|  |  |
| --- | --- |
| **Project** | DemoAutomationKatalon |
| **Document Title** | Project Document |

**DemoAutomationKatalon**

**POM Based Data Driven Web UI Automation testing using Katalon Studio**

[**https://docs.katalon.com/katalon-studio/docs/index.html**](https://docs.katalon.com/katalon-studio/docs/index.html)

***An All-in-one test automation solution!***

***Go low-code. Boost productivity. Ease Scalability!*** [***https://www.katalon.com/katalon-studio/***](https://www.katalon.com/katalon-studio/)

**INDEX**

**Topic Page**

*Introduction to Automation testing 3*

*Benefits of Automation Testing 3*

*Automation workflow for the application can be presented as follows 3*

*Environment Specifications 3*

*Framework & Design used in this Project 4 to 5*

*Tools Used in this Framework (Katalon Studio) 6*

*The DemoAutomationKatalon Explanation in detail 7*

*Design Diagram of the DemoAutomationKatalon 8*

*Installation & Configuration Steps of the tools used in this project 9*

*Creation & Configuration Steps of this project 10*

*How to run test scripts and view reports 11*

*Test Execution Lifecycle 11*

**Introduction to Automation testing:**

Testing is an essential part of a software development process. While testing intermediate versions of products/projects being developed, testing team needs to execute a number of test cases. In addition, prior to release every new version, it is mandatory that the version is passed through a set of “regression” and “smoke” tests. Most of all such tests are standard for every new version of product/project, and therefore can be automated in order to save human resources and time for executing them.

**Benefits of Automation Testing:**

* Reduction of test’s time execution and human resources required.
* Complete control over the tests’ results (“actual results” vs “expected results”)
* Possibility to quickly change test’s preconditions and input data, and re-run the tests dynamically with multiple sets of data

**Automation workflow for the application can be presented as follows:**

* First of all, it is required to identify tasks that an application has to accomplish.
* Second, a set of necessary input data has to be created.
* Third, expected results have to be defined in order one can judge that an application (a requested feature) works correspondingly.
* Fourth, Executes a test.
* Finally, compares expected results with actual results, and decides whether the test has been passed successfully.

**Framework & Design used in this Project:**

1. **Page Object Model:**

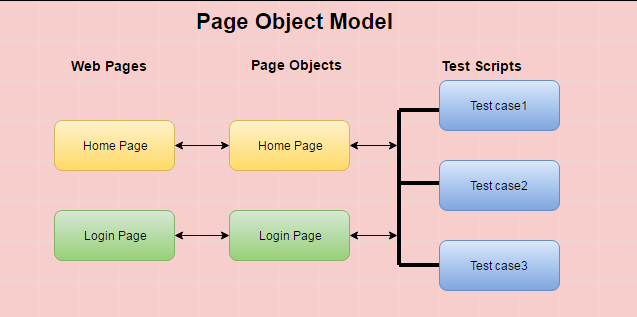
Page Object Model, also known as POM, is a design pattern that creates an object repository for storing all web elements. It is useful in reducing code duplication and improves test case maintenance.

In Page Object Model, consider each web page of an application as a class file. Each class file will contain only corresponding web page elements. Using these elements, testers can perform operations on the website under test.

**Advantages of Page Object Model:**

* + Helps with easy maintenance.
  + Helps with reusing code.
  + Readability and Reliability of scripts.

