Test Automation Architecture For TPIP

TABLE OF CONTENTS

Document Revisions

Version	Status	Date	Change Summary	Owner/Author
0.1	Completed	18/06/2021	Initial Version	Lalindra

Document Handling

This document is strictly for 1 Billion Tech and Client use only. This shall not be shared with any external party other than the client concerned. This document should always be kept securely, and employees shall use reasonable care protecting this from unauthorized use or disclosure to a third party.

1 Introduction

1.1 *Purpose*

This document presents the comprehensive architectural overview of the Test-Automation solution for **TPIP Test Automation** project. This Automation Architecture document will be used as the basis for all test automation activities. The intended audience for this document is the **TPIP** Project Team and the **1 Billion Tech** Project Team.

1.2 *Scope*

This document is related to the test automation done for the selected set of Regression Test Cases of **LGIM NavGuide** application by the 1 Billion Tech QA Team.

This document contains the scope of the automation, test framework, quality attributes, the environment details, and the critical success factors of the solution.

1.3 *Definitions, Acronyms, and Abbreviations*

Term	Description
NA	Not Applicable
TBD	To be determined
UI	User Interface
UC	Use Case
BDD	Behavior Driven Development
PO	Page Object

Table 1: Definitions, Acronyms, and Abbreviations

1.4 *References*

No	Document Name	Version	Location
1	Regression Test Suite	NA	https://drive.google.com/drive/u/0/folders/1cQkV8r_oCQdH20rl3iWurYVxtcDYVboF

Table 2: Reference Documents

2 Scope of the Automation

The scope of this automation is limited to the Registration Regression and the details of the In-Scope areas are as follows.

2.1 In-Scope

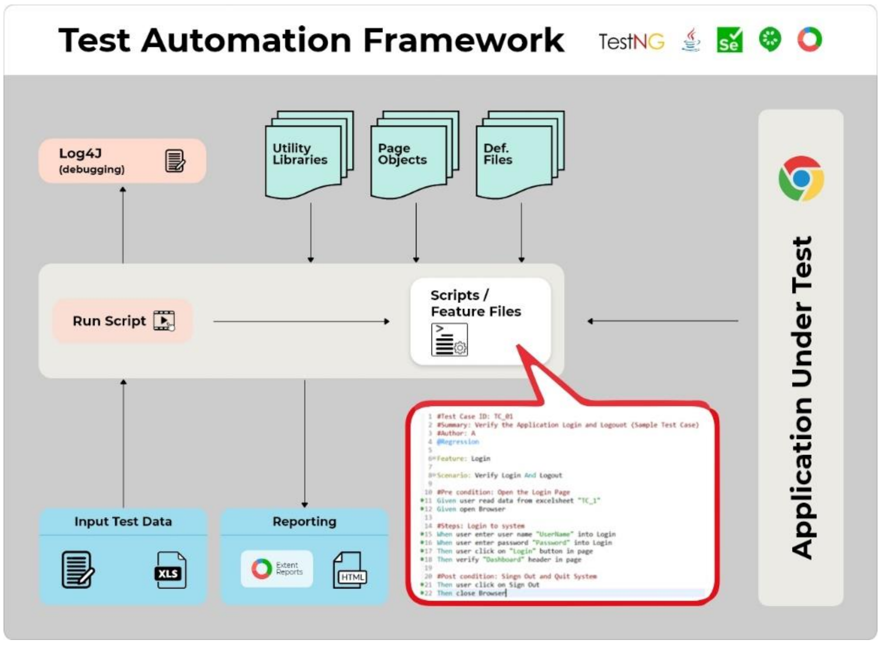
Scenario Group #	End to End Scenario
1	Verify completion of Initial proposal with client approval using document generation. (Individual scheme with Incumbent Managers)
	Verify completion of IAA
	Verify completion of IMA
	Verify completion of PMC
	Verify completion of advisory report
	Verify completion of termination letter
	Verify activation of scheme by LGIM
2	Verify completion of Initial proposal with LGIM approval using document generation. (Corporate scheme without Incumbent Managers)
	Verify completion of IAA
	Verify completion of IMA
	Verify completion of PMC
	Verify completion of advisory report
	Verify activation of scheme by LGIM
3	Verify completion of Initial proposal with LGIM approval using document generation. (Corporate scheme with Incumbent Managers)
	Verify completion of IAA
	Verify completion of IMA
	Verify completion of PMC
	Verify completion of advisory report
	Verify activation of scheme by LGIM
4	Verify completion of Initial proposal with Client approval using document generation. (Individual scheme without Incumbent Managers)
	Verify completion of IAA
	Verify completion of IMA
	Verify completion of PMC
	Verify completion of advisory report
	Verify activation of scheme by LGIM

