**Asperato ONE - Automation via Selenium WebDriver**

Version 1.0

Document Information

Information

|  |  |
| --- | --- |
| Document Name | Automation via Selenium WebDriver |
| Author |  |
| Owner while current |  |
| Owner’s Contact No |  |

Change History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Change Owner | Change Description |
| 1.0 | 06/04/2020 | Juhhi Modi | Template Creation |
| 1.1 |  | Anshita Baheti |  |

Lead Sign-Off

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Title | Company | Date | Approval Status |
| Mukesh Sharma | Project Delivery Manager | Metacube |  |  |
|  |  |  |  |  |

**Table of Contents**

[**Introduction About Document**](#_yzdvs74dkh0x) **4**

[**Introduction about Project (Asperato)**](#_8ripfi78gs0n) **4**

[**How Much Test Cases Can Be Automated (Developing proof of concept on automation)**](#_d7kujzydpsf6) **6**

[**Introduction of Selenium (Automation tool)**](#_h3s5c3mks1l3) **9**

[**How it is good approach to go ahead**](#_kg4ste4iupcv) **11**

[**Required Software Installation**](#_mo5g7ykzg16) **11**

[**Planning the test approach**](#_aszacf8tr9uk) **12**

[**Design Test Automation framework**](#_7and7oox9o0q) **14**

[**Developing the Execution Plan**](#_a1ouqa838k5i) **17**

[**Writing Scripts**](#_f9brnamk0zcn) **17**

[**Test Reports**](#_4pbhafpp02t5) **18**

[**Divide the Test Cases as per Testing type**](#_1hkmgfhbuk0g) **19**

# Introduction About Document

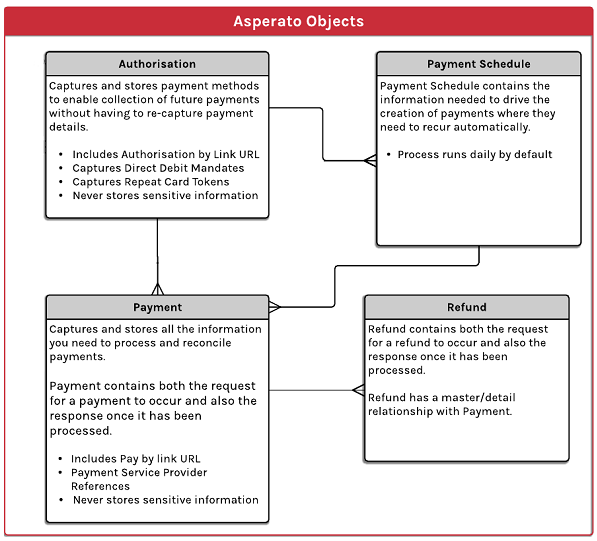
In this document we shall review the benefits of using automation testing to compliment manual testing, along with comparisons of 3 options to conduct automation testing:

1. Building selenium scripts using an offshore resource
2. Using the tool Provar
3. Using the tool SNAP provided by [www.transitusnexgen.com](http://www.transitusnexgen.com)

# Introduction about Project (Asperato)

Asperato ONE allows Salesforce to control the collection of payments. It enables you to collect payments from many different providers such as Worldpay, GoCardless, Braintree, and more. It allows you to fully automate all the payment collection activities from the single control which are -

