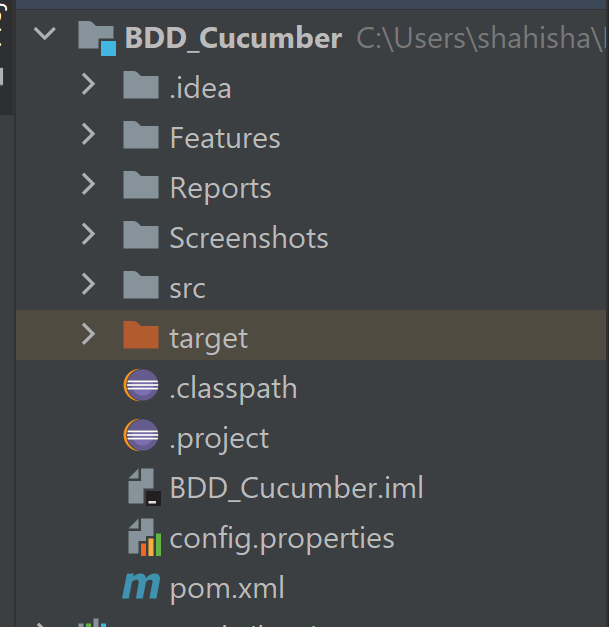
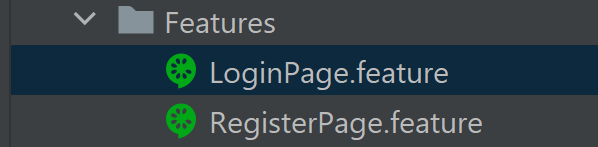
**BDD Cucumber Framework**

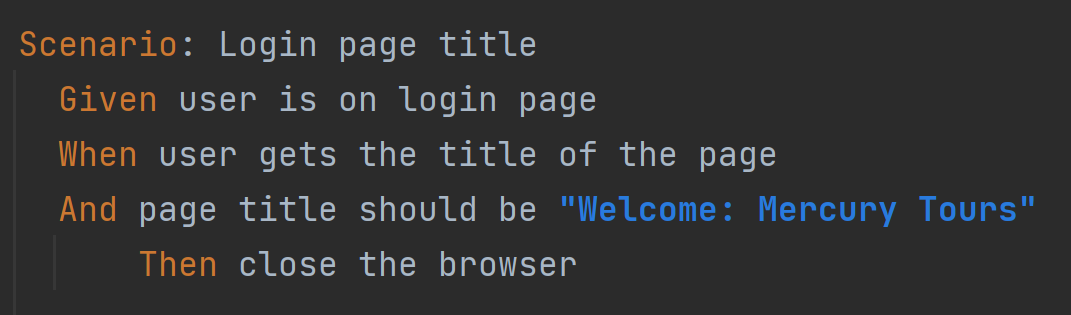
1.0 Folder Structure:



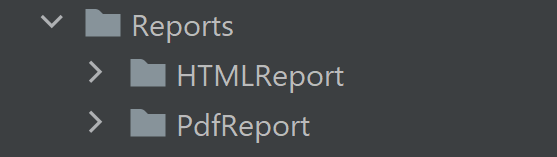
1.1 Components:

1. Features : This Will Contains All the Feature Files with respect to different pages / feature that you want to perform.   
     
   

* The Extension for the File would end with “.feature” after the name of corresponding page you want a feature file for.
* In this Feature File we can define various feature and Scenarios and Steps with respect to our Requirement using Gherkin Language
* Gherkin is a Business Readable, Domain Specific Language created especially for behavior descriptions, and uses “Given”, “When”, “And”, “Then” for the representation of the Scenario.



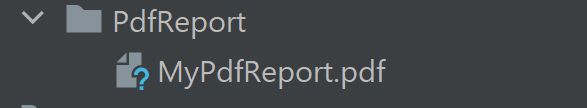
1. Reports: This Framework generates 3 kind of Reports, two of them are present in this Folder, and one is an online version which can be enabled and disabled in the TestRunner Class, which will be

covered later when we talk about the runner file. 

