc15574373)

[2. TestNG-pom Framework – Design 6](#_Toc15574374)

[2.1. TestNG-POM Automation Framework – Architecture 6](#_Toc15574375)

[2.2 Folder Structure 7](#_Toc15574376)

[2.3 Framework Overview 9](#_Toc15574377)

[3. Execution Flow 11](#_Toc15574378)

[4. Reports 13](#_Toc15574379)

[4.1 HTML Reporting 13](#_Toc15574380)

[4.2 Detailed Execution Status Report 13](#_Toc15574381)

[4.3 ECD 13](#_Toc15574382)

[4.4 Log Report 14](#_Toc15574383)

[4.5 Extent Report 14](#_Toc15574384)

[4.6 Prerequisites for Generating Reports 14](#_Toc15574385)

# INTRODUCTION

Automation testing is an emerging field that draws maximum benefits with minimum effort. Then benefit of automation testing is its ability to increase the efficiency of resources, increase test coverage, and increase the quality and reliability of the software.

While there are several frameworks that provide support for automated software testing, this document introduces one particularly effective type: Selenium TestNG-POM Test Automation Framework.

## Objectives

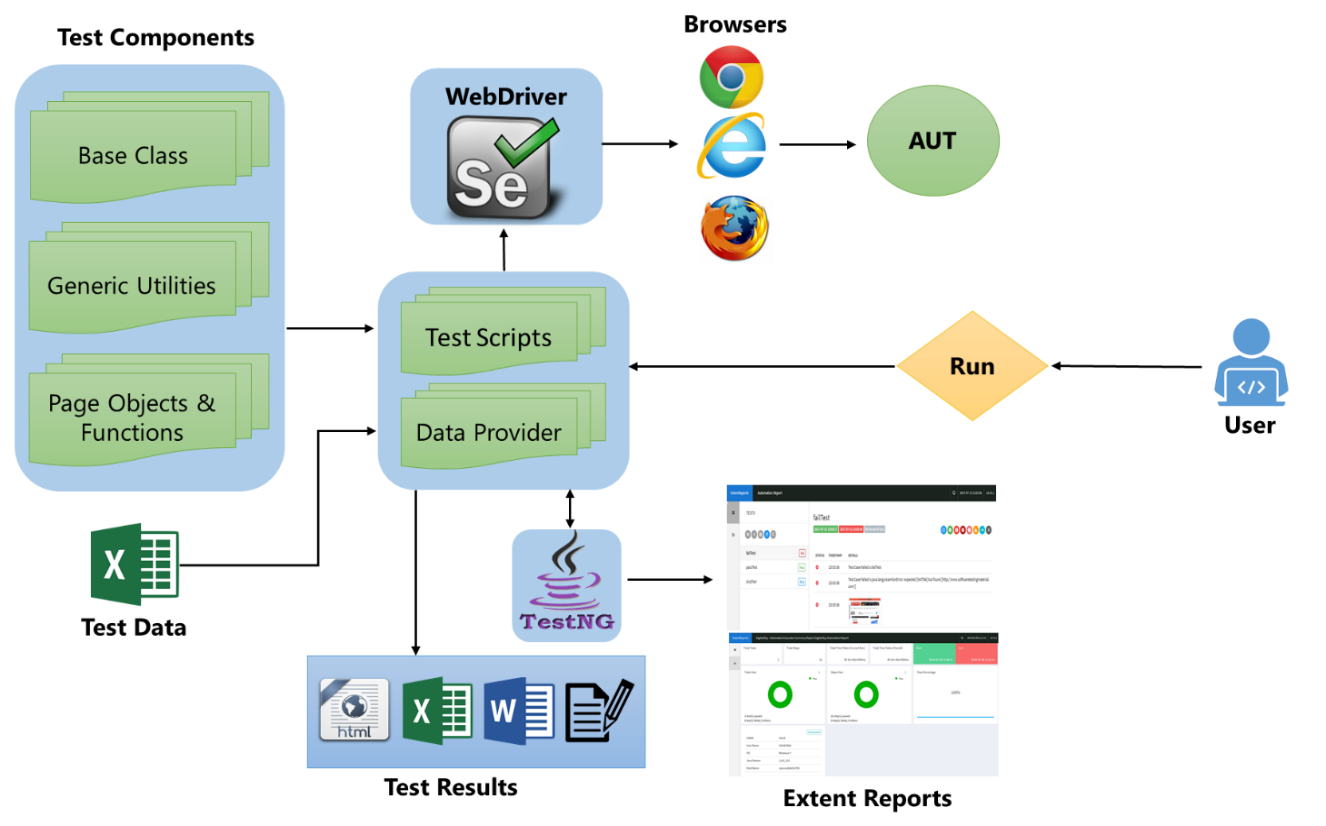
Strategic objectives in building the below framework would be addressed as detailed below