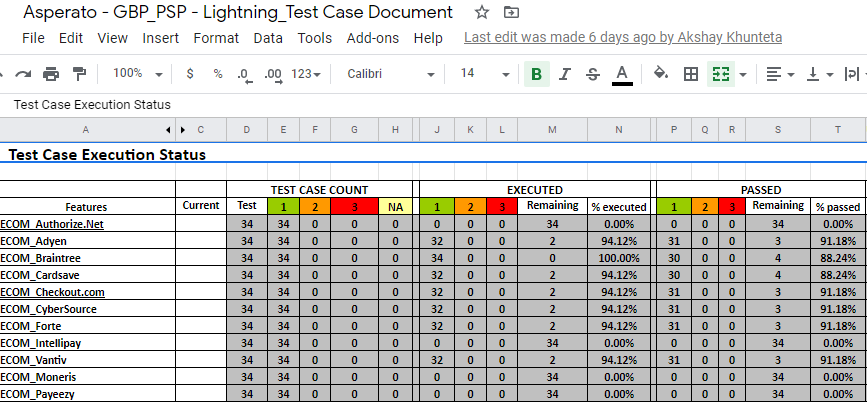
1. **Setup** - Different payment service providers need to set up the connection. It defines payment can be taken in which mode and which currencies. Each currency can have one service provider for any mode.
2. **Authorisation** - It is used to save the payment methods (like debit cards, credit card) for future use. Single customers can have multiple authorisation records.
3. **Payment** - Used to maintain the payment record. A new payment record should be created first, for the collection of payment.
4. **Refund** - Multiple or single refund for a payment can be allowed depending upon the service provided by the payment service provider. Refund is only applicable after the payment has been processed successfully.
5. **Payment Schedule** - Its function is to create Payment records, linked to Authorisations (Saved Payment methods) on a fixed schedule so that these payments can be collected automatically.



# How Much Test Cases Can Be Automated (Developing proof of concept on automation)

Test automation allows the testing team to execute test cases more frequently which actually increases the coverage of tests. During Sprint or new development of any particular module does not allow much time to execute affected areas (regression testing) so these types of tests can be covered easily via automating the testing script.

After designing the test cases for Asperato application related to different currencies and modes, the test summary looks like below -



Here it describes -

* One currency can be divided into multiple PSP’s connections along with Payment Modes
* Unique test suite can be created for each connection of each Currency
* Each module has one test case for ***Creating Connection***and remaining for ***Payment & Authorisation***

So this is for one currency, we did the same for all currencies and its related modes & connections.It means we executed 34 test cases for each connection*, so execution of the same test cases for other connections can be included as part of Automation Plan.*

**So on the basis of it, here are few points will determine if it is required at all:**

**#1.** **How much part can be automated?**

Now as per above image, we can see after executing the test cycle for one connection, we could automate the same test cases and run for other PSP’s. It could save overall testing time and ensure complete coverage.

**#2 Stability of the application?**

So far it seems we are not gonna change majorly with the current development so application should be considered for automation

**#3 Is Manual testing done for the Project?**

Since we are done with the manual testing for the given requirements and we are here with less bugs so in this case, automation can be thought of.

**#4.** **Does the organization have a budget for automation?**

Since we accessed some Record and Play tools we concluded that there is no tool which is very effective and in less budget. So here we explained below too how Selenium is good tool to go

**Here is the list of test cases that were part of manual testing of Asperato One -**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case Documents** | **Test Case Executed** | **Test Case Passed** | **Test Case Failed** |
| [Asperato generic test cases on lightening platform](https://docs.google.com/spreadsheets/d/1G4qwsFaVzJxUglZINB6NkeGjOjZOqhDCZ2EfRfju17s/edit#gid=1292776730) | 154 | 138 | 16 |
| [Asperato - USD\_PSP - Lightning\_Test Case Document](https://docs.google.com/spreadsheets/d/1ltRDq8j0wv-b6un4DT5W8iHoEIOOgP3DVdajv-SjFYw/edit) | 256 | 241 | 15 |
| [Asperato - GBP\_PSP - Lightning\_Test Case Document](https://docs.google.com/spreadsheets/d/1Ad_VtgGRN267icQrxPe8Fibf8hNM2zUEVLgrquEUCqg/edit) | 358 | 339 | 19 |
| [Asperato - EUR\_PSP - Lightning\_Test Case Document](https://docs.google.com/spreadsheets/d/18ZIcMDi941Kl9Lnqe02v7OZNB6z5H-vpzlUCTckHI2M/edit#gid=1455144840) | 32 | 30 | 2 |
| [Asperato - JPY\_PSP - Lightning\_Test Case Document](https://docs.google.com/spreadsheets/d/10DzV7PDK41bQnXzvwxBTS2phMGkTfDiE-aJN09KDLEY/edit) | 32 | 31 | 1 |
| [Asperato generic test cases on classic platform](https://docs.google.com/spreadsheets/d/1vMhj8A3JaKGdVPdCSloN4yuvO2EeUWH-s1dZ19Lq68E/edit#gid=156155796) | 146 | 132 | 14 |