Table 3: Regression Scenarios

Note: *These 25 scenarios are covered within 4 End-To-End Scenario scripts (i.e., Scenario Group 1-4).*

3 Test Framework

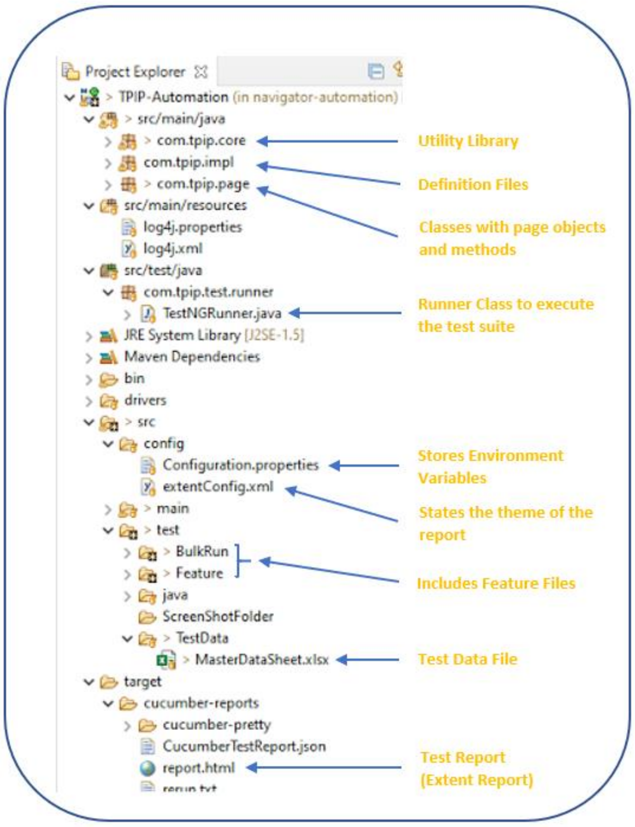
This framework relies on Selenium and Cucumber in order to support BDD which makes it easy to read and to understand the application flow for any user of the automated suite.



```
#Test Case ID: TC_01
#Summary: Verify the Application Login and Logouot (Sample Test Case)
#Author: A
@Regression

Feature: Login

Scenario: Verify Login And Logout

#Pre condition: Open the Login Page
Given user read data from excelsheet "TC_1"
Given open Browser

#Steps: Login to system
When user enter user name "UserName" into Login
When user enter password "Password" into Login
Then user click on "Login" button in page
Then verify "Dashboard" header in page

#Post condition: Singn Out and Quit System
Then user click on Sign Out
Then close Browser
```

Figure 1: Automation Framework

The structure of the automation suite is illustrated in the image below.



```
Project Explorer
v > TPIP-Automation (in navigator-automation)
  v > src/main/java
    > > com.tpip.core
    > com.tpip.impl
    > > com.tpip.page
  v src/main/resources
      log4j.properties
      log4j.xml
  v src/test/java
    v com.tpip.test.runner
      > TestNGRunner.java
  > JRE System Library [J2SE-1.5]
  > Maven Dependencies
  > bin
  > drivers
  v > src
    v config
        Configuration.properties
        extentConfig.xml
    > > main
    v > test
      > > BulkRun
      > > Feature
      > java
        ScreenShotFolder
      v > TestData
          > MasterDataSheet.xlsx
  v target
    v cucumber-reports
      > cucumber-pretty
        CucumberTestReport.json
        report.html
        rerun.txt
```

Utility Library
Definition Files
Classes with page objects and methods
Runner Class to execute the test suite
Stores Environment Variables
States the theme of the report
Includes Feature Files
Test Data File
Test Report (Extent Report)

Figure 2: Framework Structure

3.1 Library

This consists of common class files that can be used application wide. All these Library files (e.g. Driver class, ExcelUtils class, Configuration File reader class, etc.) are packaged under the “com.tpip.core” package.

3.2 Definition Files

Step definition files are used to store the mapping between each step of the scenario defined in the feature file with the code of function to be executed which are defined under page object classes. These files are packaged under the “com.tpip.impl” package.

