Table of Contents

[1. Introduction 2](#_Toc72385419)

[2. Automation Scope 2](#_Toc72385420)

[2.1 End to End Scenarios [Smoke] 2](#_Toc72385421)

[2.2 A set of most important tests [Regression] 3](#_Toc72385422)

[2.3 Test for the most important Modules 3](#_Toc72385423)

[3. Automation Solution 3](#_Toc72385424)

[3.1 Keyword-driven development 3](#_Toc72385425)

[3.2 Development architecture 5](#_Toc72385426)

[3.3 Development Tools 6](#_Toc72385427)

[4. Automation Framework 7](#_Toc72385428)

[4.1 Creation 7](#_Toc72385429)

[4.2 Execution 9](#_Toc72385430)

[4.3 Monitoring 9](#_Toc72385431)

[4.4 Reports 10](#_Toc72385432)

[4.5 Logs 10](#_Toc72385433)

[5. Continuous Integration 11](#_Toc72385434)

[6. Continuous Improvements 12](#_Toc72385435)

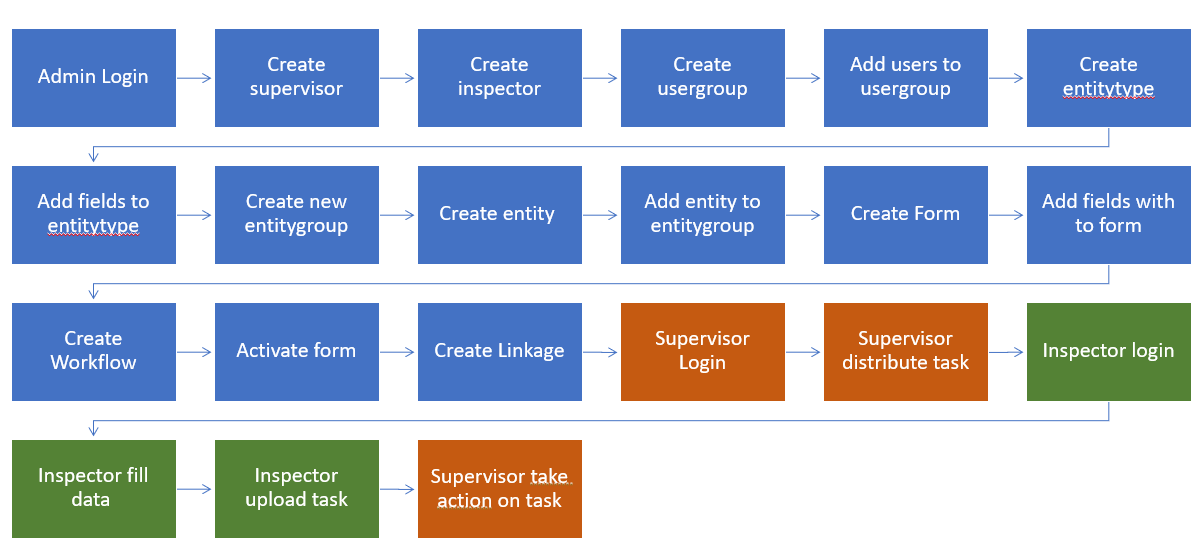
[7. Automation Artefacts 12](#_Toc72385436)

# Introduction

This document describes Test Automation Framework include Automation scope, Automation development solution, Automation Usage in creation and execution.

# Automation Scope

## End to End Scenarios [Smoke]



Example Bravo End to End scenarios

1. Admin Login with valid data
2. Admin Create supervisor with valid data
3. Admin Create inspector with valid data
4. Admin create usergroup with valid data
5. Admin add users (supervisor and inspector) to usergroup
6. Admin create new entitytype with valid data
7. Admin add field (textbox) to entitytype
8. Admin create new entitygroup with valid data
9. Admin create entity with valid data
10. Admin link entity to entitygroup and entitytype
11. Admin create new valid scheduled blank form
12. Admin add field (textinput) with default value to form
13. Admin save workflow 2 steps, 2 actors (Insp, Sup), 1 action every step triggered by step actor
14. Admin activate completed form
15. Admin add new linkage usergroup, entitygroup and form
16. Supervisor user login to system with valid data
17. Supervisor distribute task manually
18. Inspector login to mobile android app with valid data
19. Inspector fill valid data on task with one textinput field
20. Inspector upload task
21. ​​Supervisor take action on task

## A set of most important tests [Regression]

* Functionalities highly visible to users
* Cases which verify core features of the product
* Cases with frequent defects
* Integration Test cases

## Test for the most important Modules

* Task distribution
* Incident Tasks
* Visit Tracking (my Tasks)
* Mobile Task Management
* Workflow Builder
* Form Builder

# Automation Solution

## Keyword-driven development

In keyword-driven scripting there is only one control script and test definition contain a description of the tests easier to understand. It will usually contain data as does the data files, but keyword files also contain high level instructions (the keywords, or action keywords). Each keyword represents detailed interactions with the system under test. Sequences of keywords (including the relevant test data) are used to specify the test cases.