* Implement a strategy that will allow tests to be developed and executed both manually (initial test cycle) and via an automation framework (regression test cycles).
* Separate test design and test implementation to allow test designers to concentrate on developing test requirements, test planning, and test case design while test implementers build and execute test scripts.
* Employ a test strategy that assures that test cases include the navigation and execution steps to perform, the input data to use, and the expected results all in one row or record of the input data source.
* Realize an integrated approach that applies the best features of data-driven testing, and functional decomposition testing.
* Implement an application-independent test automation framework.
* Document and publish the framework.
* Develop automated build validation (smoke) tests for each release of the application.
* Develop automated environmental setup utility scripts for each release of the application.
* Develop automated regression tests for
  + GUI objects and events Application functions
  + Application special feature
  + Application performance and scalability
  + Application reliability
  + Application compatibility
  + Application performance

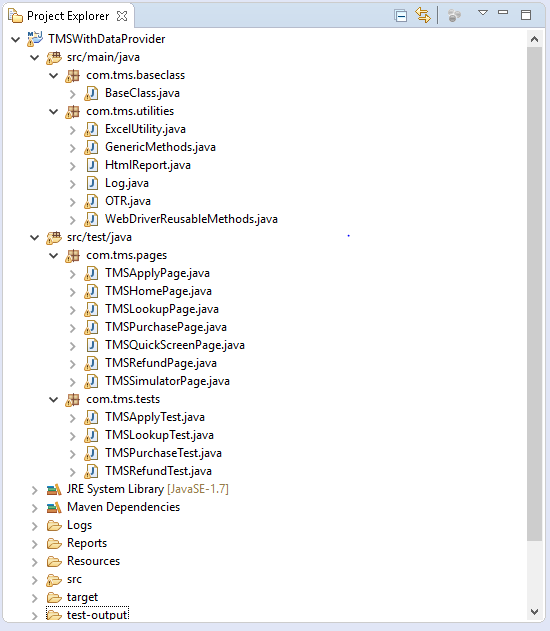
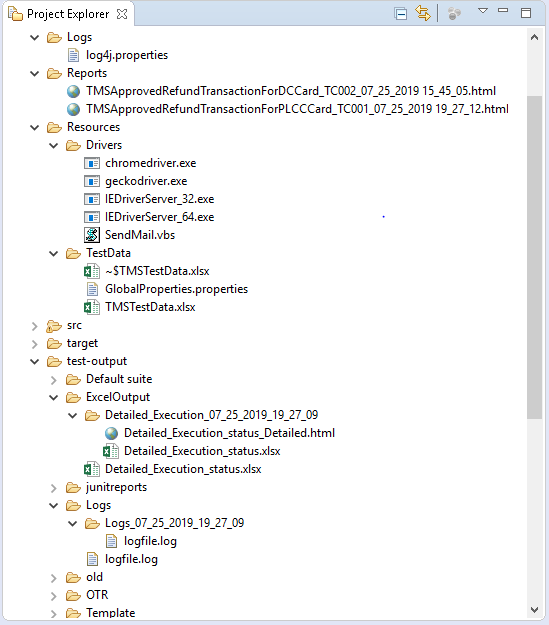
# TestNG-pom Framework – Design

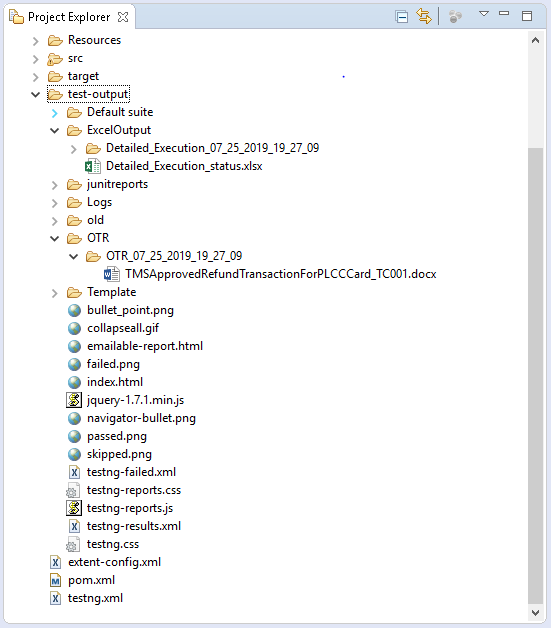
Below mentioned diagram depicts the automation test execution workflow designed using TestNG-POM Automation Framework.