3.3 Drivers

Driver will be handled by the “Driver” class located in the com.tpip.core package.

Driver exe file location will be teken from the “Configuration” property file located in “\\src\\config\\Configuration.properties”.

Chrome Driver is used for the solution, as the automated suite developed and planning to execute on Chrome browser.

3.4 Object Repository

All the classes which consist of all page objects and reusable common methods will be handled under the Object Repository. These classes are packaged under the “com.tpip.page” package.

- PO Package:

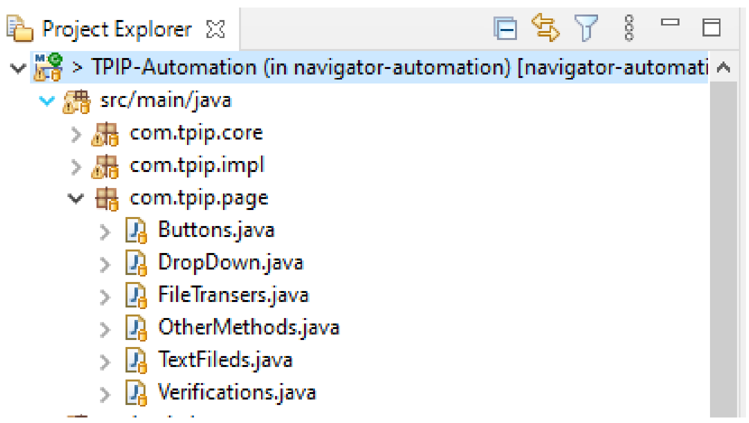
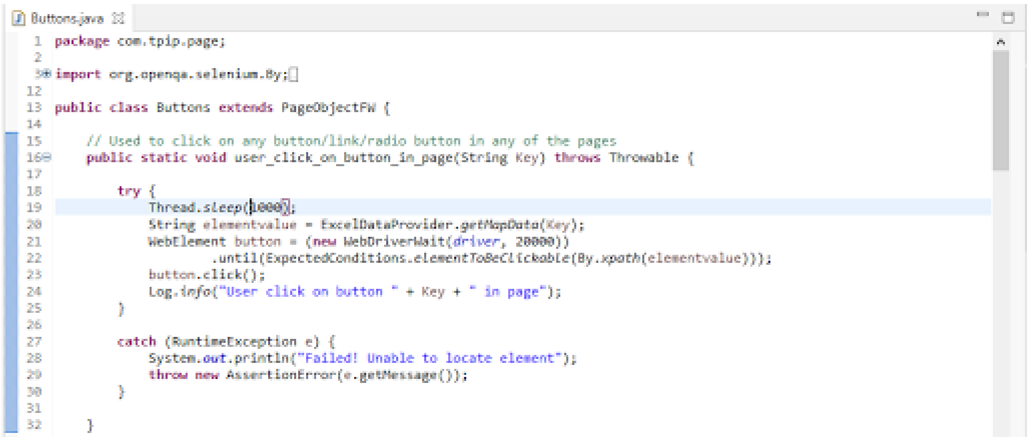


Figure 3: Page Object Classes

- Sample code snippet:



Buttons.java

```
package com.tpip.page;

import org.openqa.selenium.By;

public class Buttons extends PageObjectFW {

    // Used to click on any button/link/radio button in any of the pages
    public static void user_click_on_button_in_page(String Key) throws Throwable {

        try {
            Thread.sleep(1000);
            String elementvalue = ExcelDataProvider.getMapData(Key);
            WebElement button = (new WebDriverWait(driver, 20000))
                    .until(ExpectedConditions.elementToBeClickable(By.xpath(elementvalue)));
            button.click();
            Log.info("User click on button " + Key + " in page");
        }

        catch (RuntimeException e) {
            System.out.println("Failed! Unable to locate element");
            throw new AssertionError(e.getMessage());
        }

    }
```

