Test Framework for Plooto Web Portal

Prepared by:

Jerald Jose

[Jerald.jose@outlook.com](mailto:Jerald.jose@outlook.com)

Contents

[1. Introduction 3](#_Toc90303798)

[2. Framework design 3](#_Toc90303799)

[2.1. Platform 3](#_Toc90303800)

[2.2 A Simple Test Case Broken Down 3](#_Toc90303801)

[3. Setup and configuration 4](#_Toc90303802)

[3.1 Pre requisite for framework 4](#_Toc90303803)

[3.2 Pre requisite for Jenkins integration 4](#_Toc90303804)

[3.3 Code Repo 4](#_Toc90303805)

[4. Test Execution 4](#_Toc90303806)

[4.1 Run test locally 4](#_Toc90303807)

[4.2 Run test in Jenkins 5](#_Toc90303808)

[4.3 List of Available Test Cases 6](#_Toc90303809)

[4.4 Test Results – Sample from Jenkins Dashboard 6](#_Toc90303810)

[5. Future Enhancements 9](#_Toc90303811)

# Introduction

This document contains the details of the test framework which has been created to support the front-end validation of the static clone site of Plooto web application

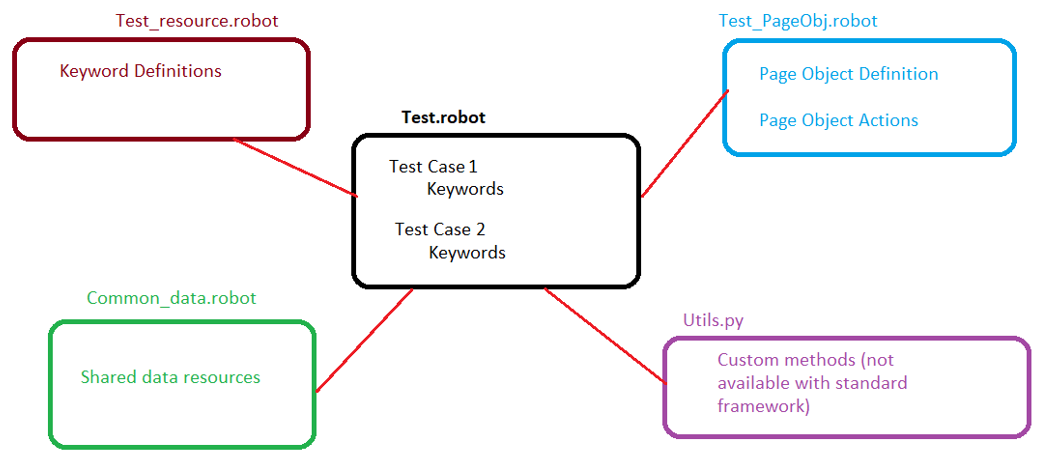
# Framework design

## Platform

The automation test framework is built as a Keyword Driven Framework using Python3, Robot Framework and Selenium2Library.

* **Python3** – Python3 is the backbone of this framework. The Robot Framework libraries are installed on the Python3 platform. This enables the framework to scale beyond the scope of the Robot Framework. Features or functionalities which are not available in Robot Framework can easily be added to our framework by adding custom libraries.
* **Robot Framework** – Robot Framework is an open-source and extensible tool which can be used for various Test Automation. Robot Framework has a syntax which is simple and easy to read.
* **Selenium2Library** – This is the Web UI testing library for Robot Framework. Selenium2Library convert all the standard Selenium use cases/actions to Keyword format and can be readily used in Robot Framework tests
* **Keyword Driven Tests** – By following a Keyword driven approach, we define a test case as a bunch of keywords. The keyword represent a test step in the testing process. This enhances the readability of the tests also eases the effort to write newer test cases.
* **Page Object Model** – By implementing Page Object Model approach, we maintain a repository of Web Elements for each page separately. This will tremendously improves the maintainability of the framework

## 2.2 A Simple Test Case Broken Down



Test Cases which are functionally related are put into a .Robot file which will act as the Test Suite. Each suite will have a corresponding .Robot resource file which will carry the Keyword definitions used by the Tests in the Test Suite. In addition to the resource file, the Test Suite also refer to Page Object robot files of various web pages which carry the Web Element definition (like Xpath, ID, value of a label) and it action definitions (like a Click or text entry). Also a Common\_Data.robot file is available for the Test suite to refer where shared/common Keywords are kept.