## TestNG-POM Automation Framework – Architecture



## Folder Structure



## Framework Overview

**Type of Framework:** This framework is a TestNG [data-driven framework](https://www.softwaretestingmaterial.com/data-driven-framework-selenium-webdriver/) using [Page Object Model design pattern](https://www.softwaretestingmaterial.com/page-object-model/) with Page Factory.

**POM:** As per the Page Object Model, separate class for every web page is maintained. Each web page has a separate class and that class holds the functionality and members of that web page. Separate classes for every individual test are maintained.

*Example:* TMS Home Page and TMS Refund Page have a separate class to store element locators. For the TMS Refund test there would be a separate class which calls the methods from the TMS Home Page class and TMS Refund Page class.

**Packages:** Separate packages are maintained for Pages and Tests. All the web page related classes come under **pages** package and all the tests related classes come under **tests** package.

All the tests and element locators (POM classes) are kept in **‘src/test/java’** folder and remaining files like base class and utility classes are kept in **‘src/main/java’**. Test data sheet, properties file and the driver executables are kept in **‘Resources’** folder.

**Base Class:** Base class (BaseClass.java) deals with the common functions used by all the pages. This class is responsible for loading the configurations from properties files, Initializing the WebDriver, Implicit Waits and generating Extent Reports.

**Utility Classes:** Utility class stores and handles the functions (The code which is repetitive in nature such as waits, actions, capturing screenshots, accessing excels, sending email etc.,) which can be commonly used across the entire framework. The reason behind creating utility classes is to achieve reusability. These classes extend the Base class to inherit the properties of BaseClass in utility classes.

Following Utility Classes have been created as part of this framework.

**Excel Utility (Excel Utility.java):** All excel related methods like getting values from excel and writing data to excel are kept in this class. This class is also responsible for creating the object of FileInputStream which is responsible for pointing towards the file from which the data should be read.

**Generic Methods (GenericMethods.java):** All methods related to property file, screen capture and browser screen capture are kept in this class.

**WebDriver Reusable Methods (WebDriverReusableMethods.java):** This class deals with the functions to launch the browser and to perform actions on web elements like clicking a button or setting value in a text box etc.

**OTR(OTR.java):** Functions for creating, saving and updating the OTR document is kept in this class.

**Logs (Logs.java):** Functions for generating log reports are kept in this class.

**HTML Report (HtmlReport.java):** Functions for generating HTML report are kept in this class.

**Properties file:** This file (GlobalProperties.properties) stores the information that remains static throughout the framework such as browser specific information, application URL, screenshots path etc.

All the details which change as per the environment and authorization such as URL, Login Credentials are kept in the GlobalProperties.properties file. Keeping these details in a separate file makes easy to maintain.

**Test Data:** All the test data will be kept in excel sheet (TMSTestData.xlsx). Data driven testing is handled using TestNG’s Data Provider. Data Provider loads data from the excel sheet and passes it to the test. We use [Apache POI](https://www.softwaretestingmaterial.com/handling-excel-files-using-apache-poi/) to handle excel sheets.

Data Provider helps to write data-driven tests which essentially means that same test method can be run multiple times with different data-sets.

**TestNG:** TestNG is used for Parameterization, Assertions and Grouping.

**Maven:** Maven is used for build, execution and dependency purpose.

**Reports:**

**Test Results:** For reporting purpose, following customized reports will be generated by the script.

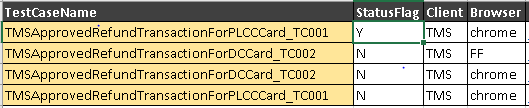
* HTML Reporting
* Detailed Execution Status Report in Excel
* OTR (ECD)
* Logs

**Extent Reports:**  Extent Reports will be generated by TestNG for reporting. It generates beautiful HTML reports. We use the extent reports for maintaining logs and to include the screenshots of failed test cases in the Extent Report.

