Icon

Description automatically generated

**Web Lite Automation Framework – Architecture Document**

version 0.1

20 October 2022

Table of Contents

[1.0 What is a Test Framework? 3](#_Toc117192882)

[2.0 Vasquez Web Lite Automation Framework 3](#_Toc117192883)

[2.1 What is POM? 3](#_Toc117192884)

[2.2 Why POM is required? 3](#_Toc117192885)

[2.3 Advantages of POM model: 3](#_Toc117192886)

[2.4 Benefits of POM: 4](#_Toc117192887)

[3.0 Architecture and Components of the framework 4](#_Toc117192888)

[3.1 Page Objects 4](#_Toc117192889)

[3.2 Test Classes 5](#_Toc117192890)

[3.3 Utilities 6](#_Toc117192891)

[3.3.1 Data Handlers 6](#_Toc117192892)

[3.3.2 Extent Reports 6](#_Toc117192893)

[3.3.3 Listeners 7](#_Toc117192894)

[3.3.4 Logs 7](#_Toc117192895)

[3.3.5 Generic Utilities 7](#_Toc117192896)

[3.4 Resources 7](#_Toc117192897)

[3.4.1 Browser Drivers 7](#_Toc117192898)

[3.4.2 Configuration Properties 8](#_Toc117192899)

[3.4.3 Test Data 8](#_Toc117192900)

[3.4.4 TestNG XMLs 8](#_Toc117192901)

[3.5 Test-Output 8](#_Toc117192902)

[3.6 POM.xml 8](#_Toc117192903)

# 1.0 What is a Test Framework?

A Testing framework is an essential part of any successful automated testing process. It is a set of guidelines or rules used for creating and designing test cases. A framework is comprised of a combination of practices and tools that are designed to help QA professionals test more efficiently.

These guidelines could include coding standards, test-data handling methods, object repositories, processes for storing test results, or information on how to access external resources.

# 2.0 Vasquez Web Lite Automation Framework

The Lite version of the automation framework that we use to test the web applications in Vasquez project uses a Java based hybrid automation framework using Page Object Model, Maven & TestNG that inculcates the benefits of both the Data-Driven and Keyword-Driven worlds.

## 2.1 What is POM?

**Page Object Model** is one of the design patterns which is used in test automation and has become most popular because of avoiding the duplication of code, supporting change of requirements and helps in maintenance of test framework. The Page objects that are created for each page in the tests acts as an interface to the respective pages present in the Application under test.

These page objects use the respective methods written for each page and interact with the UI to perform the actions. One of the best advantages of this framework is even though the UI in the AUT changes for any pages, the tests need not be changed, only the methods present in the respective page objects has to be modified. This reduces the maintenance of the framework when any change in requirements or UI happens.

## 2.2 Why POM is required?

Page object model is most popular automation frameworks because of the below reasons:

* POM provides a capability of managing the locators in such a way that if any changes to the UI is done, it is very easy to update or maintain the framework.
* POM framework is easily readable to any QA Engineer that it is self-explanatory on how the framework is structured.
* It is very easy to locate any methods related to any functionality or modules because all these methods are designed and structured on Page object basis. Number of pages is equal to number of page objects.