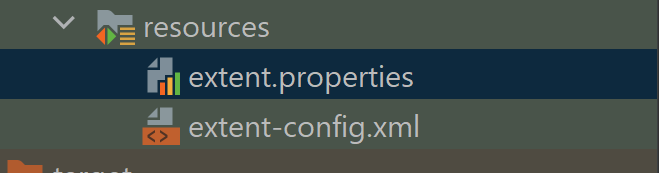
* HTML Report folder has an “.html” report which is again coming from Extent Reports Library



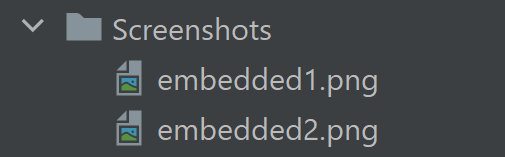
* PDF Report contains an PDF version of the report with detailed information and graphs.



* All the properties are stored in an “.properties” file stored inside test/resources



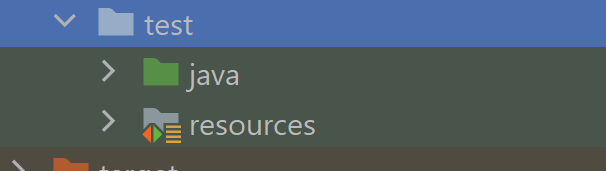
1. Screenshots: This folder consist of Screenshots which we are appending in our Extent Report on failure, using Cucumber hooks which we will discuss further.



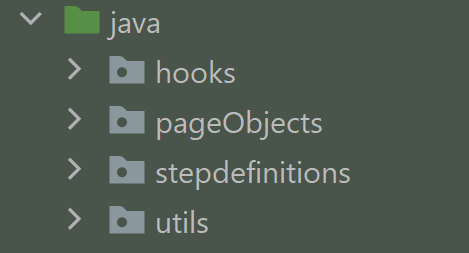
1. src folder: This Folder contains main, target and test folders in which we only have to focus on test folder and its structure.



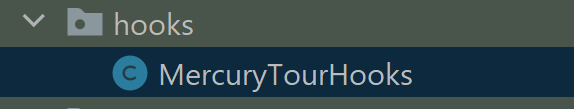
* test folder: It comprises of two folder java and resources ( refer Reports point resources is covered there )



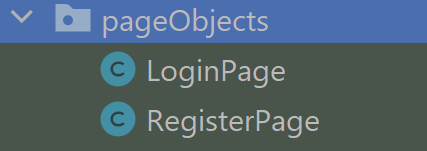
* Within java we have our packages which contains “.java” classes and methods which we use in execution of the framework



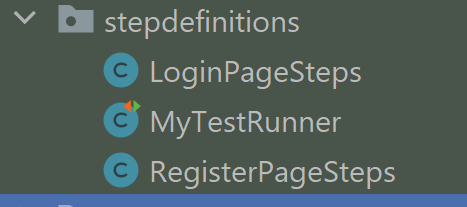
* hooks: Hooks are blocks of code that run before or after each scenario in the Cucumber execution cycle. This allows us to manage the code workflow better and helps to reduce code redundancy. Hooks can be defined anywhere in the project or step definition layers using the methods @Before and @After.