**Aspects of keyword-driven**

* Action keywords [Contain methods instructions represent each keyword referenced on Excel Sheet at each test step]
* Test cases [written on Excel Sheet]
* Test data [written on Excel Sheet]
* Test script [one control script that read test cases, test data, action keywords from Excel Sheet]
* Test reports [Generates report that show test results and monitoring it]

Diagram

Description automatically generated

Figure 1: Keyword-driven General

Diagram

Description automatically generated

Figure 2: Keyword-driven implemented, Use Excel Sheet to record test cases, test data and related actions and Use selenium and Appium in Test scripts and Generate pdf or html reports to show test results

## Development architecture

**ActionKeywords**

* AssertionActions.cs

The assertion methods like AssertElementContains, AssertElementExist, AssertUrlExist

* DriverActions.cs

The driver methods like OpenBrowser, OpenBrowser, RefreshBrowser, NavigateToURL, OpenMobileApp, CloseMobileApp

* ElementActions.cs

The element methods like Click, Input, Select, DragDrop, KeyPress

* WaitActions.cs

The wait methods like WaitUntil, WaitUntilExists, WaitUntilVisible, WaitSeconds

**Creation**

* BravoE2E.xlsx

Excel sheet used to create test suites , test cases and test data

* Locators.settings

Contains list of element locators of system under test like xpaths, IDs

**Execution**

* DriverScript.cs

The main script of keyword driven development

**Utilities**

* Dragdrop.js

JavaScript file that manage drag and drop action

* ExcelUtils.cs

Manage Excel like open workbook, read from worksheet and write in worksheet

* ExtentReporter.cs

Manage Extent reports to monitor and record test results

* Log.cs

Manage Logs to record test info, warn, error and fatal while execution

A picture containing text

Description automatically generated

Figure 3: Test automation project is a part of Bravo whole Solution

## Development Tools

**IDE/Coding**

* Visual studio
* IntelliJ IDEA
* C#
* Java
* JavaScript

**Framework**

* Nunit
* TestNG
* Selenium
* Appium

**Mobile/Emulators**

* Android studio
* Android SDK Manager
* AVD Manager

**Utilities**

* Excel
* Extent Reports
* Katalon

**Version control**

* Git

# Automation Framework

## Creation

A screenshot of a computer

Description automatically generated with low confidence

Figure 4: Create test suite on Excel contain IDs, titles of test cases

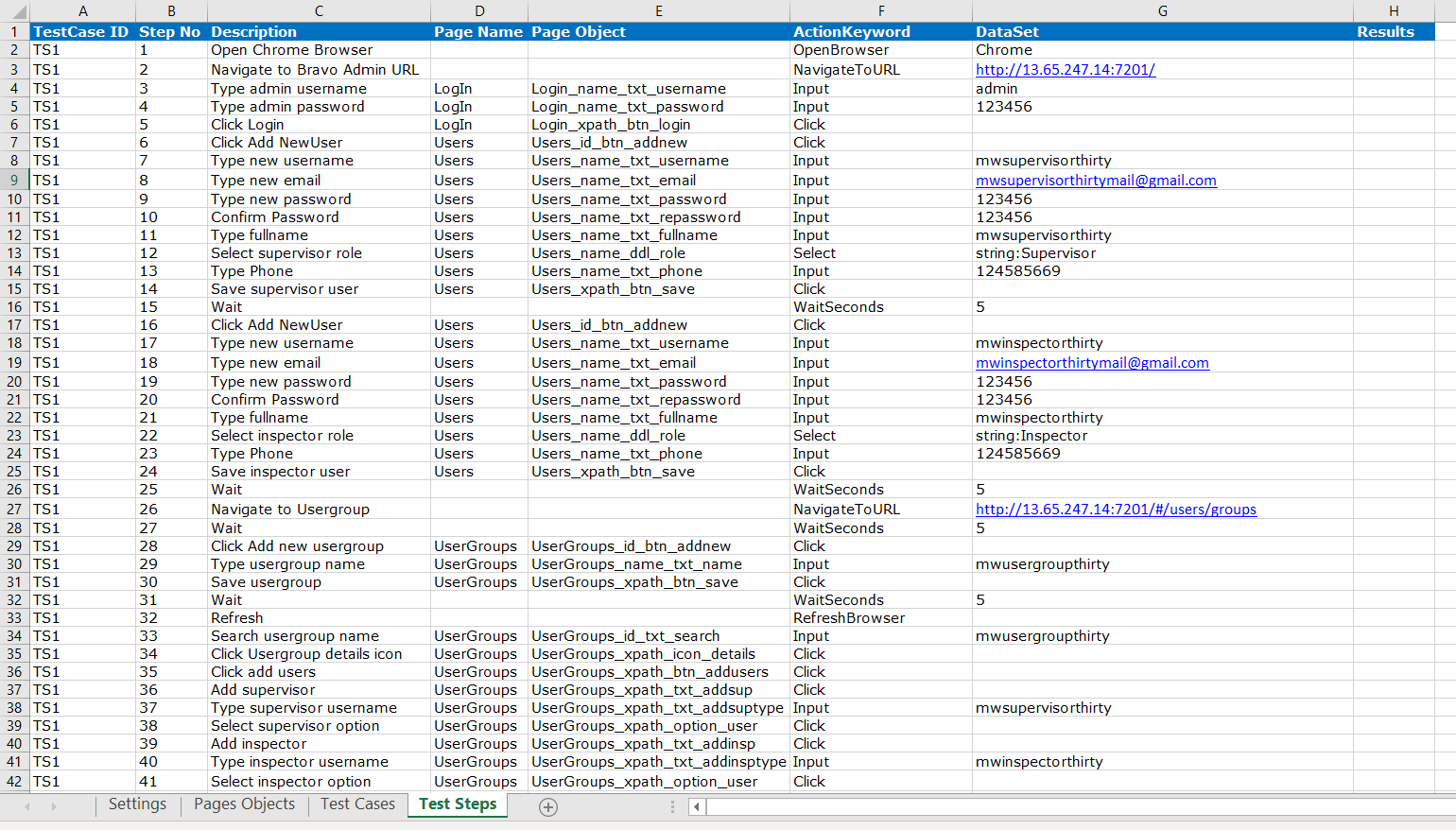


Figure 5: Create Test steps of each test case on Excel Sheet

