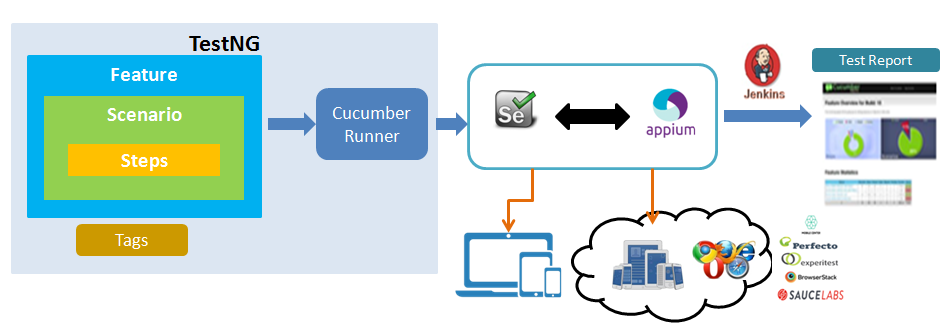
Approach: Behaviour driven approach

**Technology stack used:** BDD cucumber, Junit/TestNG Runner, Java, Feature files, JSON, XML for raw results storage, extent reporting libs for rich HTML preview, Eclipse/Intelli J IDE, Selenium libs, Appium libs(In case of Mobile), browser drivers, Maven, Jenkins for Continuous execution, Other tools ( Galen libs for layout testing, Sikuli libs for Image based execution)

**Framework Design Diagram:**



As shown in the above diagram it’s very clear that this approach follows BDD Cucumber approach with Java language.

I will convert all the business requirements in to Feature files, Write couple of scenario’s for the feature which we are intended to automate, Write all the scenario steps, examples, gather test data and complete the creation of feature file.

Once the feature file is ready configure the Cucumber Runner class to execute the scenario’s and generate reports..etc.

Generate Gluecode for all the scenario steps and implement the logic(Selenium script with Internal framework)

On very high level my Internal framework contains KeywordClass, PageObjectClass, ReportingClass, ConfigurationClass, DriverClass, ComponentClass.

**KeywordClass** – Contains all the keywords of Selenium functionality (For eg: click(), elementSendText(), waitForElementPresence(), dragAndDrop(), wait(), sleep() ..etc)

**PageObjectClass** – All the XPath and other elements are recorded in this classes, for better standard create separate class for separate pages in the application – this is equivalent to page object model.

**ReportingClass** – reporting class get logs for every test step execution, since every keyword is surrounded with try-catch exception handling, Pass/Fail log status will be recorded as XML tags vis reporting class.

**ConfigurationClass** – used to maintain and supply configuration for test executions, like number of test cases to be execute, which test data sheet to pickup, which browser, which machine(in case of Grid), name of project for reporting, number of devices to execute the script and others..

**DriverClass** – contains the configuration of drivers and the respective browsers, and mobile devices configurations

**ComponentClass –** is used to hold the components of sequence of test steps.

**Test Case Creation Approach:**

We write test cases using our own keywords For eg:

**Click**(HomePage.loginButton);

elementSendText(HomePage.userName, userName)

**Click - our custom keyword**

HomePage – Name of Page which Element is recorded

loginButton – elementName in HomePage

Test data can be directly provided from feature file with Examples via Scenario outline or we can store in any file formats like Excel, JSON or Database and write helper classes to supply test data.

**For Execution:** CucumberRunner class is used to execute and generate the results via Junit/TestNG runner classes. Rich HTML reports can be generated via our custom report storage file (XML/JSON which are generated while execution) or third party libraries like Extent reports can serve the purpose.

**An Example feature file for the Given scenario look as shown below;**

Feature: This file contain book a product Feature

Scenario: This is a scenario used to book a product in Farmfresh and purchase and verify the payment.

Given User is already registered and logs in

Then User will add few items in to the cart

And User will moves/navigates to Summary Page

Then User will verify the order

And User will select option to make payment

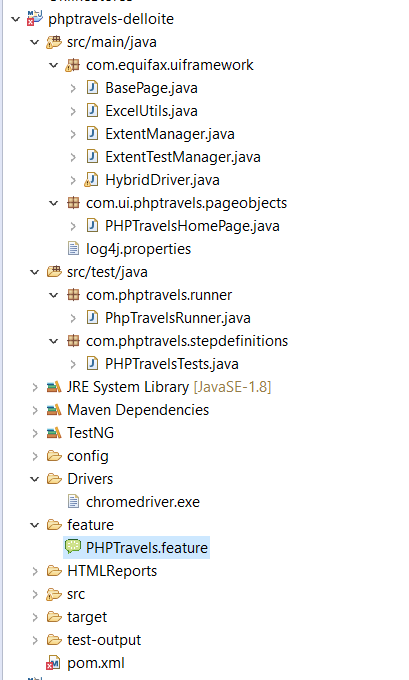
Then User will navigate to third party bill desk payment gateway and complete payment process

And User will verify the email for bill.

**Justification :**

* This approach will not have any code duplications. If the test steps are repeated same step definition can be reused across the application.
* Less time to develop the test case (Glue code and other configuration will be automatically created)
* Reports are created automatically – No code write is required from the tester to generate framework will handle.
* Test data organization is simple.
* Execution is a cake walk. (can be execute from Maven life cycle, Jenkins, or ANT build, direct from Eclipse, or From command prompt ..etc has many options to execute the test scenario’s)
* Follows BDD approach therefore anyone can understand the test case by reading Feature files.
* If teams are working together( BA, Dev and QA) then tester Job is simple to generate Glue code and execute, Feature files will provided by BA and Dev teams

After importing the project to eclipse IDE/Intelli J it looks like below;



All the elements are recorded in **com.ui.phptravels.pageobjects**  with classes **PhpTravelsHomePage**

**Features files**  are stored in Feature folder.

**Test Runner**  is in com.phptravels.runner and **Step Definitions** are in com.phptravels.stepdefinitions

Test execution can be done test runner.