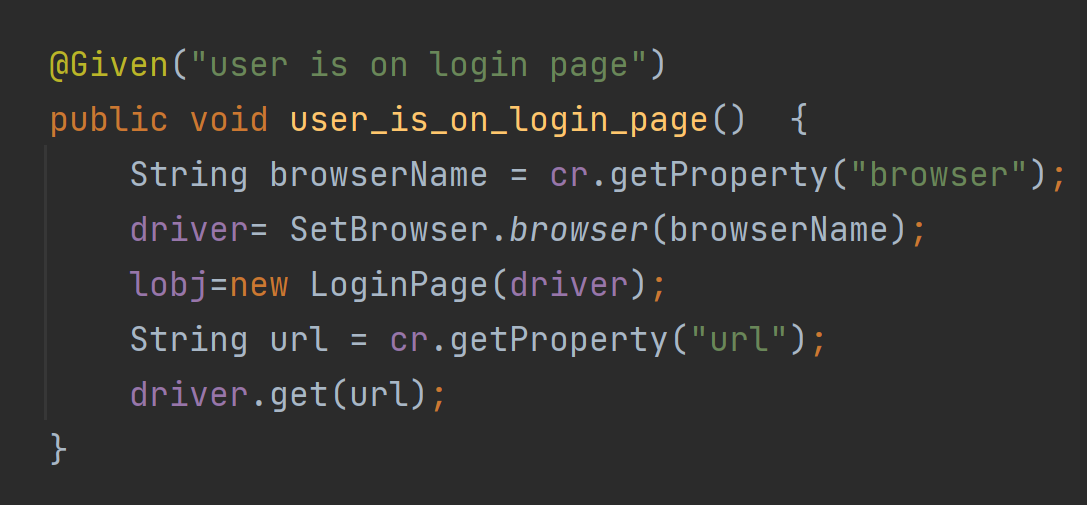
* pageObjects: This package contains the page objects of the page you want to run Automation on, for each page there will be a separate pageObjects Class, just like the Feature file. This uses PageObjectModel pattern iternally and stores the elements of the WebPage efficiently to be used during Execution of the Automation Test.



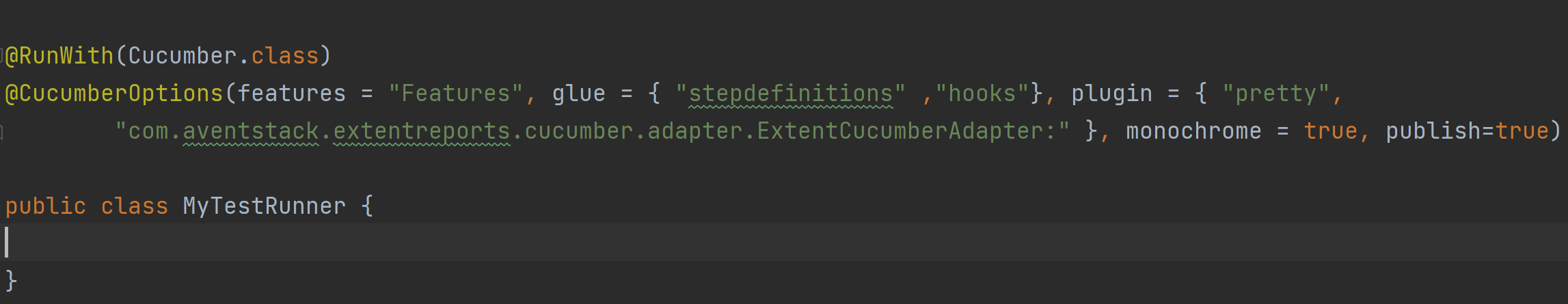
* stepdefinitions: This contains the working logic for the Feature File, for every Scenario and steps present in the feature file we have to define the stepdefinition file which tells cucumber to run the test efficiently.



