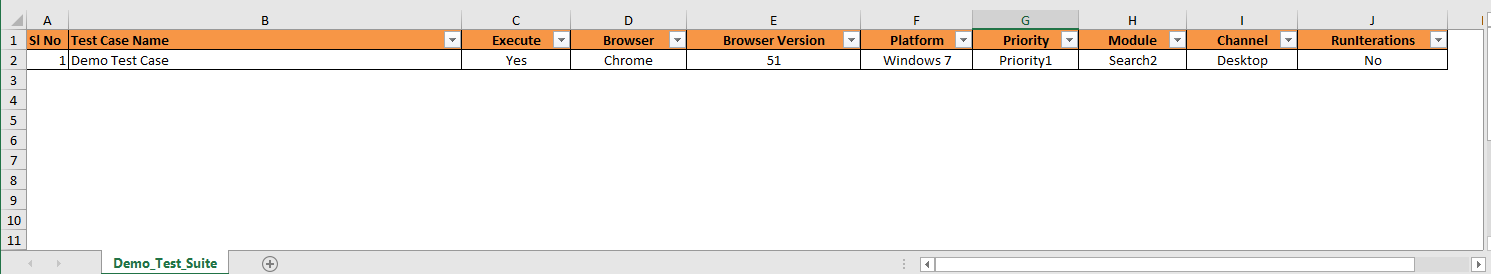
**Bug Hunt Automation Framework**

Instruction Manual

Test Case Creation

Follow the below steps to automate new test case

1. Add a new row in TestManager.xls with test case details. Test Case Name and Execute are mandatory columns. Rest all columns are customizable as per project needs. All test details (column values) can be retrieved in code using dataUtil.getTestManagerColumnVal(“columname”) method.



Sheet name given in TestManager (Demo\_Test\_Suite in screenshot) will be used as suite name. Same name must be used for TestSet property in bug-hunt.properties and file name for data sheet.

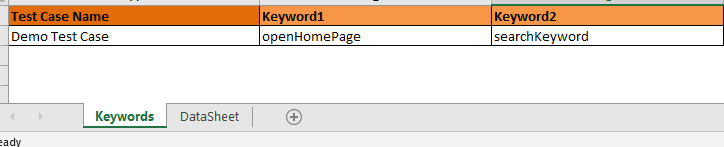
1. Create new excel file (Data file) with same name used for sheet name in TestManager.xls in folder

\src\main\resources\Data\Excel

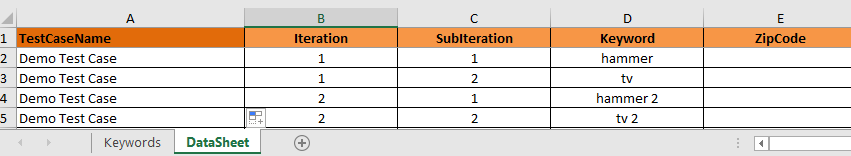
(Rename Demo excel with test suite name, this is sample template)



1. In newly created excel file in step 2 , add test case name and methods that you want to call for the test case. Method names can be added later once you create the Skeyword methods.



1. In DataSheet sheet add test case name and data you want to use. Data to be used can be updated later once you start working on test case automation

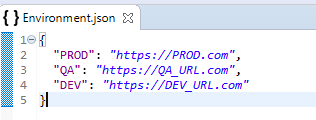


1. In bug-hunt.properties file update TestSet and Environment properties.

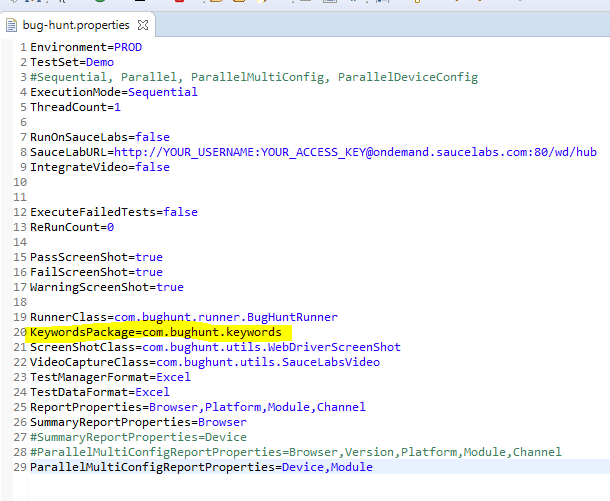
Environment=PROD - (URL is read based on mapping in Environment.json file)

