Automation Test Plan

**Document Revisions**

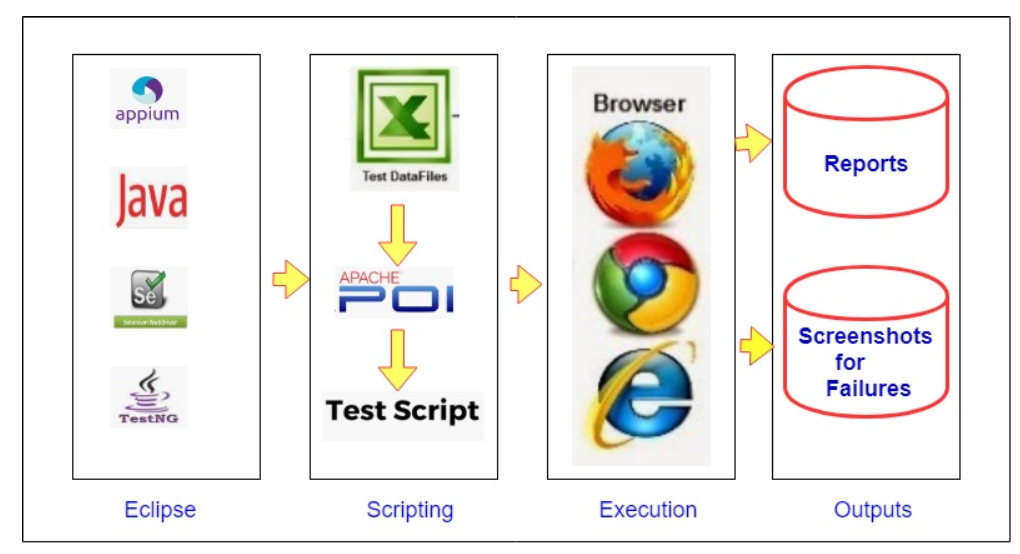
|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 31/05/2019 | 0.1 | Initial draft | Dimuthu Ramachandra |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 1. Introduction

This document describes the Architecture of the MIT Test Automation Framework & Guildlike on how to use the framework

# 2. Architecture of the framework

**2.1 Architectural Diagram**



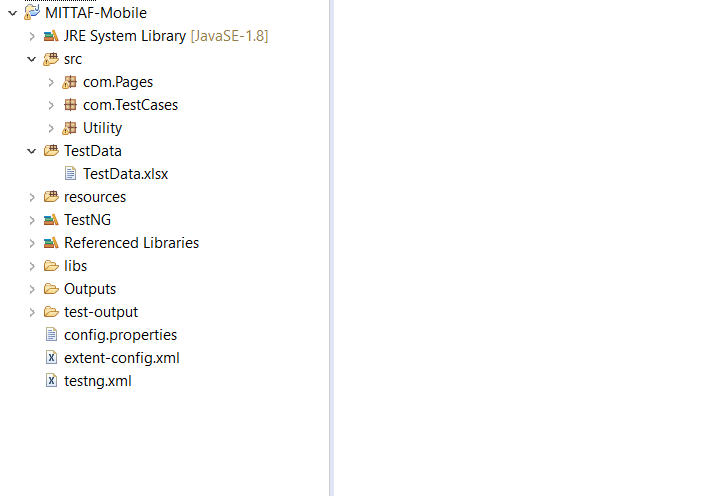
**2.1 Tools & Technologies**

* Eclipse is used as IDE
* Appium is used for the mobile automation tool
* Java is as the Primary programing language to build the framework
* Selenium WebDriver is used as execution engine
* TestNG is used as the Testing framework. It is used for assertions, annotations & screenshot capturing
* Excel is used to maintain the test data
* Apache POI is used to read data from excel

**2.1 Features**

* Support android
* It is data driven & test data is maintained in excel
* It is keyword driven. User actions has been written as keywords
* It uses Page Object Model as the design pattern
* It generates a html report using extent reports
* It is capturing screen shot to the report for the failure test cases
* It is maintaining configurable data such as “Andriod or IOS”, in property file so that use is able to change without accessing code

# 4.How the framework works?



**“Utility “package** – Contains all the reusable methods & libraries

* ExcelReader.java – Contains the library to read data from excel
* Reporter.java – Contains the library to capture screen shot & attach that to execution reports
* AppiumTestBase.java – Contain all the browser set up, tear down methods
* TestBase\_Commands.java – Contains all the keywords for the user actions
* Util.Java – Contain method to read data from property file