*So far we do not see any blocker module/section that can not be automated apart from the installation section.*

**Here is the proof of concept on automation -**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Steps** | **Sub Steps** | | | **Result** |
| **Open Chrome browser Or Any Test Browser** |  |  |  | Chrome / test browser Launched |
|  |  |  |  |  |
| **Launch and Login into the App** | Launch application |  |  | Successfully logged into the application |
| Login into the app |  |  |
| Skip mobile Number Page |  |  |
|  |  |  |  |  |
| **Select Asperato app** | Click on hamburger or dotted icon |  |  | Asperato App is launched |
| Select Asperato app via using Search box |  |  |
|  |  |  |  |  |
| **Click on Setup tab** | Add merchant group |  |  | New connection got added into Test Merchant Group |
| Add connection for any (Currency + Mode + Connection) |  |  |
|  |  |  |  |  |
| **Click on Payment tab** | Add payment record |  |  | Payment record got added and Payment got processed successfully and its status of “Collected from Customer” |
| Process payment |  |  |
| Fill data on Asperato Payment Solved gui |  |  |
| Refresh the payment screen |  |  |
| Verify the status |  |  |
|  |  |  |  |  |
| **Click on Refund button** | Enter data and submit |  |  | Refund get processed successfully |
| Verify status |  |  |
|  |  |  |  |  |
| **Click on Authorisation tab** | Add Authorisation record | Add Authorisation record | Add Authorisation record | Authorisation record got added and Authorisation got processed successfully and its status of “InForce”.  Authorisation happened from eCommerce url successfully also and able to do payment successfully |
| Process Authorisation | Click on eCommerce link | Payment using Authorisation |
| Fill data on Asperato Payment Solved gui | Fill data on Asperato Payment Solved gui | Fill data on Asperato Payment Solved gui |
| Refresh the Authorisation screen | Refresh the Authorisation screen | Refresh the Authorisation screen |
| Verify the status | Verify the status | Verify the status |
|  |  |  |  |  |
| **Click on Payment Schedule tab** | Add PS record with InForce Authorisation record |  |  | Payment Schedule record got created successfully and its related Payment record should have “Collected from customer” status |
| Wait for the next day if freq is daily |  |  |
| Open PS record |  |  |
| Verify the linked payment record |  |  |
|  |  |  |  |
| **Close application** | Close salesforce app |  |  | Close the application |

# Introduction of Selenium (Automation tool)

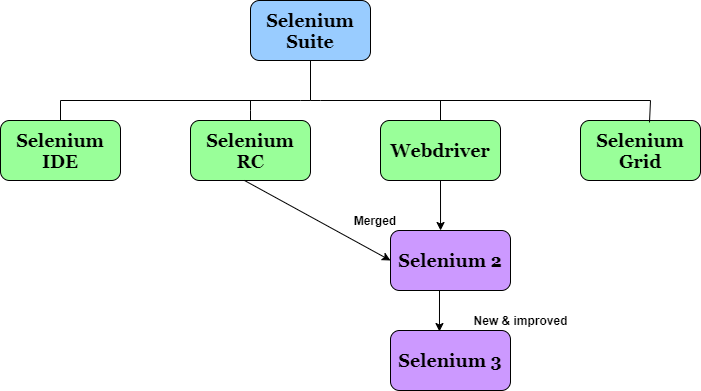
Selenium is a free open source automation testing tool which is used for automating web applications on *multiple browsers and multiple operating systems (Platforms).* It is used for both Functional and Regression Testings.