**Workflow Diagram of Page Object Model:**



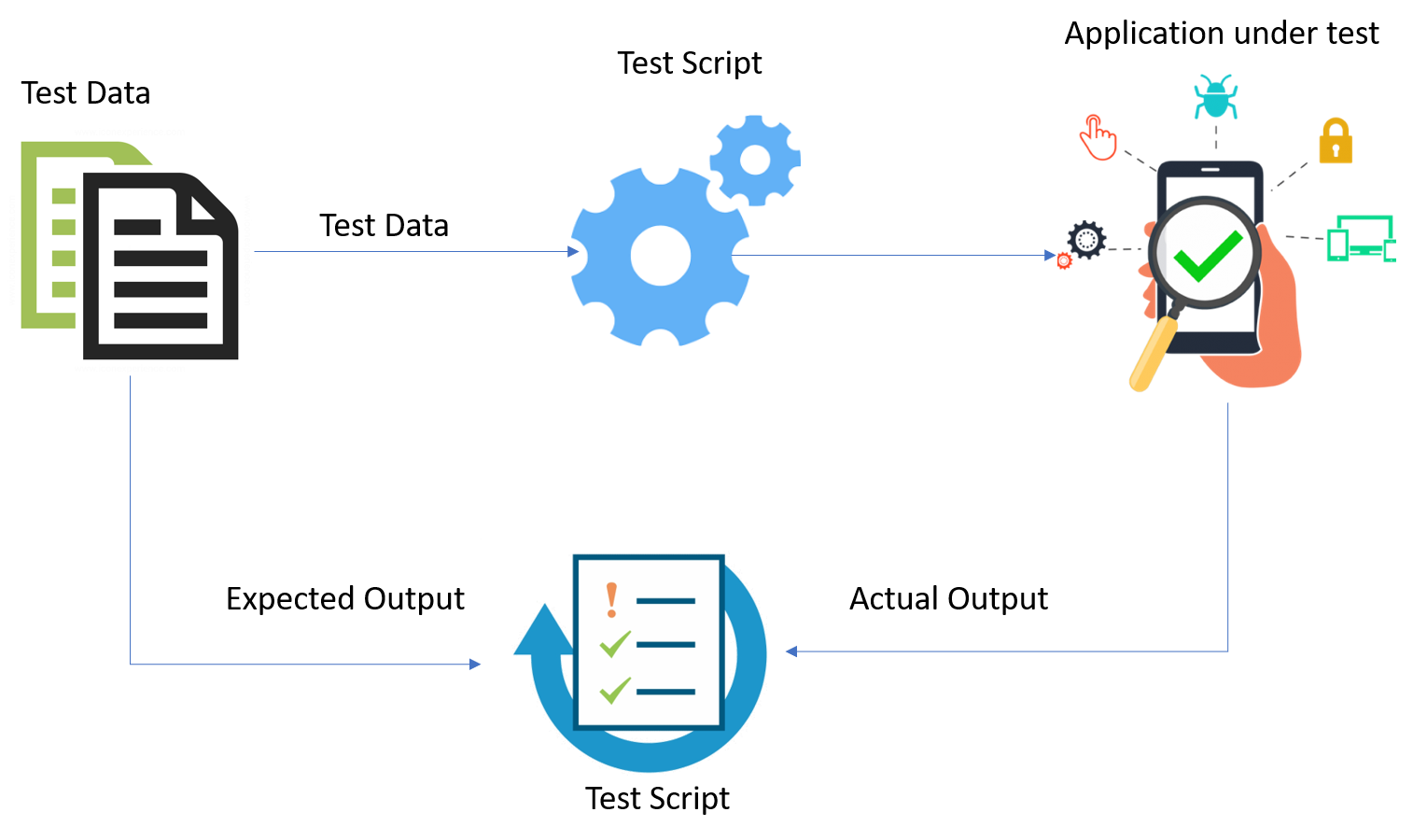
1. **Data Driven Framework:**

Data Driven framework is used to drive test cases and suites from an external data feed. The data feed can be data sheets like xls, xlsx, and csv files, Databases.

A Data Driven Framework is a technique of separating the “data set” from the actual “test case” (code). Since the test case is separated from the data set, one can easily modify the test case of a particular functionality without making changes to the code.

**Advantages of DataDriven Framework:**

* Test cases can be modified without much changes to code.
* It allows testing the application with multiple sets of data values, especially during regression testing.
* It helps us to separate the logic of the test cases/scripts from the test data.

**Workflow Diagram of Data Driven Approach:**

**Tools Used in this Framework (Katalon Studio) :**

**Katalon Studio:**

Katalon Studio is an automation testing software tool developed by Katalon, Inc. The software is built on top of the [open-source](https://en.wikipedia.org/wiki/Open-source_software) automation frameworks [Selenium](https://en.wikipedia.org/wiki/Selenium_(software)), [Appium](https://en.wikipedia.org/wiki/Appium" \o "Appium) with a specialized [IDE](https://en.wikipedia.org/wiki/Integrated_development_environment) interface for [web](https://en.wikipedia.org/wiki/Web_testing), [API](https://en.wikipedia.org/wiki/API_testing), [mobile](https://en.wikipedia.org/wiki/Mobile_application_testing) and desktop application testing.