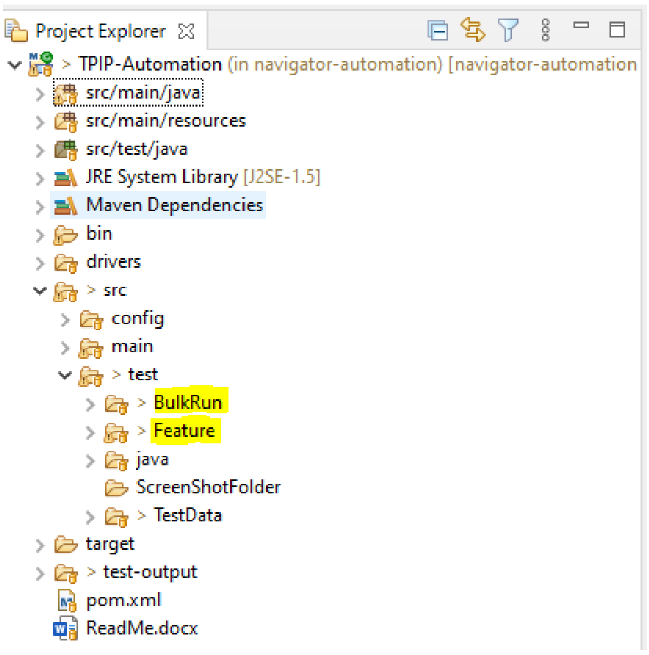
Figure 4: Sample Code

3.5 Test Case Management

Separate feature file has been created for each test case/scenario. Without packaging all test cases in a single folder, several folders can be created to group the test cases at the UC / Feature level for easy maintenance.

Given below is the list of folders aligned with the solution to store test cases.

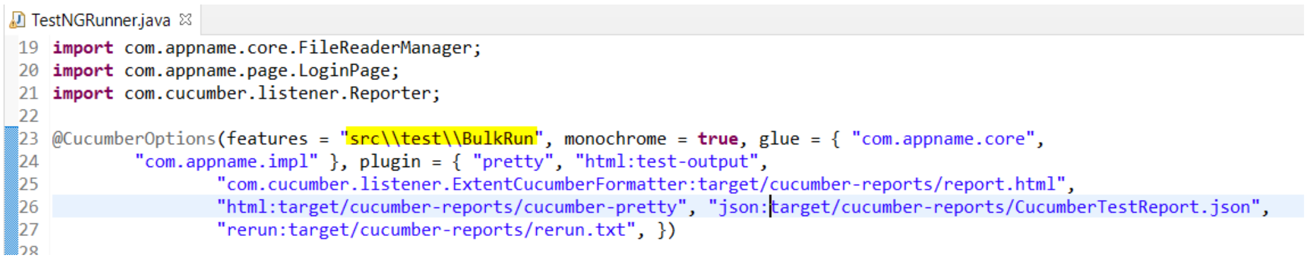
1. BulkRun – To maintain the Feature Files/Test Cases to be executed in a particular cycle.
2. Feature – To maintain all the Feature Files/Test Cases belonging to the complete project.



```
Project Explorer
v > TPIP-Automation (in navigator-automation) [navigator-automation
  > src/main/java
  > src/main/resources
  > src/test/java
  > JRE System Library [J2SE-1.5]
  > Maven Dependencies
  > bin
  > drivers
  v > src
    > config
    > main
    v > test
      > > BulkRun
      > > Feature
      > java
        ScreenShotFolder
      > > TestData
  > target
  > > test-output
    pom.xml
    ReadMe.docx
```

Figure 5: Feature File Folders

Currently the Runner class is configured to run the Feature files under "BulkRun" folder as highlighted in the below image, but this path can be changed accordingly to run the Feature files under other folders as well.