1. License free
2. Multi browser support
3. Multi language and framework support
4. Ease of implementation
5. Parallel test execution
6. Open source availability for most of tool required for it

**Components of Selenium -**

Selenium has 4 major components -

1. Selenium IDE
2. Selenium RC
3. Selenium Webdriver
4. Selenium Grid



Brief description for the above mentioned components -

|  |  |  |  |
| --- | --- | --- | --- |
| **Selenium IDE** | **Selenium RC** | **Selenium Webdriver** | **Selenium Grid** |
| * Simple record and playback tool * Works with Firefox Mozilla only * Test case can be exported in different languages like Java, Ruby, C# * Single test suite run a time | * Need to write the code, manual process called Selenium Serverthat is mandatory to start before execution * Works with different browsers which supports Javascript * It supports multiple programming languages (JAVA, Ruby, Perl, PHP, Python, and C#) * Does not support parallel testing | * Need to write the code, direct communication between code and browser * Supports multiple platforms like MAC, windows, different browsers like Mozilla Firefox, Chrome, Internet Explorer, Safari etc * It uses different languages like C#, JAVA, Ruby, Perl, Python and PHP * It allows parallel testing on different browsers at a single time | * Need to write code and it supports parallel testing * Execute test scripts on different machines at the same time * It is platform independent * It is browser independent * It is language independent |

*Note - Since most of the applications are Ajax-based so we would go via* ***Selenium WebDriver.***

*Along with this, we would use* ***Selenium Grid which*** *is used with Selenium RC to run parallel tests across different machines and different browsers all at the same time.*

# How it is good approach to go ahead

*Selenium webdriver supports multiple browsers and different languages. The tool is cost effective because it is an open source tool. Easy installation and it can be data driven also.*

**Below this is comparison of Selenium with Provar and SnapTest-**

|  |  |  |  |
| --- | --- | --- | --- |
| **Features** | **Selenium Webdriver** | **Provar** | **SnapTest** |
| **License** | It is free open source, so there will be less maintenance cost and testing cost | It is a commercial tool, requires a license and is expensive. No trial | It is a commercial tool, requires a license and is expensive. Only 7 days free trail version is available |
| **Operating System** | Selenium supports WIndow 7, Windows 10, Linux, MAC etc |  |  |
| **Programming Language** | Different languages - Java, C#, Python, Ruby, Perl, PHP, .net etc. | Java | JavaScript |
| **Browser Compatibility** | Supports almost all types of browsers - Chrome, IE11, Firefox, Safari, Opera, HTMLUnit, Android, Phantom JS etc | Browser options for provar in Chrome and Firefox | **SnapTest is** a chrome extension, but we are considering making a port for **Firefox** |
| **Application Support** | We can test only web based applications | Chrome Extension, No desktop app | Chrome extension, No desktop app |

# 

# 

# Required Software Installation

For Selenium web driver, below is the following list of softwares required -

1. JAVA (JDK) - Java development kit
2. Eclipse IDE - Integrated development environment
3. Selenium Java Client Driver
4. Different browsers driver for eg -
   1. Firefox - Mozilla Gecko Driver
   2. Chrome - ChromeDriver
   3. Safari - SafariDriver
   4. IE - InternetExplorer Driver
5. External JAR files as per the requirement
6. TestNG - Testing framework for Java Programming language
7. Maven - Build automation tool
8. GitHub - Version control software development system
9. Jenkins - Automation server (CI/CD)