**“Com.Pages” package** – All the page objects will be placed in com.Pages package

**“Com.TestCases” package** – All the test classes (test cases) will be places in com.TestCases package

**Test Data folder** – All the test data sheets will be placed in Test Data folder

**Outputs->Report folder –** Execution report will be generated & saved in Report folder inside Output folder

**Outputs->** **ScreenshotsFailure–** Screenshots for the fail test cases will be captured & saved in ScreenshotsFailure folder inside Output folder

# 4.Guild lines on How to use the framework?

**4.1. Adding Page objects**

* Create a page object in com.Pages package

You need to create a class for a one page in the system & name the class as “<PageNamem>page.java”

**Naming Conventions:**

Class name – Declare the class name as <PageName>Page &Start the first letter as capital

E.g: LoginPage.java

* Extend the page class from “TestBase\_Commands” class

**public** **class** LoginPage **extends** TestBase\_Commands

{

}

* Create a constructor

**public** LoginPage(AndroidDriver driver) {

**this**.*driver* = driver;

}

* Declare object locators as private variables

**private** By tf\_PhoneNumber =

By.*xpath*("//android.widget.EditText[@text='Phone number']");

**private** By tf\_PinNumber =

By.*xpath*("//android.widget.EditText[@text='Pin number']");

**private** By btn\_Submit =

By.*xpath*("//android.widget.TextView[@text='Submit']");

**Naming Conventions:**

Object locators

Text Field – tf\_<FieldName>

Dropdown - dd\_<Dropdown Name>

Button - btn\_<Button Name>

Label - lbl\_<Label Name>

Message - msg\_<Message Name>

Radio Button - rdo\_<Radio Button Name>

Check Box - chk\_<Check Box Name>

Links - lnk\_<Link Name>

Images - img\_<Image Name>

Icon - icn\_<Icon Name>

* Write a public method to functionality by parameterizing the inputs. We will call this as the bf “business functions”

//login to application

//Login to application

**public** **void** bf\_Login(String prm\_phoneNumber, String prm\_pinNumber) {

Type(tf\_PhoneNumber, prm\_phoneNumber);

CheckElementPresent(btn\_Submit);

Tap(btn\_Submit);

VerifyText(tf\_PinNumber, "Pinnumber");

Type(tf\_PinNumber, prm\_pinNumber);

Tap(btn\_Submit);

}

**Naming Conventions:**

Bf(Method Name) – Name as the User Action & Start with the “bf\_” & First letter as capital

E.g.: bf\_Login, bf\_AddRecoed, bf\_EditRecord

Parameters (Inputs) – declare as prm\_<parameterName> & Start with the first letter of the firs word as simple & first letter second word as Capital

E.g.: prm\_username, prm\_studentName

**4.2. Adding Test Classes**

* Create a separate class for the corresponding page in com. TestCases package & extend from SeleniumTestBase class

**public** **class** LoginPage\_Tests **extends** AppiumTestBase

{

}

**Naming Conventions:**

Class name - <PageName>\_Tests

E.g. LoginPage\_Tests

* Create an object of corresponding page class

E.g. LoginPage \_LoginPage;

* Create a method for the test case with @Test annotation

//login to application

@Test(priority = 1, enabled = **true**)

**public** **void** tc\_Verifylogin() {

\_LoginPage = **new** LoginPage(driver);

\_LoginPage.bf\_Login(ExcelReader.*getData*(0, 1, 0), ExcelReader.*getData*(0, 1, 1));

}

**Naming Conventions:**

Use a meaning full name. You may use test case name

Start with the “tc\_” & first letter of the first word as capital & first letter of second word as capital

E.g. tc\_VerifyLogin()

* Call the object of corresponding page class

@Test(priority=1)

**public** **void** tc\_Verifylogin ()

{

\_LoginPage = new LoginPage(driver);

}

* Call the relevant business functions implemented in corresponding page

@Test(priority=1)

**public** **void** tc\_Verifylogin ()

{

LoginPage = new LoginPage(driver);

\_LoginPage.bf\_Login(“Input Date”,”Input Date”));

}

* Mention the test data in the test data sheet located in Test Data folder
* Pass the data read from excel in to test script

@Test(priority=1)

**public** **void** Verifylogin ()

{

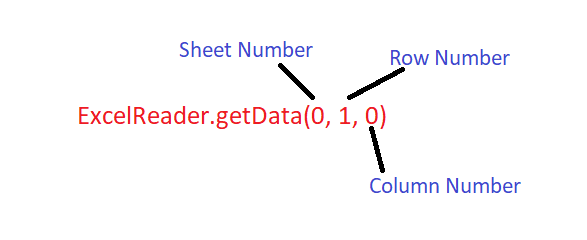
LoginPage = new LoginPage(driver);

\_LoginPage.bf\_Login(ExcelReader.getData(0, 1, 0), ExcelReader.getData(0, 1, 1));