Katalon Studio provides a dual interchangeable interface for creating test cases: a manual view for the less technical users and a script view gearing toward experienced testers to author automation tests with [syntax highlight](https://en.wikipedia.org/wiki/Syntax_highlighting) and [intelligent code completion](https://en.wikipedia.org/wiki/Intelligent_code_completion). It follows the Page Object Model pattern. [GUI](https://en.wikipedia.org/wiki/Graphical_user_interface) elements on web, mobile, and desktop apps can be captured using the recording utility and stored into the Object Repository, which is accessible and reusable across different test cases.

**Technologies:**

The [test automation framework](https://en.wikipedia.org/wiki/Test_automation) provided within Katalon Studio was developed with the [keyword-driven](https://en.wikipedia.org/wiki/Keyword-driven_testing) approach as the primary test authoring method with [data-driven](https://en.wikipedia.org/wiki/Data-driven_programming) functionality for test execution.

The user interface is a complete [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) implemented on Eclipse [rich client platform](https://en.wikipedia.org/wiki/Rich_client_platform) (RCP).

The keyword libraries are a composition of common actions for web, API, and mobile testings. External libraries written in [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) can be imported into a project to be used as native functions.

The main programming language used in Katalon Studio are [Groovy](https://en.wikipedia.org/wiki/Apache_Groovy) and [.](https://en.wikipedia.org/wiki/Java_(programming_language)) Katalon Studio supports cross-environment test executions based on Selenium and Appium.