# Planning the test approach

We will be dividing the test cases into Phases where

***Phase 0*** *- Covers Priority 1 Test cases as per Test Case Document*

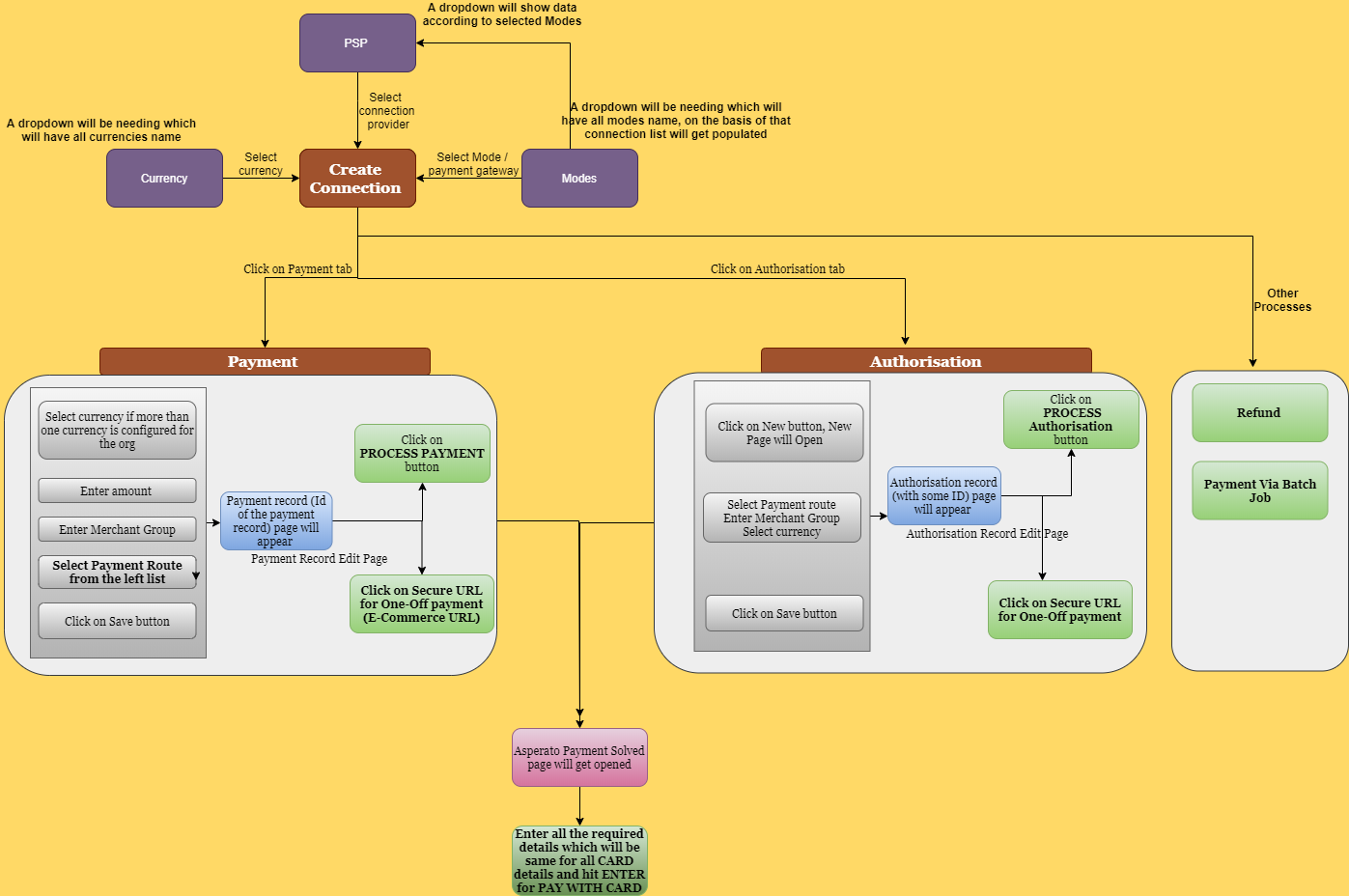
***Phase 1*** *- Covers Priority 2 Test cases as per Test Case Document*