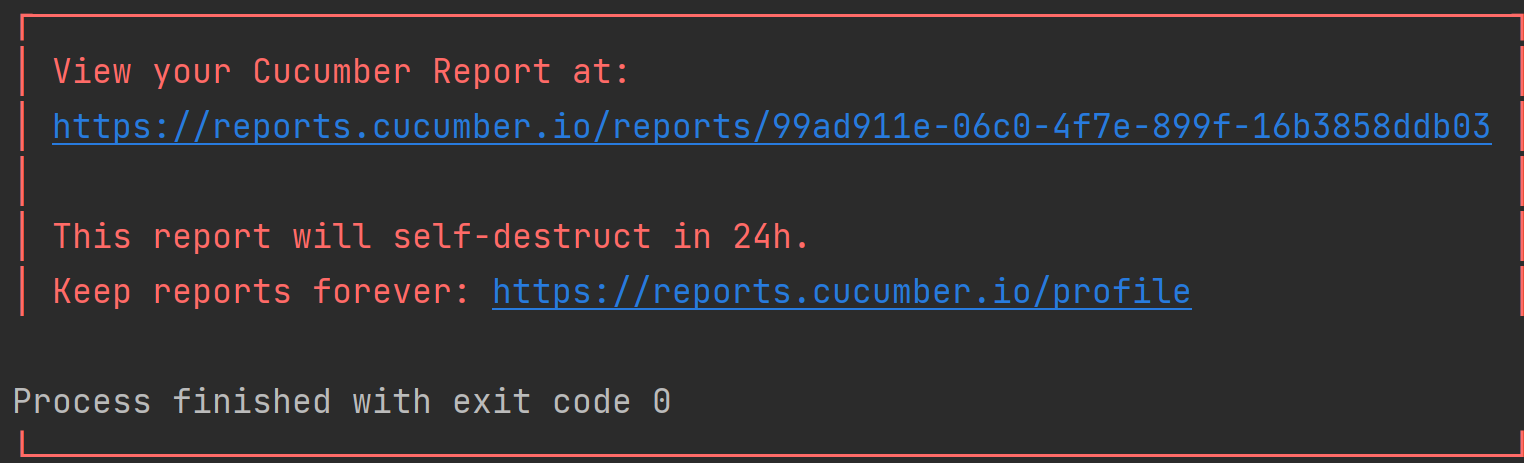
Working Logic of one given step ( Example ) :



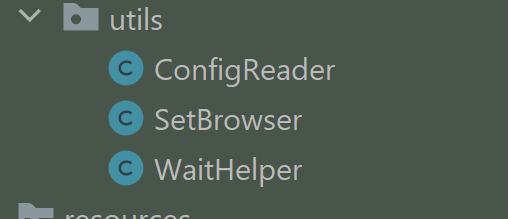
* MyTestRunner Class: TestRunner is a program used in Cucumber to access Feature file. Our Cucumber test are ran using this runner file, we can glue and add plugins, hooks, reports in this runner file.