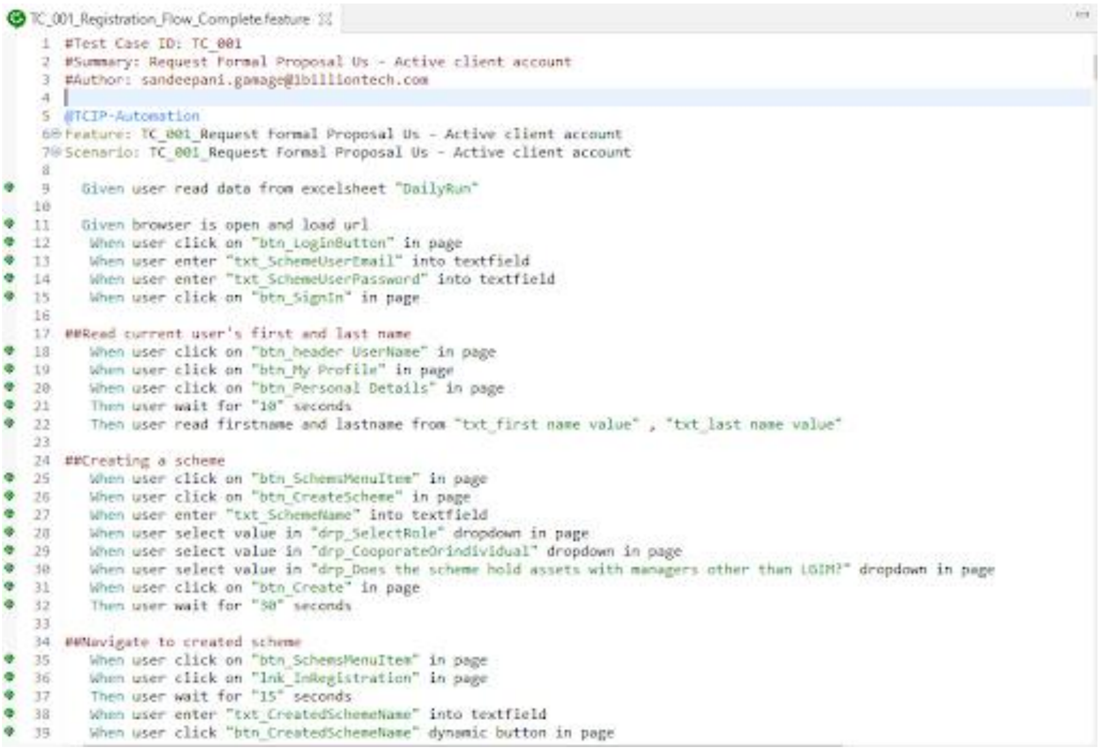


TestNGRunner.java

```
import com.appname.core.FileReaderManager;
import com.appname.page.LoginPage;
import com.cucumber.listener.Reporter;

@CucumberOptions(features = "src\\test\\BulkRun", monochrome = true, glue = { "com.appname.core",
        "com.appname.impl" }, plugin = { "pretty", "html:test-output",
                "com.cucumber.listener.ExtentCucumberFormatter:target/cucumber-reports/report.html",
                "html:target/cucumber-reports/cucumber-pretty", "json:target/cucumber-reports/CucumberTestReport.json",
                "rerun:target/cucumber-reports/rerun.txt", })
```

Figure 6: Feature File Folder Path in Runner Class

- Sample code snippet of a Feature File / Test Case:



TC_001_Registration_Flow_Complete.feature

```
#Test Case ID: TC_001
#Summary: Request Formal Proposal Us - Active client account
#Author: sandeepani.gamage@1billiontech.com

@TCIP-Automation
Feature: TC_001_Request Formal Proposal Us - Active client account
Scenario: TC_001_Request Formal Proposal Us - Active client account

  Given user read data from excelsheet "DailyRun"

  Given browser is open and load url
   When user click on "btn_LoginButton" in page
   When user enter "txt_SchemeUserEmail" into textfield
   When user enter "txt_SchemeUserPassword" into textfield
   When user click on "btn_SignIn" in page

##Read current user's first and last name
   When user click on "btn_header UserName" in page
   When user click on "btn_My Profile" in page
   When user click on "btn_Personal Details" in page
   Then user wait for "10" seconds
   Then user read firstname and lastname from "txt_first name value" , "txt_last name value"

##Creating a scheme
   When user click on "btn_SchemsMenuItem" in page
   When user click on "btn_CreateScheme" in page
   When user enter "txt_SchemeName" into textfield
   When user select value in "drp_SelectRole" dropdown in page
   When user select value in "drp_CooporateOrindividual" dropdown in page
   When user select value in "drp_Does the scheme hold assets with managers other than LGIM?" dropdown in page
   When user click on "btn_Create" in page
   Then user wait for "30" seconds

##Navigate to created scheme
   When user click on "btn_SchemsMenuItem" in page
   When user click on "lnk_InRegistration" in page
   Then user wait for "15" seconds
   When user enter "txt_CreatedSchemeName" into textfield
   When user click "btn_CreatedSchemeName" dynamic button in page
```