***Phase 2*** *- Covers Priority 3 Test cases as per Test Case Document*

***Phase 3*** *- Covers Payment Schedule test cases*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case Documents** | **Test Case Count** | **Priority 1**  **(Phase 0)** | **Priority 2**  **(Phase 1)** | **Priority 3**  **(Phase 2)** |
| [Asperato generic test cases on lightening platform](https://docs.google.com/spreadsheets/d/1G4qwsFaVzJxUglZINB6NkeGjOjZOqhDCZ2EfRfju17s/edit#gid=1292776730) | 151 | 94 | 34 | 26 |
| [Asperato - USD\_PSP - Lightning\_Test Case Document](https://docs.google.com/spreadsheets/d/1ltRDq8j0wv-b6un4DT5W8iHoEIOOgP3DVdajv-SjFYw/edit) | 256 | 32\* 8 | 256 |  |
| [Asperato - GBP\_PSP - Lightning\_Test Case Document](https://docs.google.com/spreadsheets/d/1Ad_VtgGRN267icQrxPe8Fibf8hNM2zUEVLgrquEUCqg/edit) | 358 | 32\*11 | 352 |  |
| [Asperato - EUR\_PSP - Lightning\_Test Case Document](https://docs.google.com/spreadsheets/d/18ZIcMDi941Kl9Lnqe02v7OZNB6z5H-vpzlUCTckHI2M/edit#gid=1455144840) | 32 |  |  |  |
| [Asperato - JPY\_PSP - Lightning\_Test Case Document](https://docs.google.com/spreadsheets/d/10DzV7PDK41bQnXzvwxBTS2phMGkTfDiE-aJN09KDLEY/edit) | 32 |  |  |  |
| [Asperato generic test cases on classic platform](https://docs.google.com/spreadsheets/d/1vMhj8A3JaKGdVPdCSloN4yuvO2EeUWH-s1dZ19Lq68E/edit#gid=156155796) | 146 | 86 | 36 | 24 |

|  |  |  |
| --- | --- | --- |
| **Phases** | **No of Test Cases** |  |
| Phase 0 | 788 | For single currency, single PSP has been mentioned, but it will cover other connections also -   |  |  | | --- | --- | | GoCardless (Direct Debit) - GBP | 32 | | Stripe (ECOM & MOTO) - GBP | 32 | | Generic test case for lightning platform | 49 | | Generic test case for classic platform | 49 | | Authorize.NET (eCheck) - USD | 32 | |
| Phase 1 | 70 | |  |  | | --- | --- | | Generic test case for lightning platform | 34 | | Generic test case for classic platform | 36 | |
| Phase 2 | 50 | |  |  | | --- | --- | | Generic test case for lightning platform | 24 | | Generic test case for classic platform | 26 | |
| Phase 3 | 37 | Test cases which are related to payment schedule ( lightning and classic) |

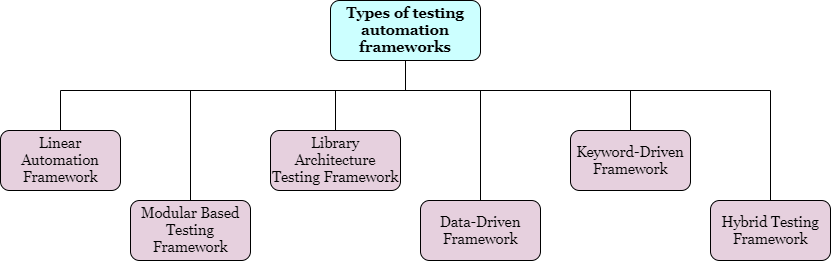


# Design Test Automation framework

A framework defines a set of rules or best practices which we can follow in a systematic way to achieve the desired results. Benefits of test automation framework -

* Improved test efficiency
* Lower maintenance costs
* Minimal manual intervention
* Maximum test coverage
* Reusability of code

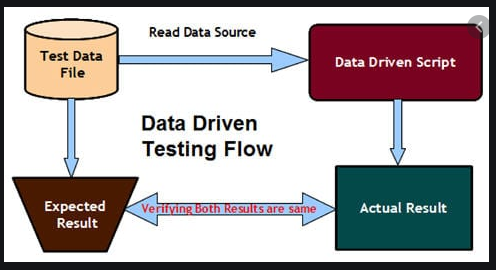
There are different types of testing automation frameworks available -



In Asperato, we can use - Hybrid testing framework which will include modular based testing framework and data-driven framework. Application can be divided into modules - setup, authorisation, payment and refund. POM (project object model) can be used for designing the automation script.