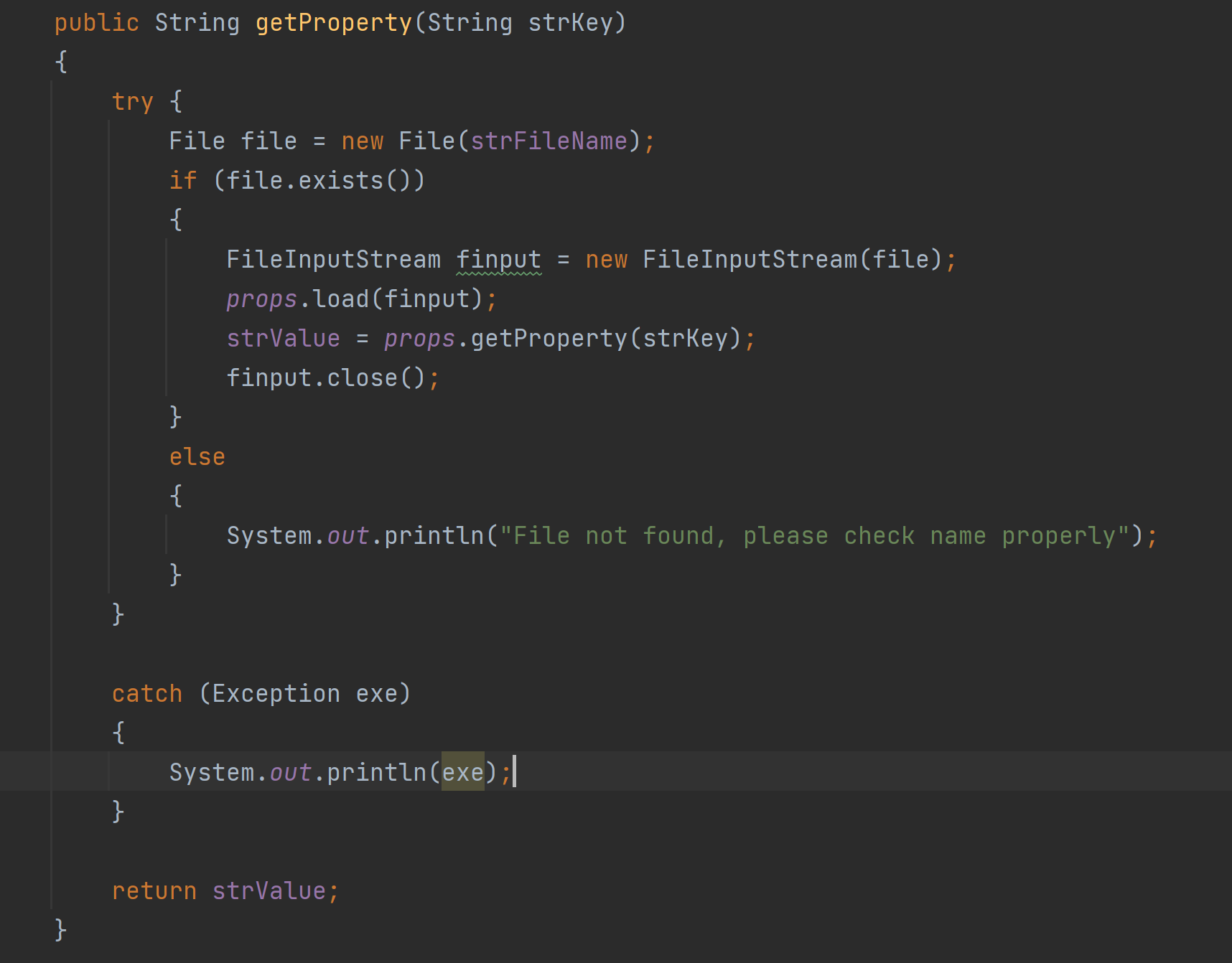
* To generate an online version of report we can turn the “publish=true” in the plugin, shown in above image, and we will get an link in the console which can be shared.



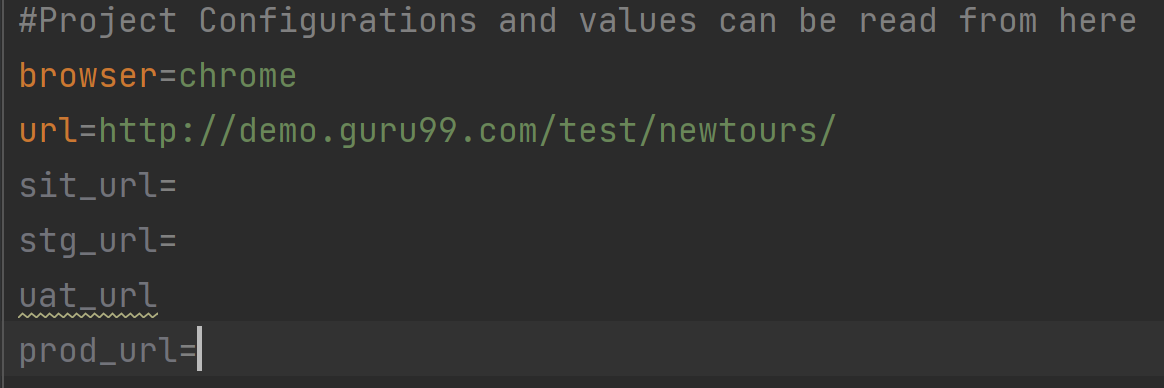
* utils: This package contains the java classes which are re-useable and adding utility / help to the framework, to do certain steps / actions.



* ConfigReader: It is an Java class which helps in reading the “.properties” file.



* It has an method , getProperty which takes “String” as an input and returns the “String” value for that particular valueKey which is passed.
* Properties files look like this, and is in key and value format:



* SetBrowser: It is an Java Class which helps you set the browser of your choice you just need to mention the name of the browser in the “config.properties” file and it will do the rest of the work, if your browser is not present in methods you will have to create one / else it runs on Chrome by default.
* A glimpse of working logic, which uses WebDriverManager internally to handle the WebDriver versions and dependency.