## 2.3 Advantages of POM model:

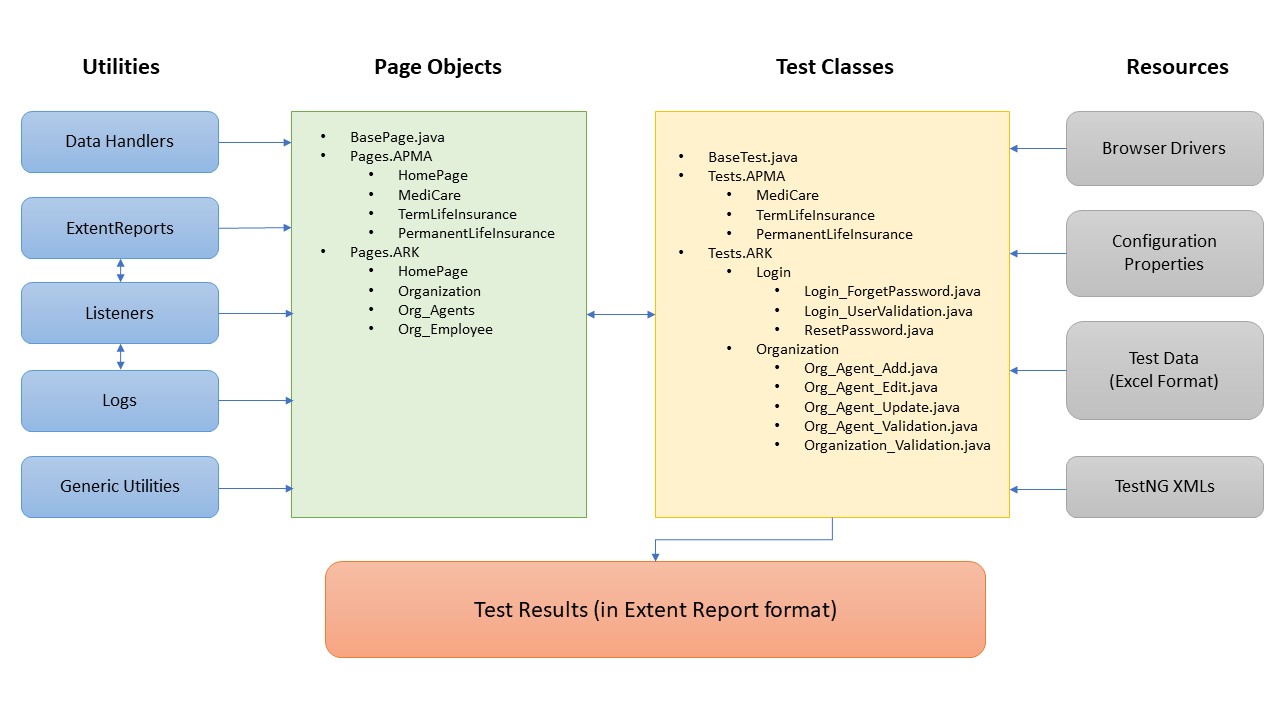
* **Reusability:** We can reuse the page class if required in different test cases which means we don’t need to write code for identifying the web elements and methods to interact with them for every test case.
* **Maintainability:** As we can see from the above picture test case and page class are different from each other which means we can easily update the code if any new web element is added or existing one updated.
* **Readability:** As we can see in the above picture page code is separated form test code which helps to improve code readability.

## 2.4 Benefits of POM:

* POM is a comprehensive framework which is more easily readable and understandable framework. This user-friendly nature has made POM popular among other frameworks in the automation world.
* POM helps us to keep the code clean and reduces the duplication of code.
* POM helps us to create compact test cases as the entire logic of validating the modules and locating the elements lies within the Page objects.

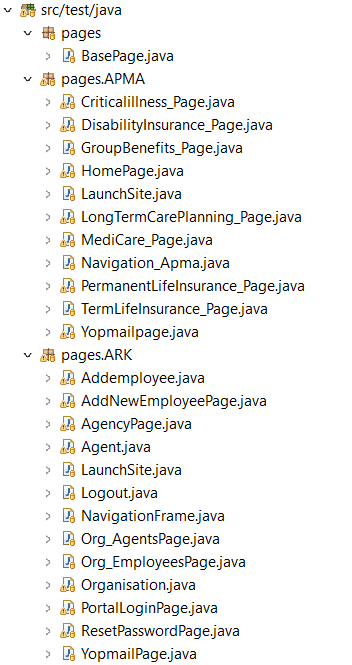
# 3.0 Architecture and Components of the framework

The architecture diagram is given below,



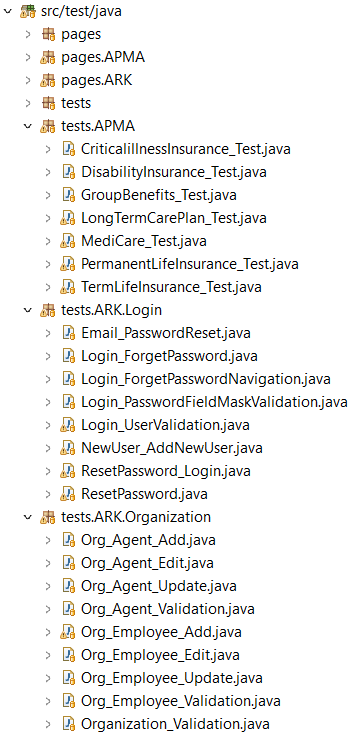
## 3.1 Page Objects

Page Objects classes are those that encompasses the object locators and the methods that uses these locators together. Usually, the page classes are created per page and hence we need to have one class file for each page, for example, one for login page, one for home page, one for each of the plan pages and so on. Furthermore, we can group the related pages into packages and have them structured. We have packages defined at application level wherein we have one for APMA and another for ARK. Please find the page packages and related details below,