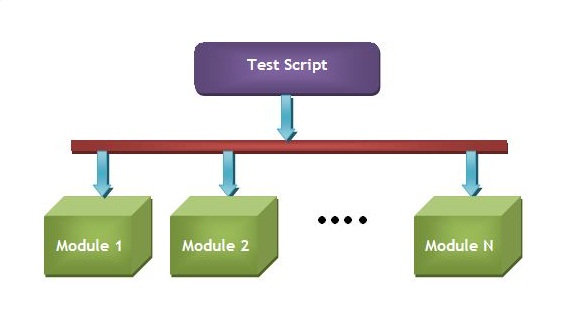
**Data driven framework -**

* Data-driven is a test automation framework which stores test data in a table or spread spreadsheet format.
* In Data-driven test automation framework, input data can be stored in single or multiple data sources like xls, XML, csv, and databases



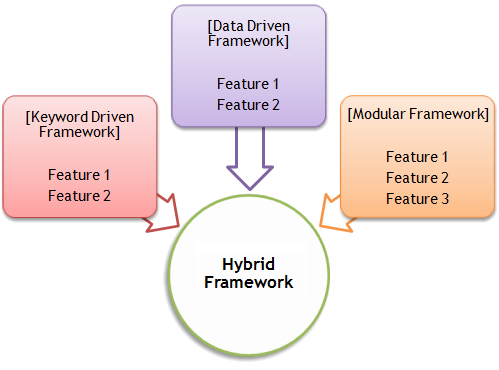
**Modular Based testing framework-**

* *Modular framework is like the creation of small, independent scripts that represent modules, sections and functions of the application under test.*
* *Modular testing framework generally divide an application into multiple modules and create test scripts individually*



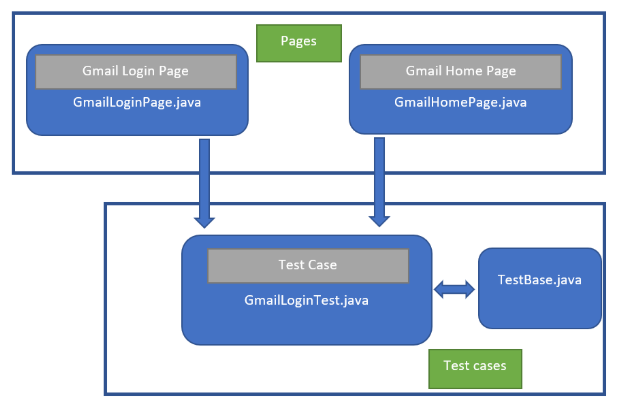
**Hybrid testing framework -**

Hybrid Test framework is a concept where we can use 2 or more frameworks

****

**POM Design Pattern -**

* *POM is a design pattern which is commonly used in Selenium for Automating the Test Cases. This design pattern can be used with any kind of framework like keyword-driven, Data-driven, hybrid framework, etc.*
* The Page object is an object-oriented class which acts as an interface for the page of your Application under test.
* Page class contains web elements and methods to interact with web elements.
* While Automating the test cases, we create the object of these Page Classes and interact with web elements by calling the methods of these classes.



# Developing the Execution Plan

The execution plan includes selecting which environments the scripts will be executed. The environment includes OS, Browser and different hardware configurations.