* WaiHelper: This Java class contains methods to help Selenium Synchronize the Waits, and Timeouts which one needs to give for proper execution, this class is yet to be worked on and can be extensively used , currently only few methods are present which can be used as an when required.

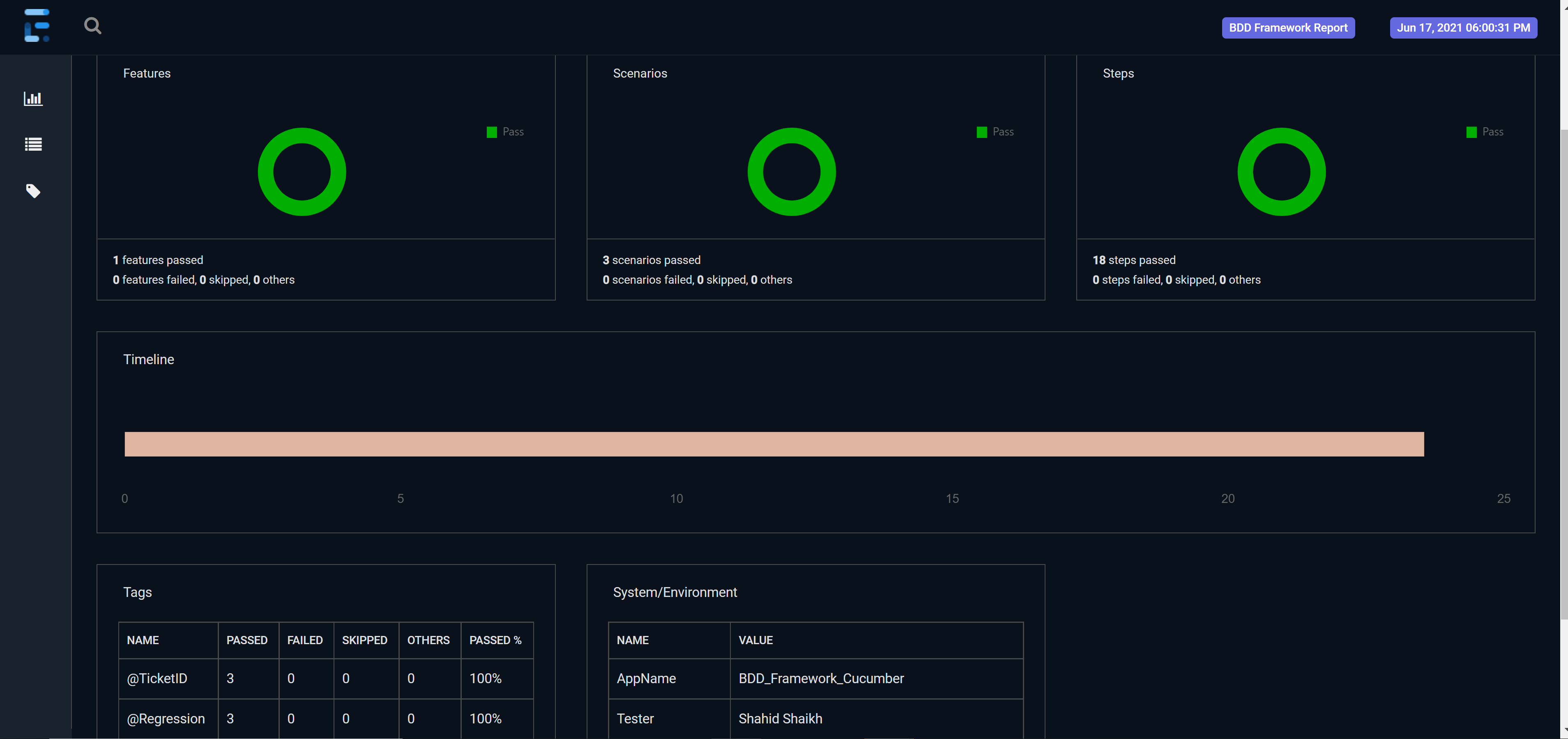


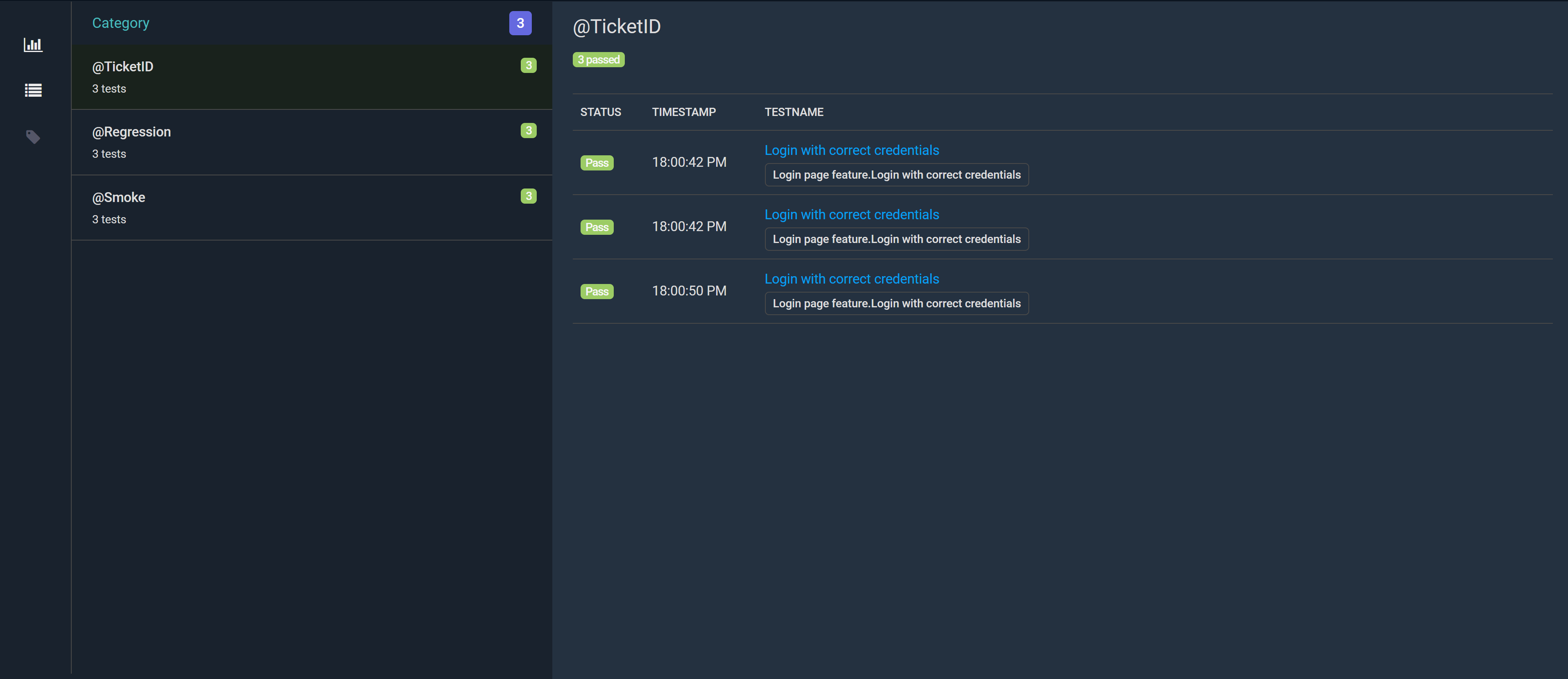
1. config.properties : This file can be used to give the general details about the project , and some other credentials which can be then read using the ConfigReader class