## 3.2 Test Classes

Test Classes are those files that has the actual test case steps written in the Keyword format. Each test class should have the TestNG @BeforeMethod, @AfterMethod and @Test annotations. The Before Method (one per class file) is used to setup the web driver instances and create the page instances that are to be used by the test cases. The After method (one per class file) should have the tear down process of save the test data file and quitting/closing the web driver instance to free up the space. The Test method (multiple test methods can be placed in one class file) are created in one-to-one fashion with the manual test case. The test method should simply orchestrate the test based on the test steps by calling the different page methods. We have packages defined at application and module level based on the volume of test cases. Please find the page packages and related details below,



## 3.3 Utilities

The Utilities package will have multiple sub packages, that cater to specific needs of the framework. The list of packages are as follows,

### 3.3.1 Data Handlers

This sub package contains the class files that handles the Configuration properties as well as the test data. The current version of the framework has the test data handled in Excel format and hence the excel handling classes can be found here. Future plan is to incorporate JSON related classes to provide support for JSON test data

### 3.3.2 Extent Reports

This sub package contains the class files that handles the Extent reports. In this framework we use Extent report as the main test result report in addition to the default TestNG report.

### 3.3.3 Listeners

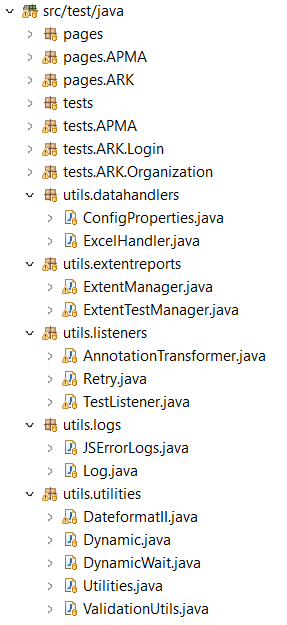
This sub package handles the class files that are dedicated to listeners. We use the iTestListener for handling the TestNG file and orchestrating the test execution. Other listeners that are used are iRetryAnalyzer and AnnotationTransformer which are used for automatic rerun of failed test cases.

### 3.3.4 Logs

This sub package contains the class files that handles the JavaScript Error Logs and Logger logs which will be captured for the entirety of the test execution

### 3.3.5 Generic Utilities

This sub package comprises of class files that can be used for common methods like click, wait till element is displayed, select a radio button, select a drop-down list, data formatter, etc as well as common navigation related methods



## 3.4 Resources

The resources folder is used to collate all the external resources like test data, test case set, configuration properties and browser drivers that are required by the page and test classes for test execution. The sub packages are listed below,

### 3.4.1 Browser Drivers

This folder contains the browser driver executables for the different browsers likes Chrome, Edge, Safari and Firefox (geckodriver)

### 3.4.2 Configuration Properties

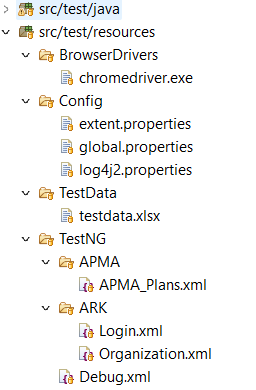
This folder contains the common configuration properties for the test run as well as log4j and extent report related properties

### 3.4.3 Test Data

This folder contains the test data in excel format

### 3.4.4 TestNG XMLs

This folder contains the TestNG XML files and can be further divided into sub folders based on the application and its modules.



## 3.5 Test-Output

The Test-Output folder contains both the extent report file as well as the TestNG report file. This folder will not be included as part of the framework and will be created automatically when the execution is triggered

## 3.6 POM.xml

The POM Maven repository file contains the details of the project like the GroupID, ArtifactID, Version and common project properties, It also have a list of dependencies whose libraries will be downloaded at run time and gets installed. It also uses the maven-surefire plugin to assist the test case execution.