Figure 7: Sample Feature File

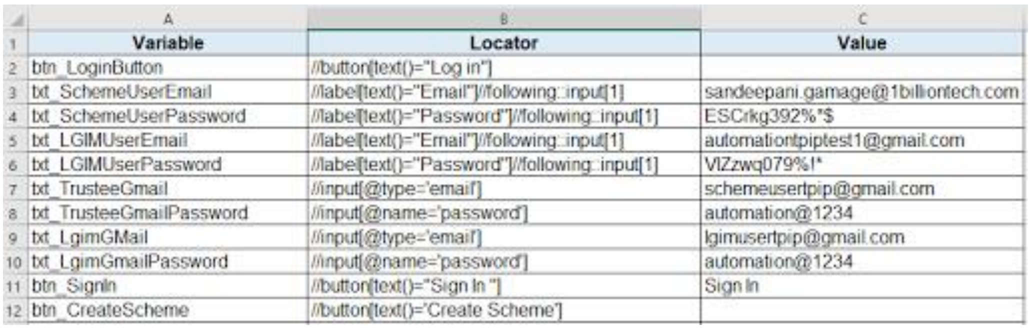
3.6 *Data Management*

This framework is blended with the Data Driven capabilities by storing the data in spreadsheet format to maintain all Test Data separately.

A separate class named: "ExcelUtils" has been used to implement this capability.

The spreadsheet is located under "/webcrm/src/test/TestData/" and the data will be grouped under different tabs for each Feature file.

This spreadsheet comprises Variable, Locator, and the Data Value. The structure and the details of the data file looks like below.



	A	B	C
1	**Variable**	**Locator**	**Value**
2	btn_LoginButton	//button[text()="Log in"]	
3	txt_SchemeUserEmail	//label[text()="Email"]//following::input[1]	sandeepani.gamage@1billiontech.com
4	txt_SchemeUserPassword	//label[text()="Password"]//following::input[1]	ESCrkg392%*$
5	txt_LGIMUserEmail	//label[text()="Email"]//following::input[1]	automationtpiptest1@gmail.com
6	txt_LGIMUserPassword	//label[text()="Password"]//following::input[1]	VIZzwq079%!*
7	txt_TrusteeGmail	//input[@type='email']	schemeusertpip@gmail.com
8	txt_TrusteeGmailPassword	//input[@name='password']	automation@1234
9	txt_LgimGMail	//input[@type='email']	lgimusertpip@gmail.com
10	txt_LgimGmailPassword	//input[@name='password']	automation@1234
11	btn_SignIn	//button[text()="Sign In "]	Sign In
12	btn_CreateScheme	//button[text()='Create Scheme']	

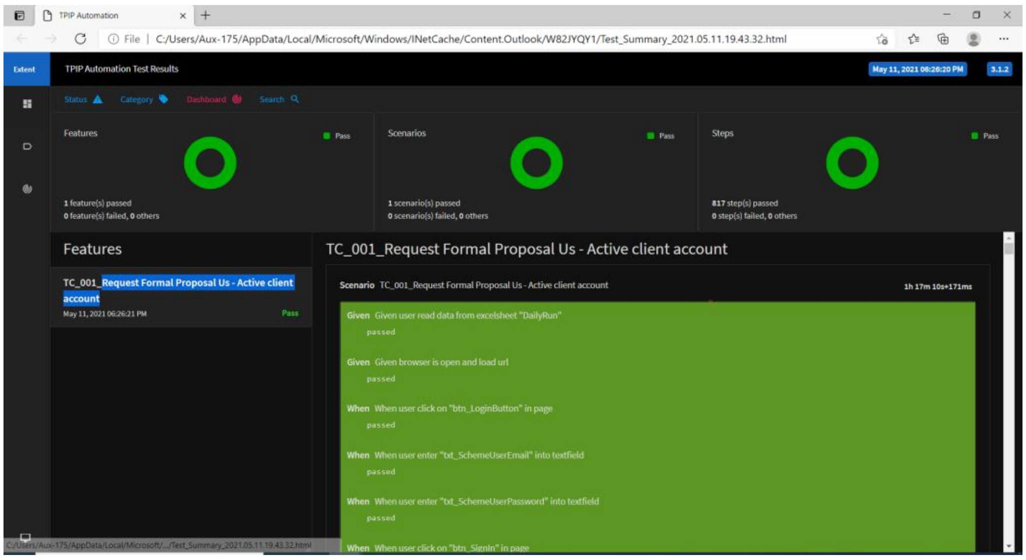
Figure 8: Data Sheet Attributes

3.7 *Test Reporting*

In order to make the reports more interactive and detailed to the user, Extent Reports are used to provide the final test results. Each Test Case will be designed in a manner to log a custom message in the report at each major verification point to show whether it satisfies the expected result or not.