# Execution Flow

To execute any test, status flag of the test should be set to “**Y**” and enabled attribute of @Test annotation should be “**true**”.

Status Flag for the test in test data sheet should be “**Y**”.



Enabled attribute of @Test annotation of the test method should be “**true**”.



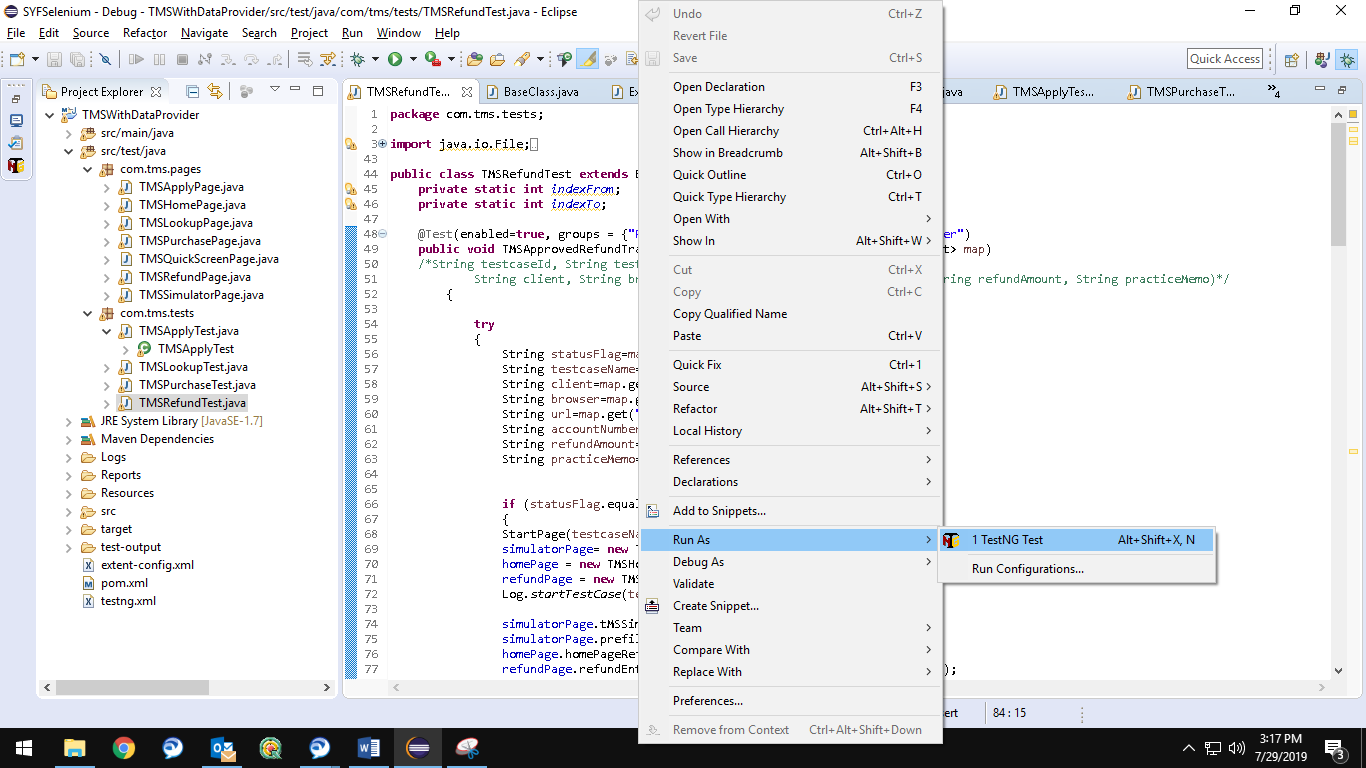
User will trigger the execution from the test script and the test script will load all the test components and test objects. The data will be loaded from the excel sheet using data provider and then the test script will initiate the web driver to launch the web browser and the application under test.

Once the execution is completed the test results will be captured by the script and different reports will be generated.

Test execution can be triggered in the following ways.