In addition to all the above, framework also provide an option to write custom python libraries which can be used to perform various actions which are not readily available in the standard libraries.

# Setup and configuration

## 3.1 Pre requisite for framework

The below listed should be installed for setting up and run the test framework

* Python 3
  + Python 3 can be installed from official website www.python.org
* RobotFramework library for Python
  + Once python is installed, use the command *pip install robotframework* to install the Robot Framework
* Selenuim2Library
  + Selenium2Library can be installed using command *pip install robotframework-selenium2library*
* Chrome Web Driver
  + Download and copy the Chrome web driver to the Python path. The available versions can be found at <https://chromedriver.chromium.org/downloads>. Please make sure the Chrome browser and web driver version are matching/compatible.

## 3.2 Pre requisite for Jenkins integration

The test framework is ready to be plugged into Jenkins. If you are planning to use Jenkins, please install the plugin Robot Framework in Jenkins in addition to items mentioned in 3.1

## 3.3 Code Repo

The framework code is maintained in the github.com. Please use the below links to access the code repo.

* URL - <https://github.com/josejerald/plootochallenge>
* Git checkout URL - <https://github.com/josejerald/plootochallenge.git>

Note – This is kept as a public repo for easy access. So no credential is needed to access or use the repo

# Test Execution

## 4.1 Run test locally

To run the test locally, pull the code from GitHub repo (refer step 3.3) and run the below command to execute the tests

Run the complete tests:

***python –m robot “{Path to framework root folder}\TestCases”***

Run the test with tag “Smoke”:

***python –m robot –include Smoke “{Path to framework root folder}\TestCases”***

Run a specific test:

***python –m robot –t “<TestCase Title>” “<Path to framework root folder>\TestCases”***

## 4.2 Run test in Jenkins

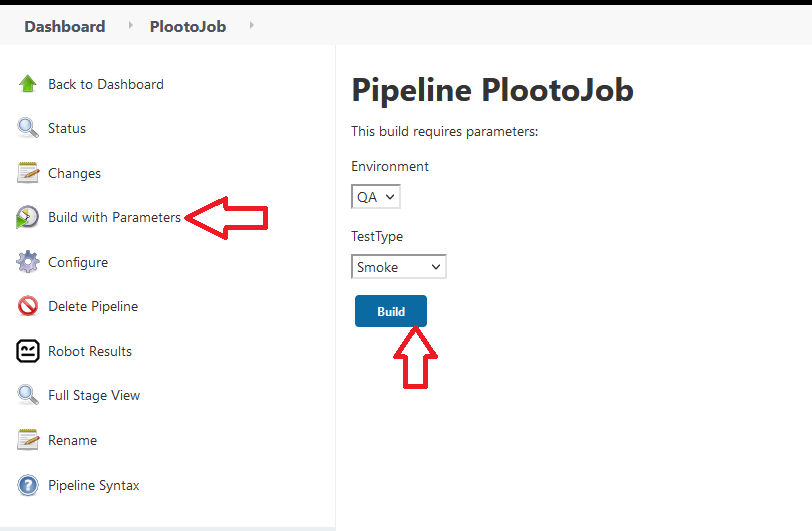
Create a Jenkins job by importing the below xml file using Jenkins-CLI



CLI command to create the Job:

