# **Test Automation Framework Documentation**

**1. Introduction**

**1.1 Purpose**

* The purpose of this Test Automation Framework is to provide a structured and efficient approach for automating the testing of UI and API Testing. The framework aims to improve the speed, reliability, and coverage of the testing process, ensuring high-quality software delivery with minimal human intervention.

**1.2 Scope**

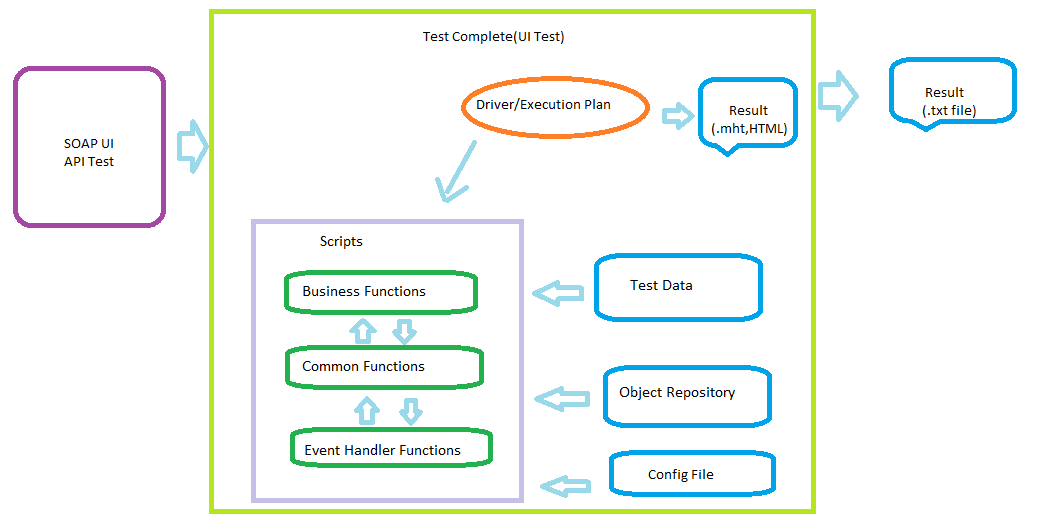
* The scope of this Test Automation Framework encompasses the automation of functional, regression, and sanity testing for the UI testing and to test API testing as well. It defines the boundaries and limitations of the test automation efforts and outlines what will and will not be covered under this framework.

1. **Test Types(UI)**:
   * **Functional Testing**: Automating test cases to verify that the core functionality of the application is working as expected.
   * **Regression Testing**: Automating a suite of test cases to ensure that recent code changes do not introduce regressions.
   * **Smoke Testing**: Automating basic tests to validate the primary functionality of the system after each deployment.
2. **Supported Platforms & Browsers**:
   * Web application testing across popular browsers, including Chrome, Firefox, and Edge.
   * Mobile application testing on Android and iOS devices (if applicable).
   * API Testing
3. **Integration with CI/CD Pipeline**:
   * Integration with the existing Continuous Integration/Continuous Deployment (CI/CD) tools (e.g., Jenkins, DevOps CI) to trigger automated tests on every code commit.
   * Generation of test reports (HTML, notepad) post-execution and storing them in a central location for analysis.
4. **Test Data Management**:
   * Test data comes from external excel file.
   * Option to pass both positive and negative test data
   * One can change the test data without knowing the coding and framework.
5. **Error and Exceptional Handling:**

* Error handling is implemented using tool in-build Event handler to handle unexpected and overlapping windows. And exceptional handling is implemented using try catch method in script.

**2. Architecture Overview**

**2.1 High-Level Architecture**

**2.2 Framework Components**

* **Tools**: Soap UI(SoapUI-5.8.0) Open source and Test Complete(15.73.16.7 x64) 14 days trial version with Desktop, Web and Mobile extensions.
* **Scripting Language**: JavaScript.
* **Test Scripts**: This includes Business level functions, Common functions and Event handler functions.
* **Test Data:** Test data comes from Excel file. For different modules different sheets are maintained.
* **Test Runner**: we can execute test cases using Test Complete(using execution plan),TestExecute. For command line execution please refer: <https://support.smartbear.com/testcomplete/docs/working-with/automating/command-line-and-exit-codes/command-line.html>
* **Reporting:** There are 2 types of reporting. 1) Testcomplete report (.mht or html file) for detailed report and analysis. 2) Text file. This gives high level report like test case name and pass or fail.
* **Configuration File**: This file contains project suite level data like application URL, Browser Type (like Chrome, Edge, Firefox) etc.

**3. Setup and Installation**

**3.1 Prerequisites**

* **Test Complete**: System with Test Complete tool with version 15.73.16.7 or later version with extensions Web and Desktop should be installed. If you are using later version of 15.73.16.7 then migrate project to latest version installed. For migration please refer: <https://support.smartbear.com/testcomplete/docs/general-info/migrating.html> .
* **SoapUI** : SoapUI(version 5.8.0) should be installed under : C:\Program Files\SmartBear
* **Operating System**: Windows 7 with Service Pack 1 or later version or Vista or Server 2012 or Server 2012 R2 or Server 2016 or Server 2008 or Server 2008 R2
* **Hardware**: Intel Core 2 Duo 2 GHz or higher. Minimum 2GB RAM, Minimum 200 MB of free space on the system disk, 1024 × 800 or higher resolution monitor
* **Internet**: System should be connected to internet to access application.

**3.2 Installation Steps**

* Download or copy “TestAutomation” folder from <https://github.com/DayanandBhat/TestingRepo> to local drive.

**4. Framework Usage**

**4.1 UI Test Execution**

* **Test Cases Automated:** Test case are written for one banking application i.e. <https://parabank.parasoft.com/> . Two test cases added as of now 1) User Registration for positive test case. 2) User Registration for negative case.
* **Running Tests Locally**:. Open Test Complete and open Test Suite “ProjectSuiteAPIandUI.pjs” (which is under “TestAutomation”) from File Menu.
* **Run UI Test Case**: Go to UI Project right and click on “Run UI Project”



* **Run API Test Case**: Go to APIProject right and click on “Run API Project”



**4.3 Test Reporting**

* **UI Test**: There are two types of reporting
  1. **Test Complete report**: This is detailed report which helps to analysis each step and for debugging purposes if any test cases fails. This report is in the form of .mht, HTML or XML format. We can export/save this report in a shared repository in CI/CD implementation.

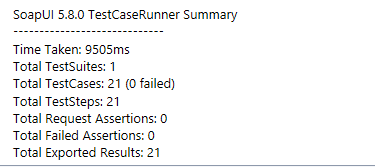
Extract above Result.rar and open index.hmt file to view

Sample result.

* 1. **External File:** This is simple text file generated and updated during test execution. This gives high level information like Test case name and status pass or fail and execution date time. Path of this file is mentioned in config file.

****

* **API Test:** As of now we have Test complete report. We can see the detailed report in Testcomplete report Detail section

****

**5. Best Practices**

**5.1 Test Design Guidelines**

* Re-usable code. Ex. Common Functions and Business Functions. We can use these functions in driver script.
* Followed coding standards