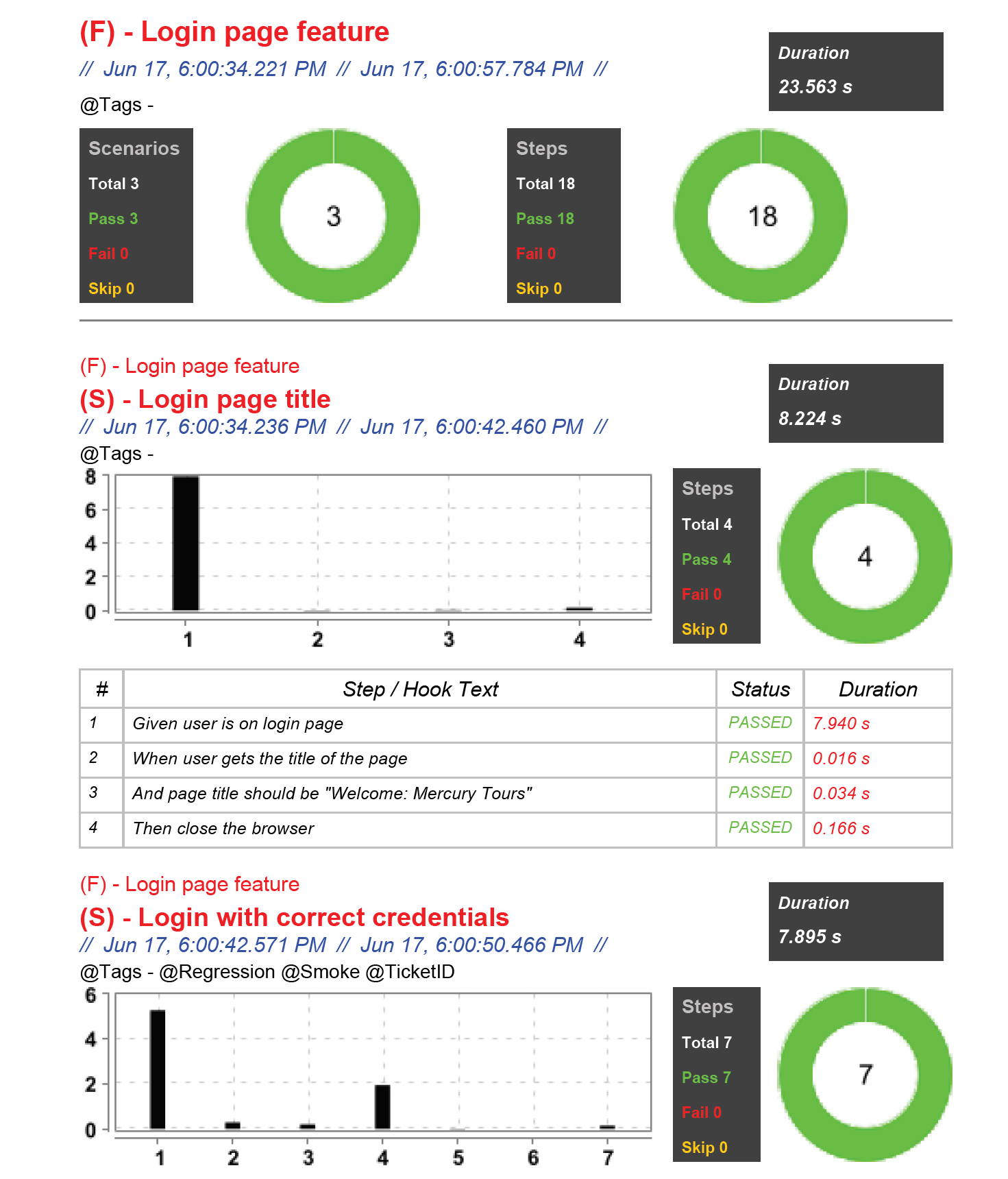
2.0 Snapshots of Reports:

HTML:





PDF:



3.0 Gitlab Link:

Clone this Project -  
 <https://main.gitlab.in.here.com/shahisha/bddframework>

Maintained & Created by – Shahid Shaikh

Contact for issues / doubts / enhancements ideas –

[shahid.shaikh@here.com](mailto:shahid.shaikh@here.com)