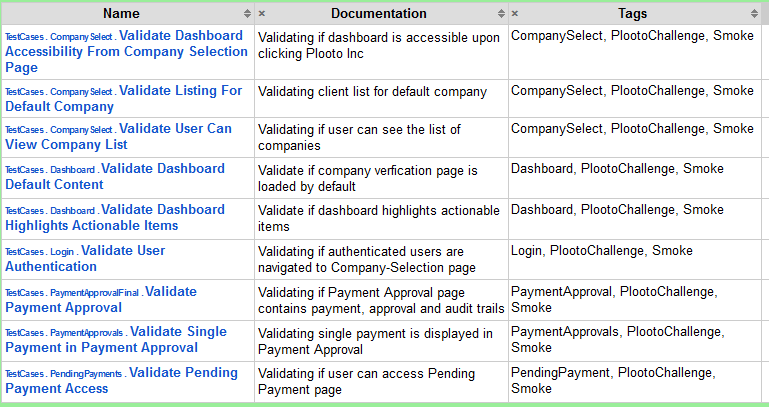
***java -jar jenkins-cli.jar -s {****server:port****} auth {****user****}:{****password****} create-job {****jobname****} < {xmlfile}.xml***

Once the Job is created, simply run the job using Build with Parameters option.



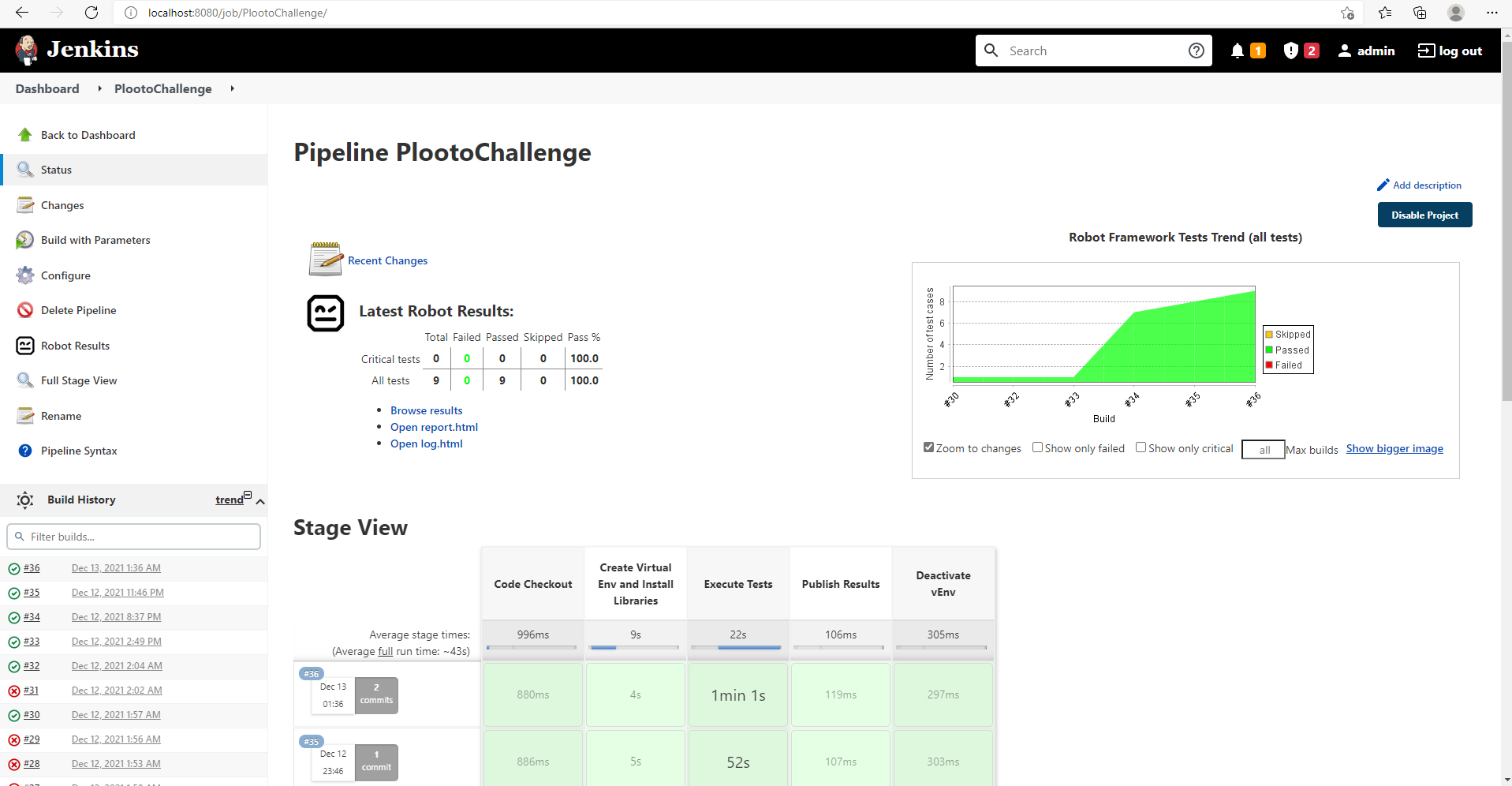
## 4.3 List of Available Test Cases

* The below given are the available test cases in the framework