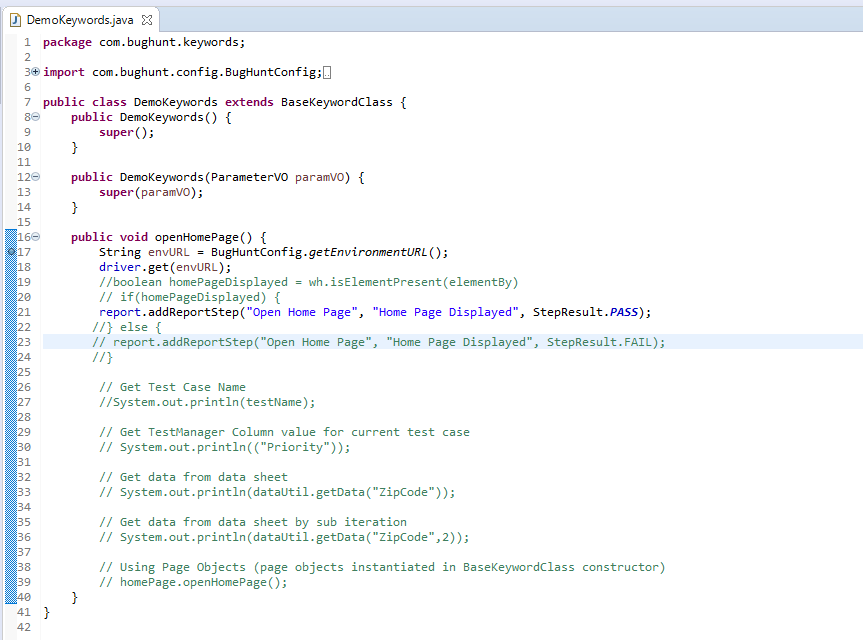
TestSet= Demo\_Test\_Suite

1. URL mapping is done in src/main/resources/Environment.json, which is used in bug-hunt.properties file



1. Create keyword class by extending BaseKeyword Class in com.bughunt.keywords package. Package name is configurable in bug-hunt.properties file. But base class name must be BaseKeywordClass and must have one default and one parameterized constructor. Do not change this base class constructor signature.