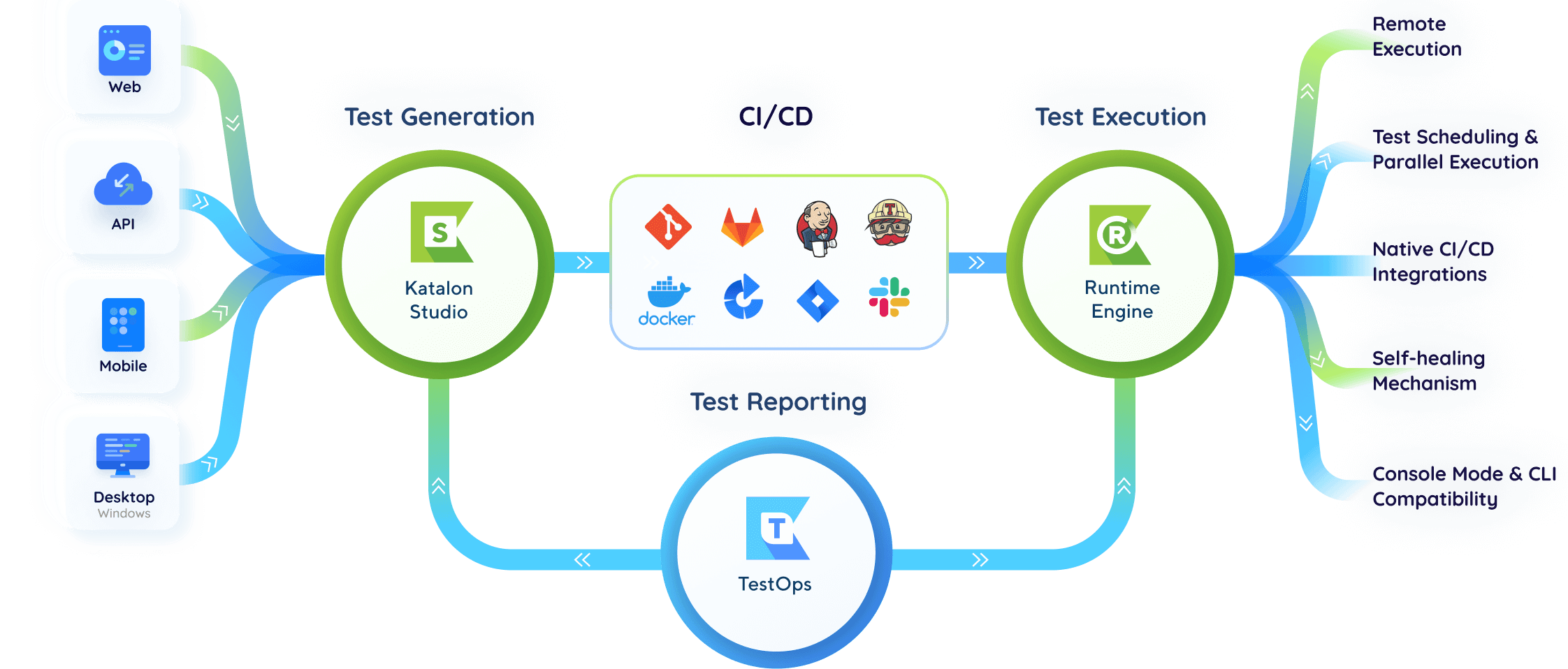
**Supported Technologies:**

Modern web technologies: HTML, HTML5, JavaScript, Ajax, Angular

Windows desktop apps platforms: Universal Windows Platform (UWP), Windows Forms (WinForms), Windows Presentation Foundation (WPF), and Classic Windows (Win32)

Cross-browser testing: Firefox, Chrome, Microsoft Edge, Internet Explorer (9,10,11), Safari, headless browsers.

Mobile apps: Android and iOS (Native apps and mobile web apps)

Web services: RESTful and SOAP

**The DemoAutomationKatalon Explanation in detail:**

**1. Object Repository:**

UIMap is a concept for defining, storing, and serving UI elements of an application or a website. The UIMap properties file contains a set of ‘key-value’ pairs, where key is an alias of the UI element, and a value is the locator.

**2. Test Data/Data files:**

Data set stores the data files, Script reads test data from external data sources and executes test based on it. Data sets increases test coverage by performing testing with various inputs and reduce the number of overall test scripts needed to implement all the test cases.

**3. Test Automation Scripts/Test Cases:**

A test is considered as a single action or a sequence of actions, that defines whether a specific feature meets functional requirements. It has multiple test files / packages / class files which will be executed based on the configurations defined in testng.xml.

**4. Reports / Executed Results:**

Test report/results is a document which contains summary of test activities. After execution is completed, it is very important to communicate the test results and findings to the project manager along with the [screenshots for failed tests](http://seleniumeasy.com/testng-tutorials/how-to-take-screenshot-for-only-failed-tests-using-webdriver) and with that decisions can be made for the release.

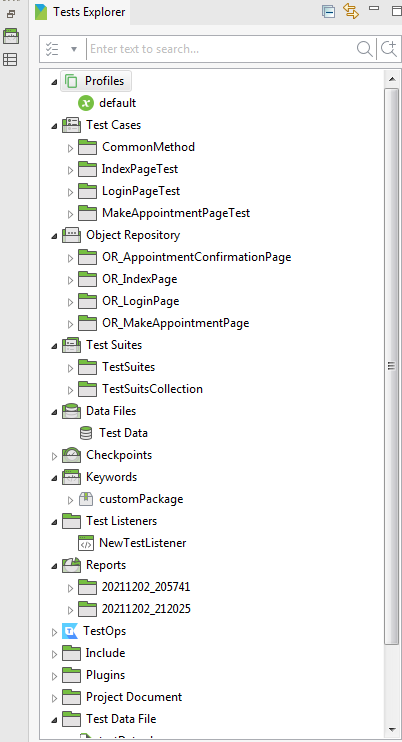
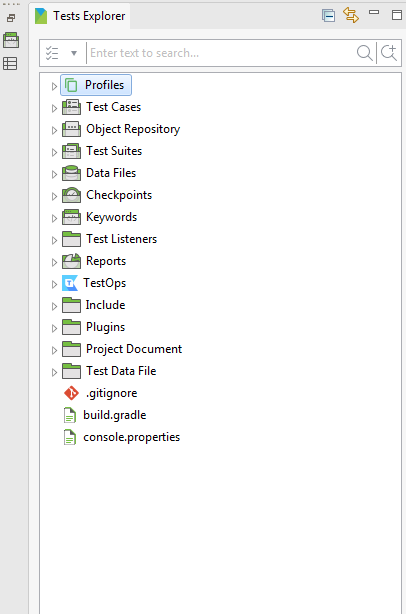
**5. Test Listeners:**

**Test Listeners** are test steps created based on your own criterion. Therefore, it's a useful and flexible way to help you extend your current testing flows. When matching the condition, you can execute Test Listeners.