All failures are tracked by attaching a screen shot of the failed step to provide a comprehensive view of the failure.

- Sample Extent Report:



TPIP Automation

C:/Users/Aux-175/AppData/Local/Microsoft/Windows/INetCache/Content.Outlook/W82JYQY1/Test_Summary_2021.05.11.19.43.32.html

Extent

TPIP Automation Test Results

May 11, 2021 06:26:20 PM

3.1.2

Status Category Dashboard Search

Features — Pass
1 feature(s) passed
0 feature(s) failed, 0 others

Scenarios — Pass
1 scenario(s) passed
0 scenario(s) failed, 0 others

Steps — Pass
817 step(s) passed
0 step(s) failed, 0 others

Features

TC_001_Request Formal Proposal Us - Active client account
May 11, 2021 06:26:21 PM Pass

TC_001_Request Formal Proposal Us - Active client account

Scenario TC_001_Request Formal Proposal Us - Active client account 1h 17m 10s+171ms

Given Given user read data from excelsheet "DailyRun"
passed

Given Given browser is open and load url
passed

When When user click on "btn_LoginButton" in page
passed

When When user enter "txt_SchemeUserEmail" into textfield
passed

When When user enter "txt_SchemeUserPassword" into textfield
passed

When When user click on "btn_SignIn" in page

C:/Users/Aux-175/AppData/Local/Microsoft/.../Test_Summary_2021.05.11.19.43.32.html

Figure 9: Extent Report

4 Quality Attributes

4.1 Maintainability

As part of achieving higher maintainability, this framework is designed with 3 major layers: Page Object level, Test Case level, and the Data level.

Page Object level consists of all the page level elements as well as all common methods which can be called easily at the Test Script/Feature File level.

Application/logic changes can be accommodated in the automation framework by changing the common methods and the data files without changing the test scripts.

4.2 Usability

Cucumber integration with Selenium will achieve greater readability of the application logic covered by each Test Case and thereby the Users of the test suite can be easily understood what test that each Test Case would cover.

Also grouping of Test Cases/Feature Files under specific folders will help the users to manage Test Runs simply.

4.3 Re-Usability

A generic package (com.tpip.page) will be used to store classes with all common methods that are going to be used by different test cases of the test suite in order to inherit high re-usability.

New test cases can be designed by using these re-usable methods with a simple data change in the Data file with minimal effort.

4.4 Reliability

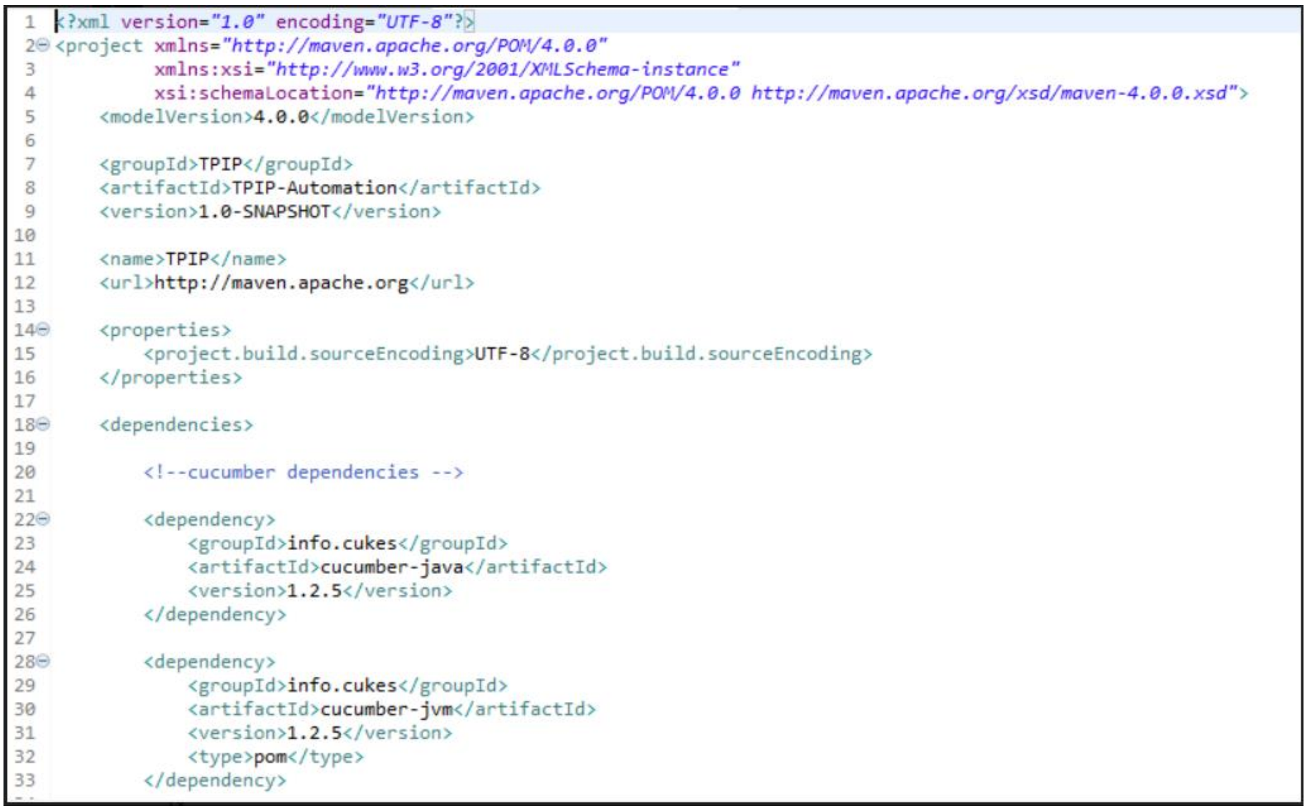
Loading time of some of the features in the **LGIM NavGuide** application is a concern such as time taking to generate PMC legal docs, time to load schemes in registration tab, etc. and this is one of the key issues which could cause unnecessary test failures. This framework is designed in a way to handle this situation by applying the below techniques.