How to add page objects and locators

Graphical user interface, application

Description automatically generated

Figure 6: Use Application Settings on visual studio automation project to create element locators and page objects

## Execution

A picture containing text, monitor, screenshot, black

Description automatically generated

Figure 7: Execution from Test Explorer on Visual studio automation project

## Monitoring

Graphical user interface, application

Description automatically generated

Figure : Monitoring test results

## Reports

Graphical user interface, text, application, email

Description automatically generated

Figure : Report test results details

## Logs

A picture containing text

Description automatically generated

Figure : Logging test execution

# Continuous Integration

Create new Build and link to repository and branch

Graphical user interface, application

Description automatically generated

Add E2E visual studio test task

Graphical user interface, application, email

Description automatically generated

Queue new build

Graphical user interface, text, application, email

Description automatically generated

# Continuous Improvements

|  |  |  |
| --- | --- | --- |
| Task | Category | State |
| Keyword driven Framework | Framework Code | Done |
| Creation and Execution by Excel | Framework Code | Done |
| End to End test [Bravo] | Test script | Done |
| A set of most important tests [Bravo] | Test script | To Do |
| Modules Tests [Bravo] | Test script | To Do |
| Continuous Integration configuration | Continuous Integration | In Progress |
| Dynamic test data using DB | Framework Code | To Do |
| Using Separate server for CI | Continuous Integration | In Progress |
| Run test framework in All OS platforms | Framework Code | To Do |
| MAC POC to test iOS IPA | Framework Code | To Do |
| Resistant to UI changes [Fetch Elements Locators] | Framework Code | In Progress |
| Support Behavior Driven Development | Framework Code | In Progress |
| Using Container [Docker] | Continuous Integration | In Progress |
| Support Sending Mails | Continuous Integration | To Do |
| Creation and Execution By GUI | Framework Code | To Do |
| Add scheduler for tests | Framework Code | To Do |
| Add Metrics for test analysis | Framework Code | To Do |

# Automation Artefacts

Test Automation deliveries contains:

* Test Automation overview Presentation
* Test Automation overview Document
* Test Automation Source code
* Test Automation Guide videos