**6. Keywords:**

In addition to built-in keywords, you can define custom keywords to extend the capabilities of Katalon Studio. Once created, custom keywords can be used when implementing test cases, just like other built-in keywords.

**Design Diagram of the DemoAutomationKatalon:**



**Installation & Configuration Steps of the tools used in this project:**

**Install Katalon Studio on macOS/Windows :**

**Requirements:**

* + - A valid email to register a Katalon Account
    - An active Internet connection to download Katalon Studio

**Note:** From Katalon Studio version 7.9.1 onwards, we only support 64-bit Windows, macOS, and Linux.

**Download Katalon Studio:**

**Step 1) Go to the Katalon Studio Website:** [**https://www.katalon.com/download/**](https://www.katalon.com/download/)

**Step 2)** Register a Katalon account with an active email or sign in to your Katalon account if you already have one.

**Step 3)** To download Katalon Studio, click **Create your first test** in the **Katalon Studio** box. Katalon will lead you to the downloading page where it automatically detects and downloads a suitable version for your system.

**Step 4)** You can also select a preferred version for your system on the page. Then Download.

**Activate Katalon Studio:**

**Step 1)** The Katalon Studio app launches, then the **Katalon Studio Activation** dialog appears. Sign in to your Katalon account to activate your license.

**Step 2)** After you are done, click Activate. Now Katalon Studio is ready for use.

**Creation & Configuration Steps of this project:**

**Creation Steps of this project:**

**Step 1)** Select **File > New > Project** from the main menu. In the displayed **New Project** dialog, choose a desired project **type**.

**Step 2)** Specify **Name**, **Location**, and **Description** for the new project and click **OK**.

**Step 3)** Katalon Will generate project accordingly.

**DemoAutomationKatalon open Steps:**

**Step 1)** Select **File > Open Project** from the menu. Browse to the folder where your project is located and select it.

**Step 2)** You can also quickly open recent test project by selecting from a list displayed under the **File** menu.

**How to Run test scripts and view Reports:**

**Step 1)** Open DemoAutomationKatalon test suit you want to execute.

**Step 2)** Click on the icon to execute test suite.

**Step 3)** Observe the Test suite execution and logs.

**Step 4)** Once Test suite execution gets finished you can see the summary result on the screen.

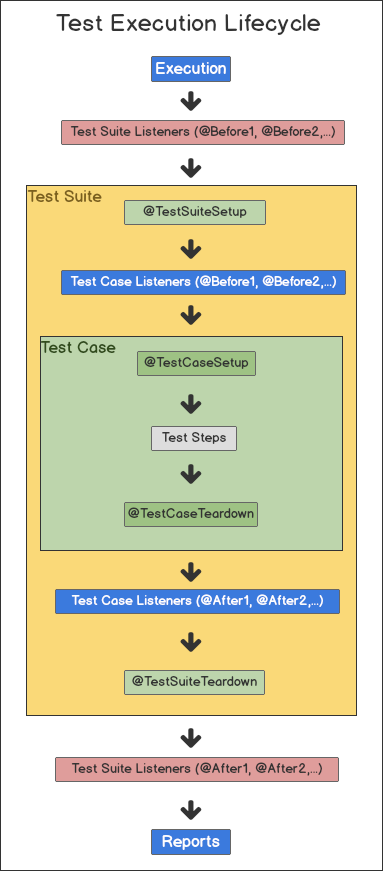
**Step 5)** Navigate to the Reports folder in the same project directory.

**Step 6)** Choose your test suite and open it.

**Step 7)** You can see reports (HTML,PDF,CSV as setting done) file here for detailed reports.

**Step 8)** You can also see the project level detailed report here: <https://analytics.katalon.com/>

**Test Execution Life Cycle:**



*Thank you!*