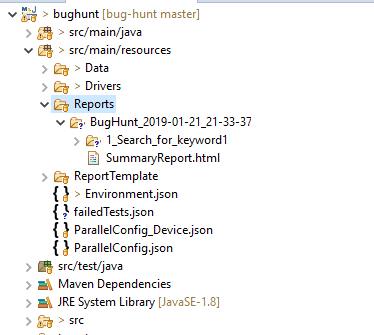




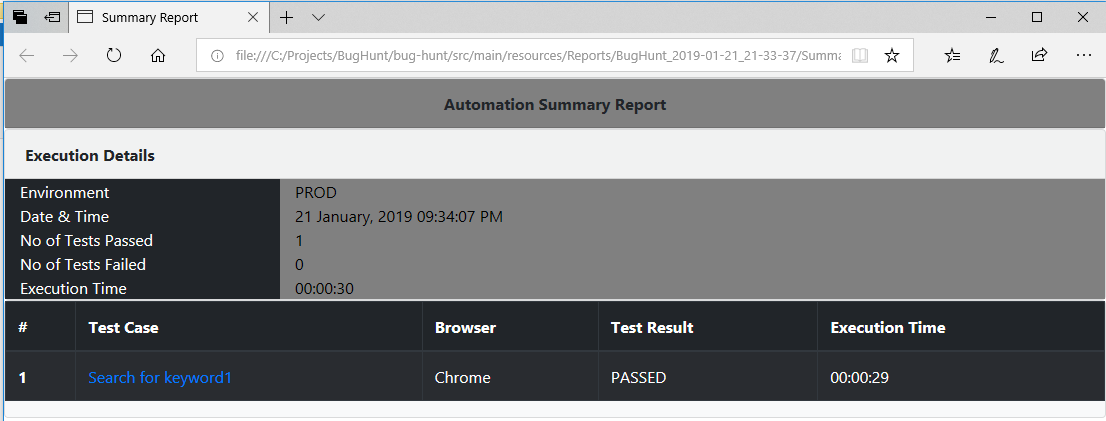
Above screenshot shows how we can use

* Reporting
* Get Test Case Name
* Get column values from Test Manager columns for current test case
* Get Data from Test Data Sheet
* Use Page Objects (user can come up with own design, this is just sample)

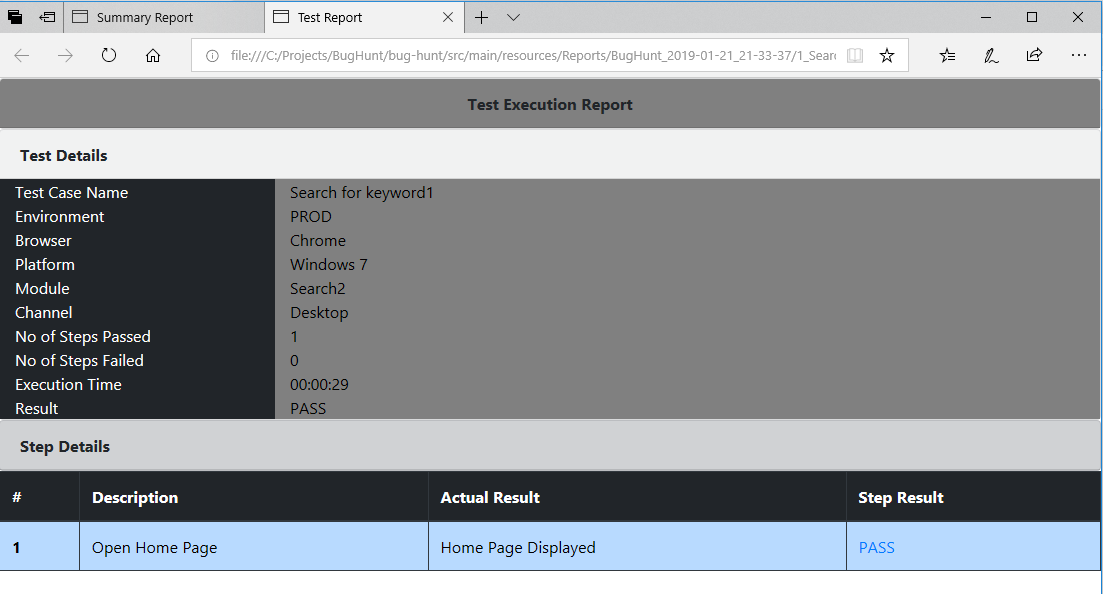
1. To execute test suite open BugHuntRunner.java class and run as Java Application
2. Once execution is completed report is generated in src/main/resources/Reports