**Test environment required for Asperato** - Windows (OS), Chrome, Firefox (browsers).

The automation scripts will be run by the testing team (developing the scripts also) on every build and the remainnig other things will depend upon the requirements.

# Writing Scripts

After designing the framework and finalizing the execution planning, scripts can be written.

Scripts should be written in an organized manner with proper naming conventions.

Proper comments should be used for future use.

For eg - Open browser, open login and enter credentials and click on login button

Public void login {

//open browser

System.setProperty("webdriver.chrome.driver","\\define path");

WebDriver driver = new ChromeDriver();

//wait - next statement get execute after 60 seconds

driver.manage().timeouts().implicitlyWait(60, TimeUnit.SECONDS);

//enter username

driver.findElement(By.id("username")).sendKeys("XXX");

//enter password

driver.findElement(By.id("password")).sendKeys("XXX");

//click on login button

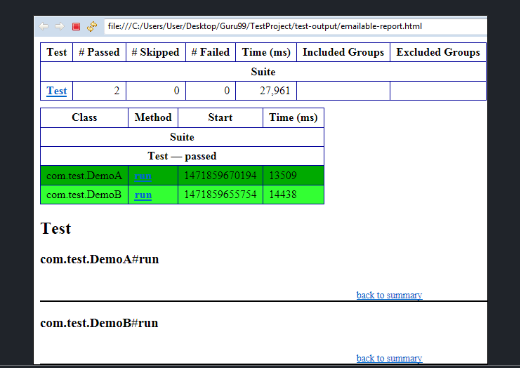
driver.findElement(By.id("Login")).click();

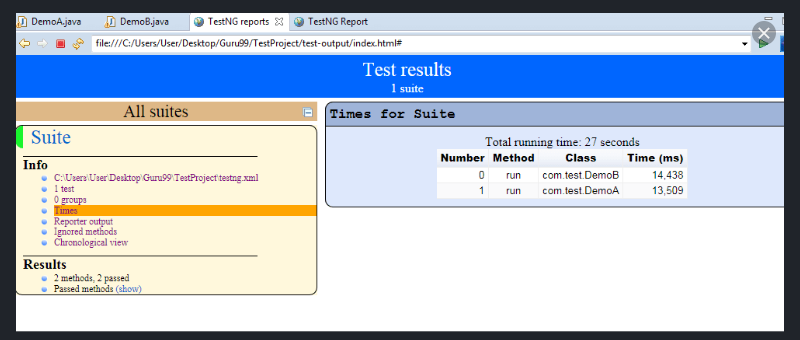
}

# Test Reports

Reports for automated test cases can be managed through Selenium. “Testng” will define the number of pass and fail tests in the console. Report can be viewed in the browser with details.

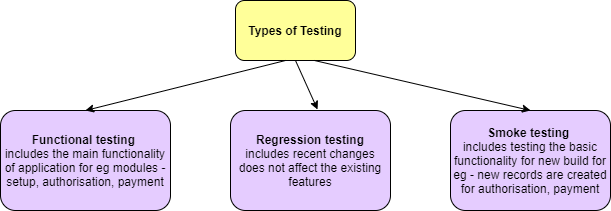
Following are examples of report





# Divide the Test Cases as per Testing type

Any project can have several testing types as per the project duration and project type. On considering Asperato, below are the testing that can be done on several phases of project stage -



**Functional testing** - ensures that the requirements are properly satisfied by the application.

**For eg** -

1. Login into application
2. Setup the new group and connection , delete connection
3. Create authorisation record and process it, edit/delete the record
4. Create payment record and process it, edit/ delete the record
5. Create payment schedule record and save it

**Regression testing** - ensures the new changes do not affect the old and existing functionality.

**For eg** - any change in payment module should not affect its existing functionality of creating record, process payment, edit/delete record

**Smoke testing** - ensures that the basic functionality is working properly with new build

**For eg -**

1. Setup the new group and connection
2. Create authorisation record and process it
3. Create payment record and process it
4. Create payment schedule record and save it