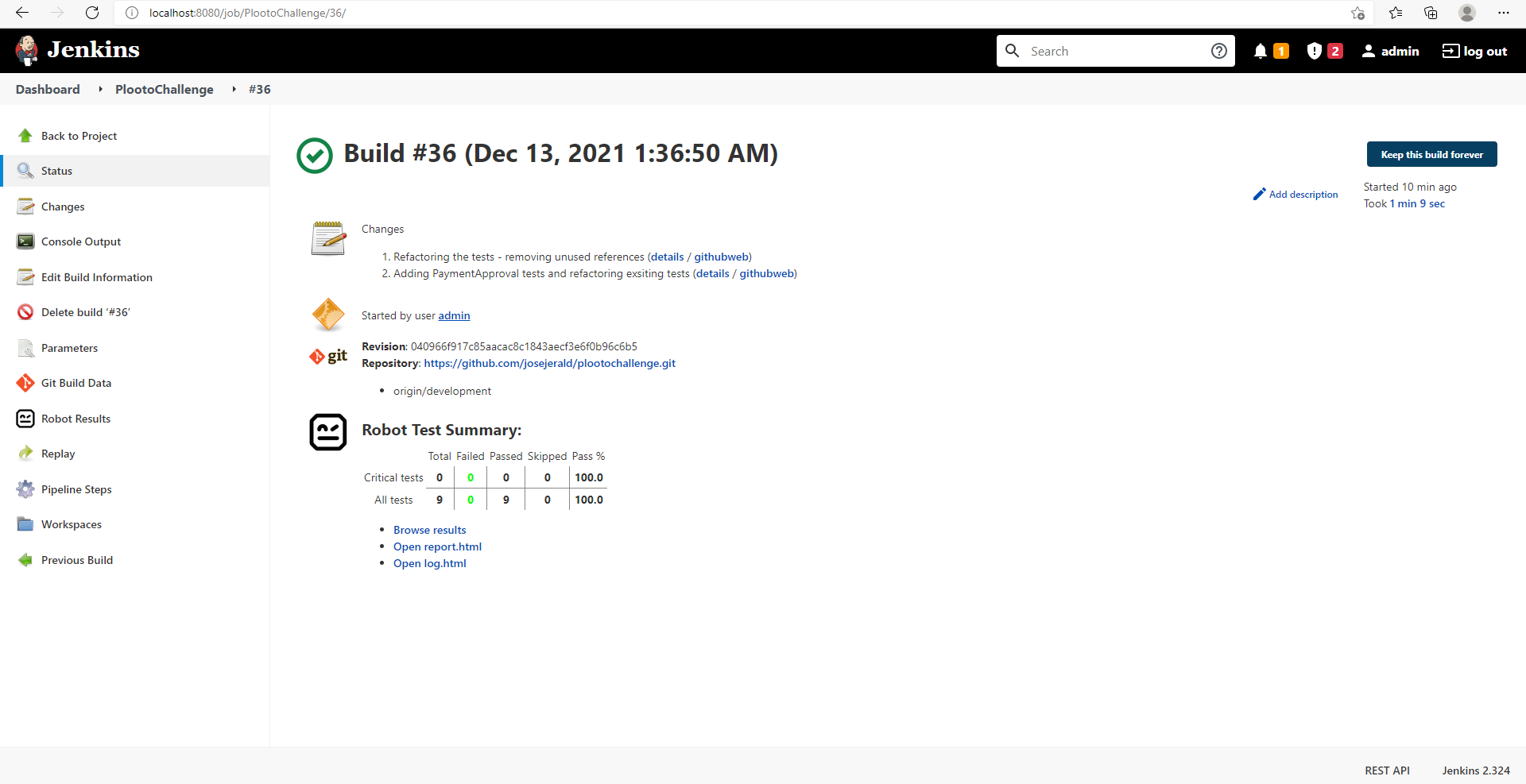


## 4.4 Test Results – Sample from Jenkins Dashboard

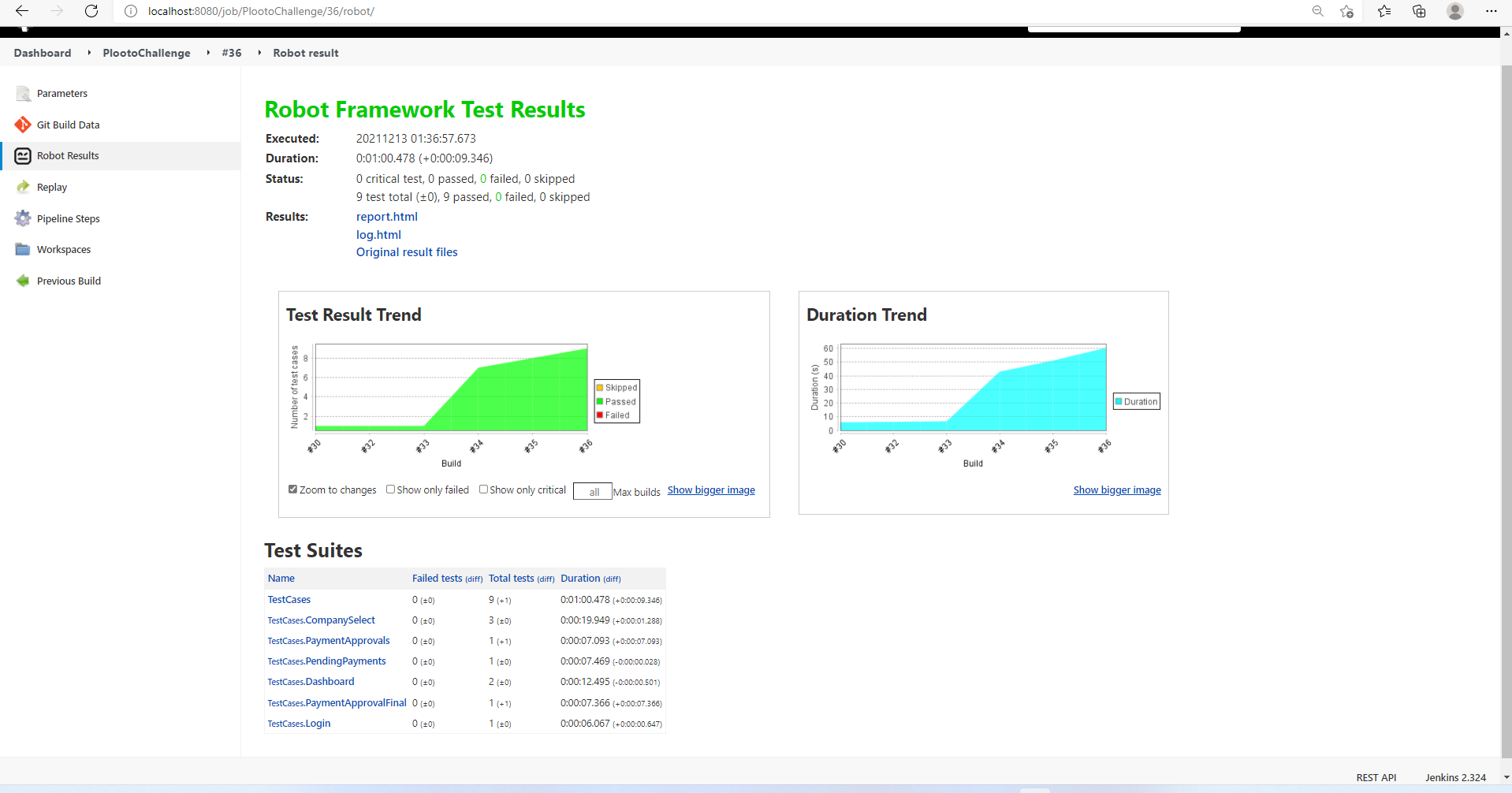
* Jenkins Job Dashboard



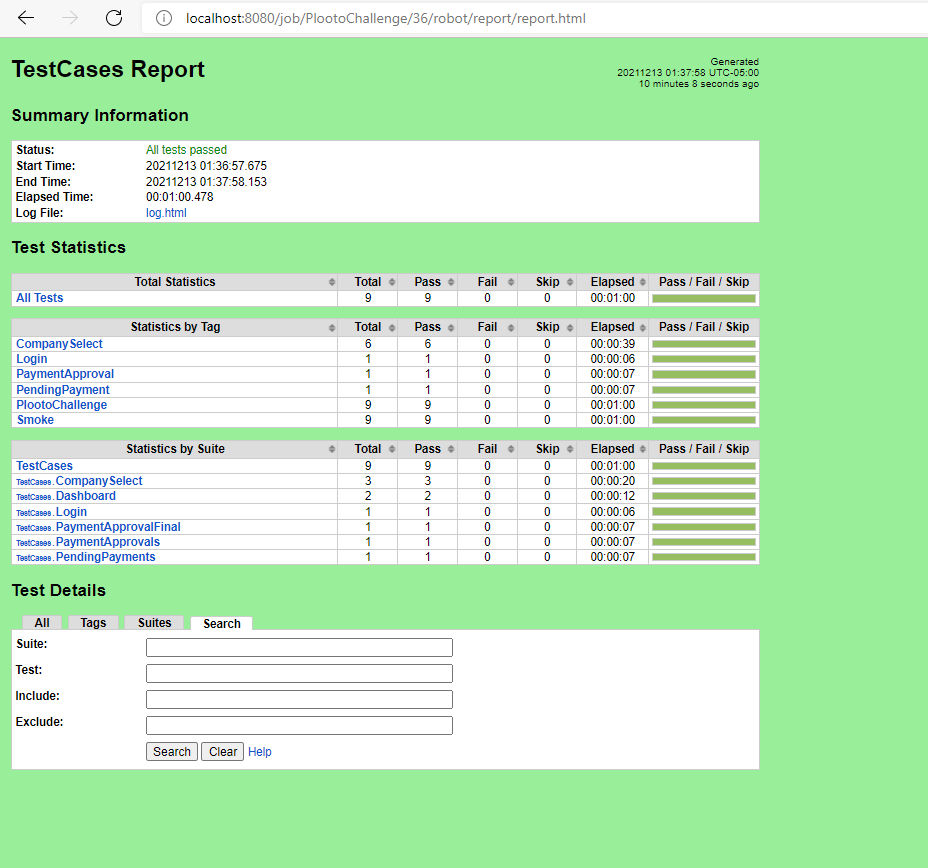
* Jenkins Job Details



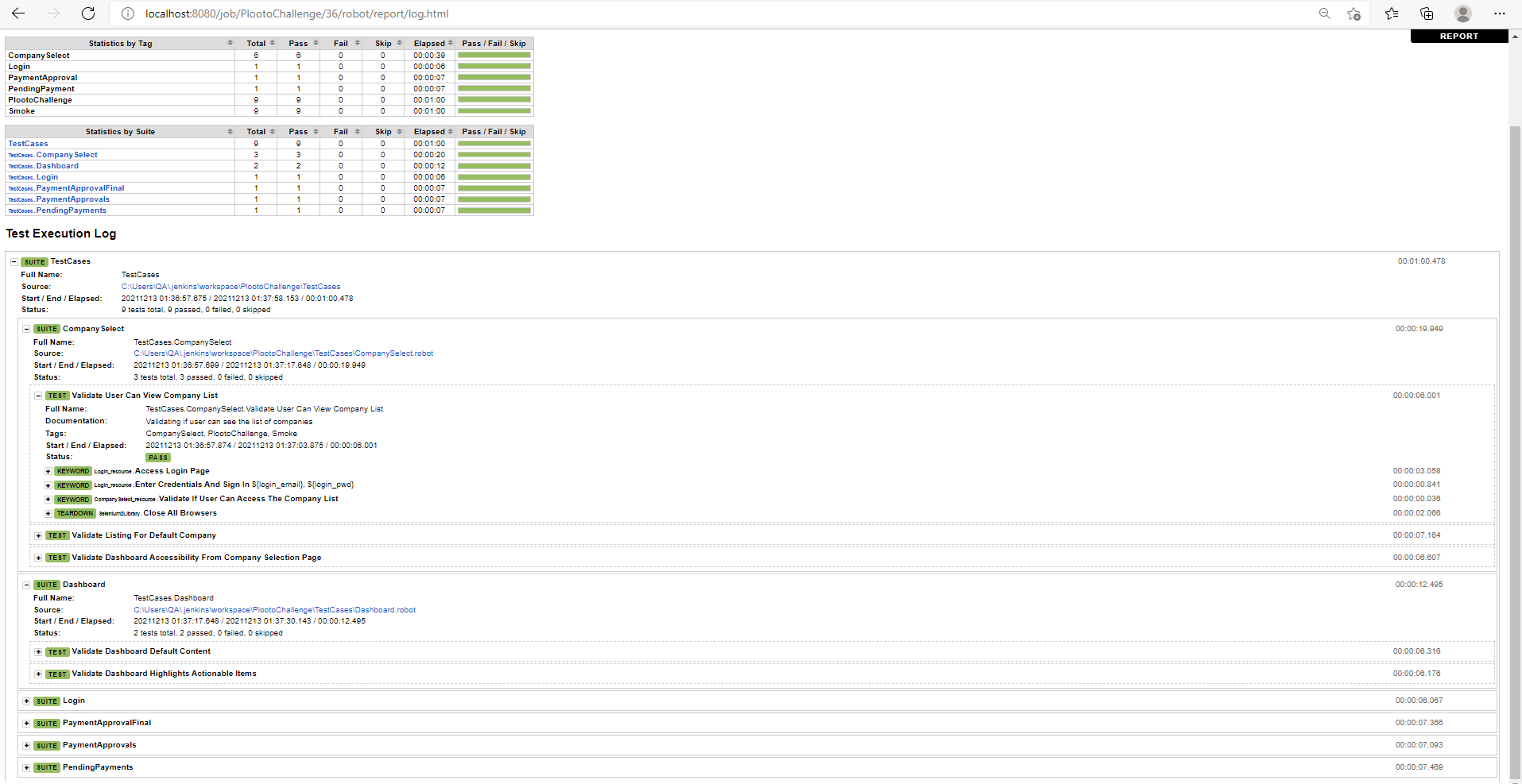
* Job/Test Execution Summary



* Test Execution Details



* Test Execution Log



# Future Enhancements

The below future enhancements can be made to the framework.

* **Multiple Browser Support** – Currently the framework support test execution in Chrome browser only. With minimal changes in the *Login\_resource.robot* file, we can achieve multi browser compatibility.
* **Multi Environment Support** – The framework is already designed to support multiple environments by simply appending the environment with its respective URL in the *Utilities.py* file
* **Encryption** – By adding custom encryption to the credentials used for authentication and any Personal Identifiable Information, we can achieve a secure system adhering to the compliances. Custom Encryption methods can be added to Utilities.py and called across the test suites.
* **Parallel Test Execution –** Using the PaBot library in Robot Framework, the tests can be configured to run parallel. This will reduce the test execution time
* **Grafana Dashboard –** With the existing Jenkins setup, Grafana and InflexDB can be integrated to generate a more advanced Test Result dashboard.
* **API Test Automation –** The existing framework can be used to test API by adding the API support libraries.
* **Docker Containerization** – The existing framework can be put to a Docker image. This make the framework portable and be able to plug-in anywhere we want.