1. *Explicit Wait command to direct the WebDriver to wait until a certain condition occurs before proceeding with executing the code.*
2. *Areas that cannot be handled via Explicit Wait, the standard wait command (e.g. Thread.sleep) has been used.*

4.5 *Portability*

Since the framework is integrated with Maven, the automated suite can be deployed in different machines with a minimal effort.

All dependencies are available in the 'pom.xml' file and these can be changed accordingly to upgrade the framework with any new versions of the dependent technologies whenever required.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>TPIP</groupId>
    <artifactId>TPIP-Automation</artifactId>
    <version>1.0-SNAPSHOT</version>

    <name>TPIP</name>
    <url>http://maven.apache.org</url>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>

        <!--cucumber dependencies -->

        <dependency>
            <groupId>info.cukes</groupId>
            <artifactId>cucumber-java</artifactId>
            <version>1.2.5</version>
        </dependency>

        <dependency>
            <groupId>info.cukes</groupId>
            <artifactId>cucumber-jvm</artifactId>
            <version>1.2.5</version>
            <type>pom</type>
        </dependency>
```

Figure 10: POM File

5 Environment

The test environment outlined in this section is necessary for the successful execution of test scripts. All hardware and software outlined in below sections is needed to be available prior to the test scripting / execution.

5.1 Design Time Environment

QA Application Environment used by the Manual Test Team has been used to automate the test cases.

- Application URL - http://user-p5u-tpip.net.s3-website.eu-west-2.amazonaws.com/

- Software/Hardware Requirements

No.	Machine / Device	Hardware	Software
1.	Desk Top Client	**RAM** – 8GB	**OS** – Windows 10 Pro
		Hard Disk – 100GB or Higher	**Browser** – Chrome
		Processor – Intel Core i7	**Build Tool** - Apache Maven 4.0.0
			Java – jdk-1.8
			IDE – Eclipse /
			Drivers – chromedriver

Table 4: Software and Hardware Requirements

5.2 Run Time Environment

Same environment mentioned above can be used in the run time environment as well.

6 Critical Success Factors

Following are some principles which are critical to the success of this Test Suite.

1. Resources responsible for running the scripts / creating the test suite for testing is required to know the steps of each test case and their dependencies (e.g. data files, test reports, etc.).
2. Test Data provided in the data file by the test team should be accurate and testable.
3. Having a good Internet Connection Speed is vital for a successful run of the automated test suite.
4. All major application Performance issues should be fixed prior to run the automated test suite.
5. All Critical / Show Stopper defects in the application should be fixed to avoid unnecessary test failures.
6. Data file needs to be updated with correct object identifiers / locators in case if object properties get changed with new developments.

Appendix

[Include additional information related to the Project that must be provided as part of this document.]