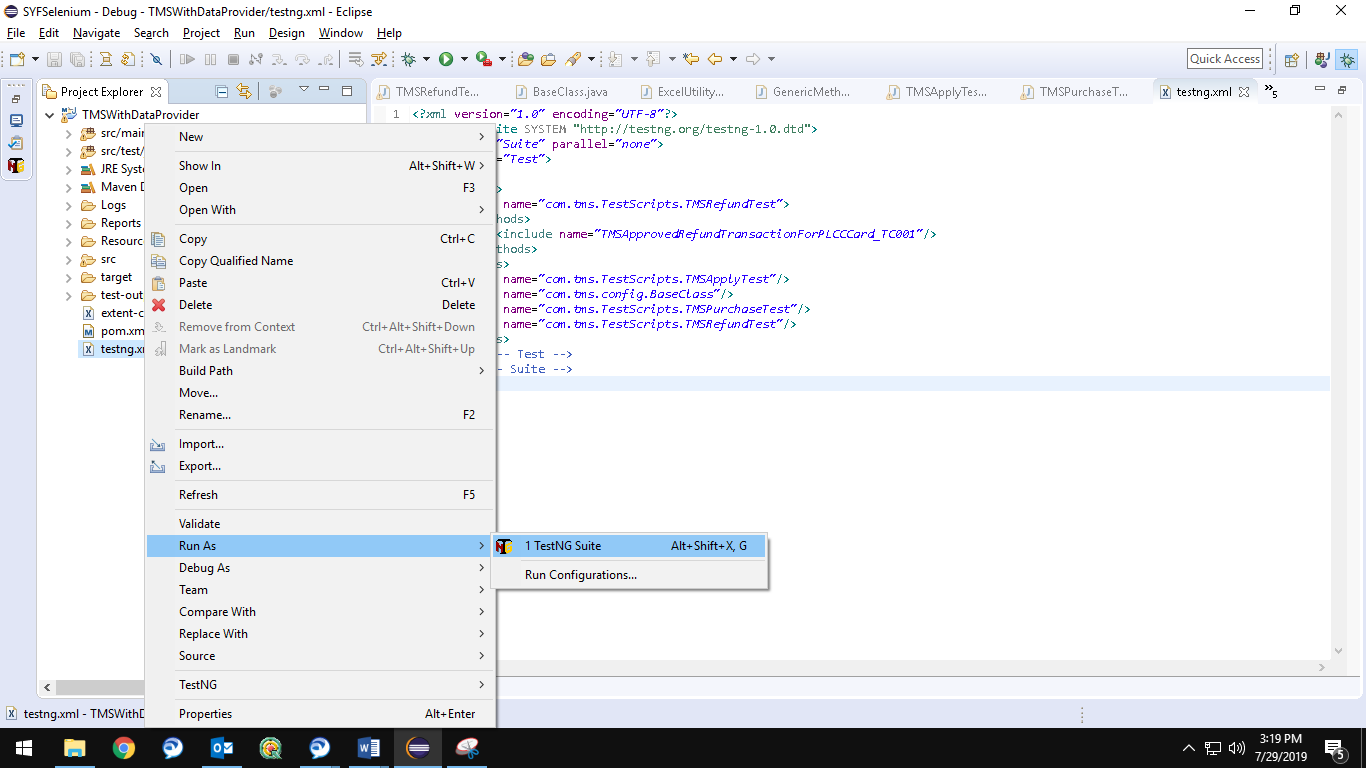
1. **From Test Script**

Right Click on Test ->Run As->TestNG Test



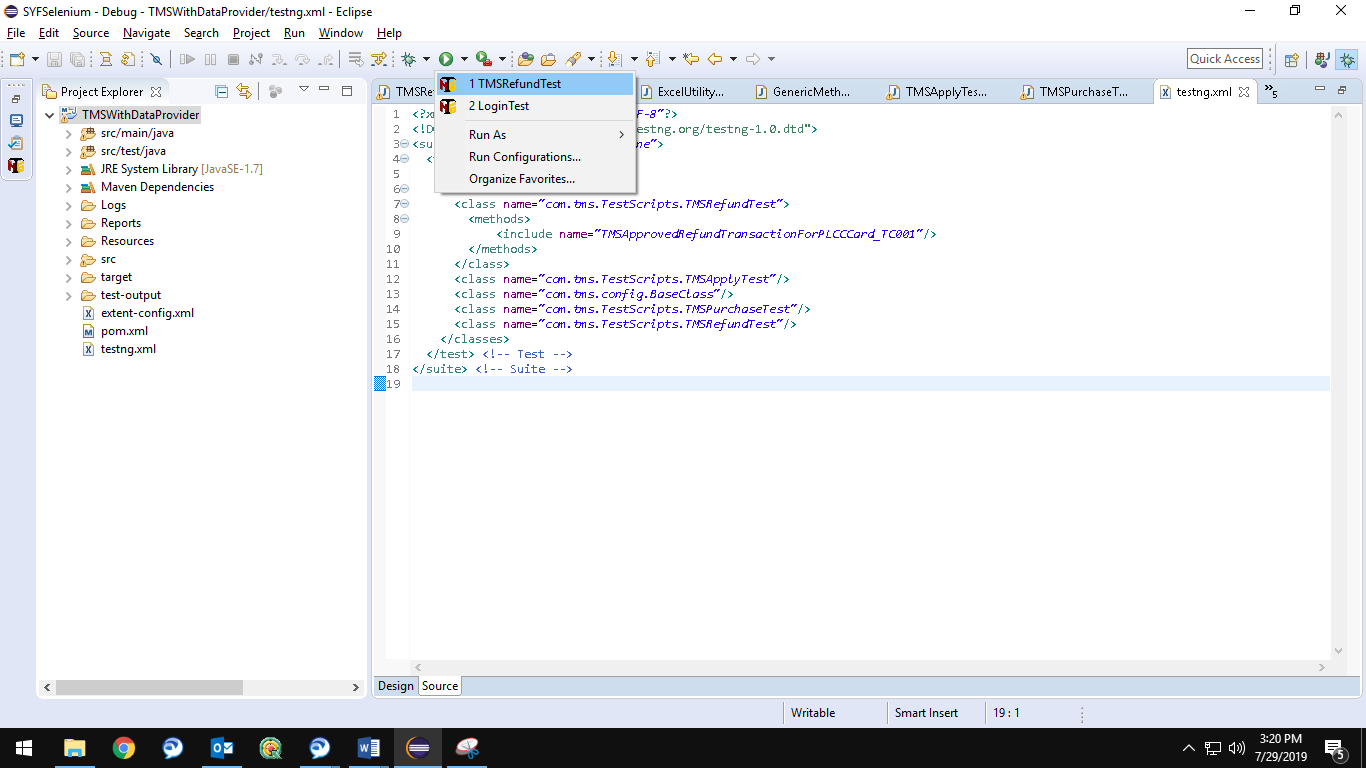
1. **From testing.xml**

Right Click on testing.xml->Run As->TestNG Suite



1. **Run from Menu Bar**

Click on Run Button on Menu Bar



# Reports

Following customized reports will be generated by the framework

### HTML Reporting

* HTML report is a customised report which contains test name, ECD’s path and the execution status along with the comments if applicable. Below is the sample report for the same.



### Detailed Execution Status Report

* Detailed Execution Status is a customised excel report which provides the summary of the test run, below is the sample report for the same.



### ECD

* Evidence capture document is a word document which captures step by step details of the automation execution with screen shots. Below is the sample report for the same.



### Log Report

* Log report is a log file which contains brief description of the actions performed step by step on the AUT. Below is the sample report for the same.



### Extent Report

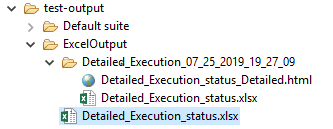
* Extent report is a report generated by TestNG. It is generated in html format and it contains system information and detailed status of every test step along with the screenshots. Below is the sample report for the same.



### Prerequisites for Generating Reports

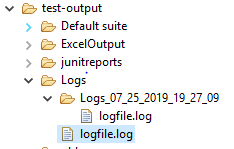
The following images and templates should be placed in the respective folders under test-output before running the test for the reports to be generated.

* Below Detailed\_Execution\_Status.xlsx template should be placed in ExcelOutput under test-output folder. Based on this template, detailed execution status excel report will be generated for each run.



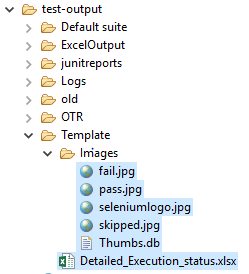


* Logfile.log template should be placed in Logs under test-output folder. Log file will be generated based on this template.





* Logos for Selenium, pass, fail and skipped test cases should be kept in Images under test-output folder. Html reports use these images to create the report.





*Example:*