}

You can use the below method to read & pass the test data from the test data sheet

ExcelReader.getData(0, 1, 0)

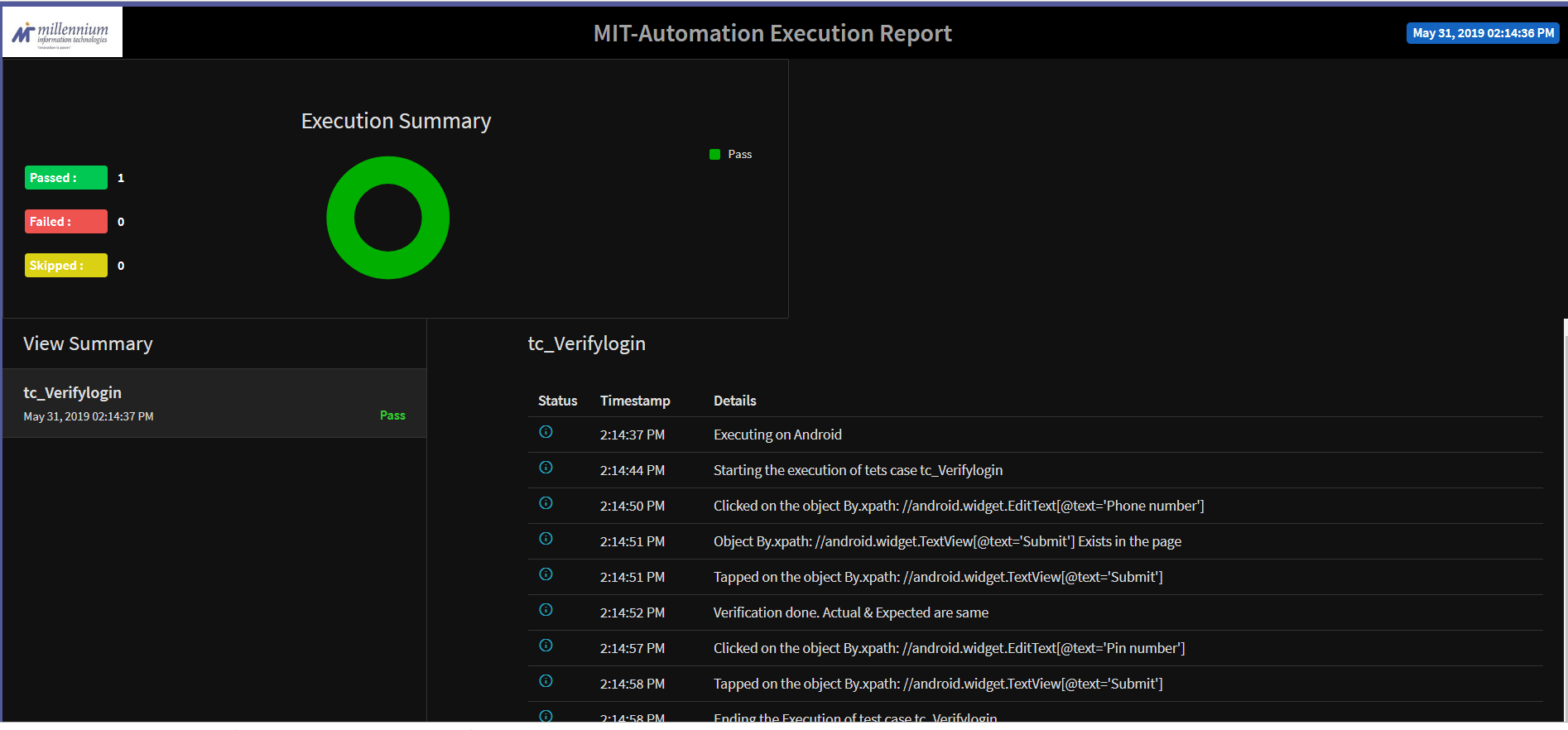


**4.3. Executing Tests**

* Right click on test class located in the com.TestCase packages & Click Run as TestNG Test

**4.4. Verifying the result**

* Navigates to Report folder & Open the report



# 5.Commands Repository

All the user actions have been written as reusable commands

1. Type (“prm1”,” prm2”)

|  |  |
| --- | --- |
| Description | This command is used to enter inputs for the text fields |
| Parameters | prm1 – object locator of the text field prm2 – Input date to be entered |
| Selenium Command | driver.sendKeys("") |

1. Tap (“prm1”)

|  |  |
| --- | --- |
| Description | This command is used to click on an element |
| Parameters | prm1 – object locator of the element to be clicked |
| Selenium Command | driver.Click() |

1. VerifyText (“prm1”,” prm2”)

|  |  |
| --- | --- |
| Description | This command is used to verify the text message or label name of an element |
| Parameters | prm1 – object locator of the field or message to be verified prm2 – Expected message or text |
| Selenium Command | Assert.assertEquals(); |

1. VerifyTitle (“prm1”)

|  |  |
| --- | --- |
| Description | This command is used to verify the title of the page |
| Parameters | prm1 – Expected Title to be verified |
| Selenium Command | driver.getTitle(); |

1. CheckElementPresent (“”)

|  |  |
| --- | --- |
| Description | This command is used to verify whether element is presented & visible in a web page |
| Parameters | prm1 – object locator of element |
| Selenium Command | driver.isDisplayed (); |

1. IsElementPresent (“”)

|  |  |
| --- | --- |
| Description | This Command is to get the Boolean status of element present |
| Parameters | prm1 – object locator of element |
| Selenium Command | driver.isDisplayed (); |

1. CheckElementEnabled (““)

|  |  |
| --- | --- |
| Description | This command is used to verify whether particular element is enabled |
| Parameters | prm1 – object locator of element |
| Selenium Command | driver.isEnabled (); |

1. CheckElementSelected (““)

|  |  |
| --- | --- |
| Description | This command is used to verify whether particular element is selected |
| Parameters | prm1 – object locator of element |
| Selenium Command | driver.isSelected () |

1. VerifyElementProperty (“”,””,””)

|  |  |
| --- | --- |
| Description | This command is used to verity the property of a WebElement |
| Parameters | prm1 – object locator of element prm2 – Property of the element prm3 – Expected value of the property |
| Selenium Command | driver.getAttribute (""); |

1. SelectValueFromDrowDown (“”,””,””)

|  |  |
| --- | --- |
| Description | This command is used to select a value from dropdown |
| Parameters | prm1 – object locator of dropdown prm2 – Select Type (SelectByText, SelectByIndex, SelectByValue) prm3 – Option to be selected |
| Selenium Command | Select ddValue = new Select (); |

1. Clear (“”)

|  |  |
| --- | --- |
| Description | This command is used to clear the input in a text field |
| Parameters | prm1 – object locator of element |
| Selenium Command | driver. Clear (); |

1. StoreText ("")

|  |  |
| --- | --- |
| Description | This command is used to store a text message into a variable |
| Parameters | prm1 – object locator of element |
| Selenium Command | driver.getText (); |

1. StoreProperty (“”,””)

|  |  |
| --- | --- |
| Description | This command is used to store a property of an element in to a string |
| Parameters | prm1 – object locator  prm2 – property of the element to be stored |
| Selenium Command | builder. driver. getAttribute(); |

1. getCount ("")

|  |  |
| --- | --- |
| Description | This command is used to get a item count in a element table or list |
| Parameters | prm1 – object locator of element |
| Selenium Command | size (); |

1. VefirySelectedValue (“”,””)

|  |  |
| --- | --- |
| Description | This command is used to verify the default value in a drop down |
| Parameters | prm1 – object locator of element prm2 – expected default option |
| Selenium Command | getFirstSelectedOption (); |

1. WaitForElementPresent ("","")

|  |  |
| --- | --- |
| Description | This command is used to wait specific amount of time till element is presented (Explicit Waits) |
| Parameters | prm1 – object locator to be presented prm2 – time in second to be wait |
| Selenium Command | wait = new WebDriverWait (driver, ""); wait.until(ExpectedConditions.presenceOfElementLocated("")); |

1. NavigateBack()

|  |  |
| --- | --- |
| Description | This command is used to navigates back to previous page |
| Parameters | N/A |
| Selenium Command | driver. navigate().back(); |

1. NavigateFoarwad()

|  |  |
| --- | --- |
| Description | This command is used to navigates forwad to next page |
| Parameters | N/A |
| Selenium Command | driver. navigate().forward(); |

1. Wait (“”)

|  |  |
| --- | --- |
| Description | This command is used to wait specific amount of time (Implicit Waits) |
| Parameters | prm1 – time in second to be wait |
| Selenium Command | driver.manage().timeouts().implicitlyWait("",TimeUnit.SECONDS); |

1. Sleep (“”)

|  |  |
| --- | --- |
| Description | This command is used to wait specific amount of time (Tread Sleep) |
| Parameters | prm1 – time in second to be wait |
| Selenium Command | Thread.sleep("") |

1. PreseKey (“”)

|  |  |
| --- | --- |
| Description | This command is used to press the key board keys |
| Parameters | prm1 – Name of the key |
| Selenium Command | Robot \_Robot = new Robot(); \_Robot.keyPress(KeyEvent.VK\_ENTER) \_Robot.keyRelease(KeyEvent.VK\_ENTER); |