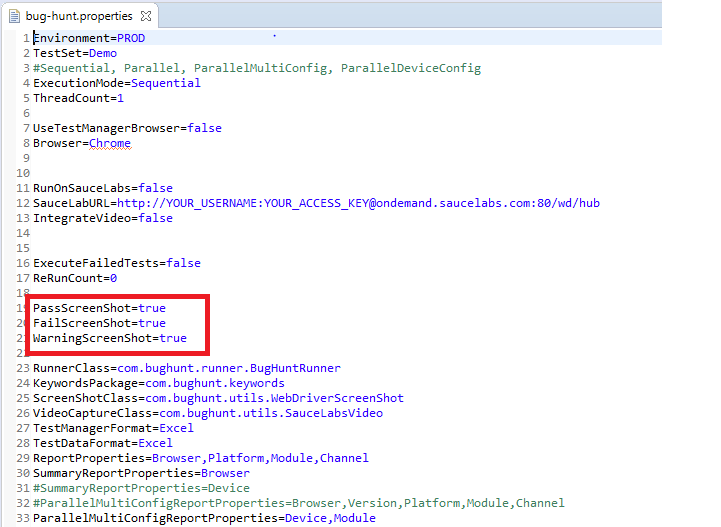
Summary Report



Test Case Report

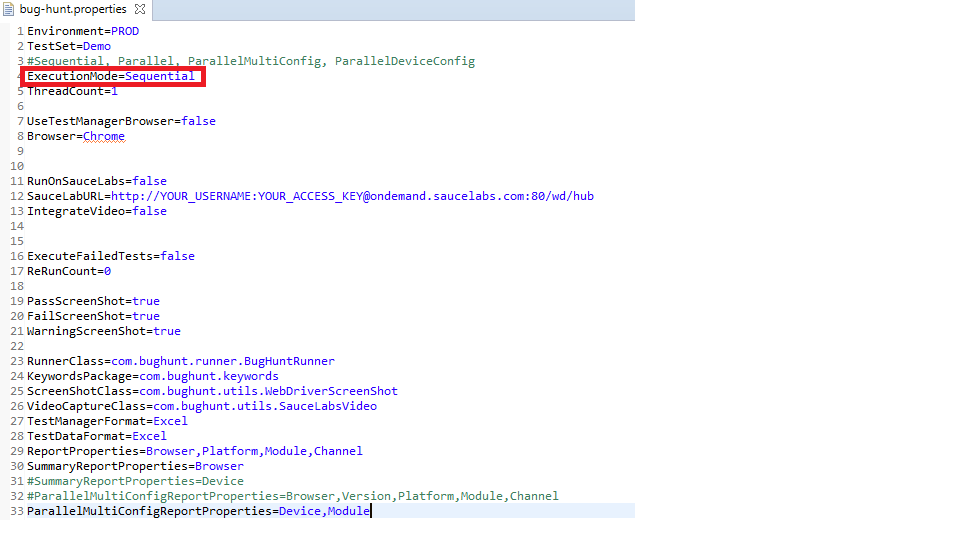


Clicking Step Result link will open screenshot in new tab. Screenshots configuration can be done in bug-hunt.properties file



Execution Modes

1. Sequential – To execute tests in sequential mode set ExecutionMode=Sequential in bug-hunt.properties

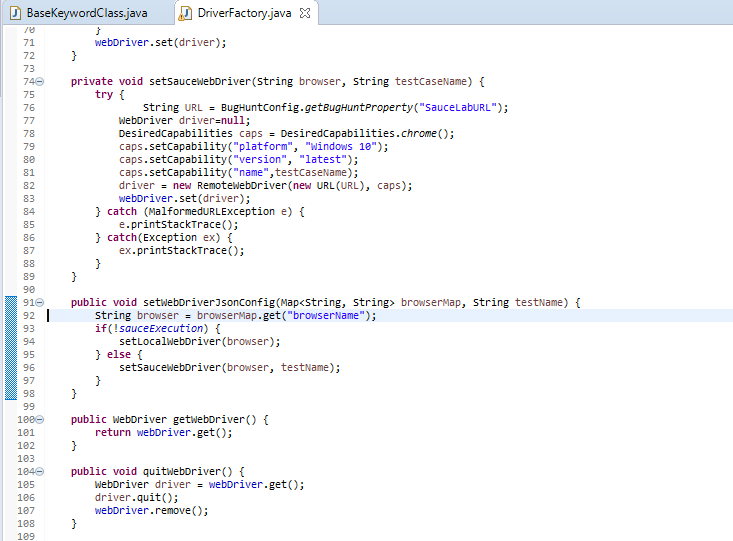
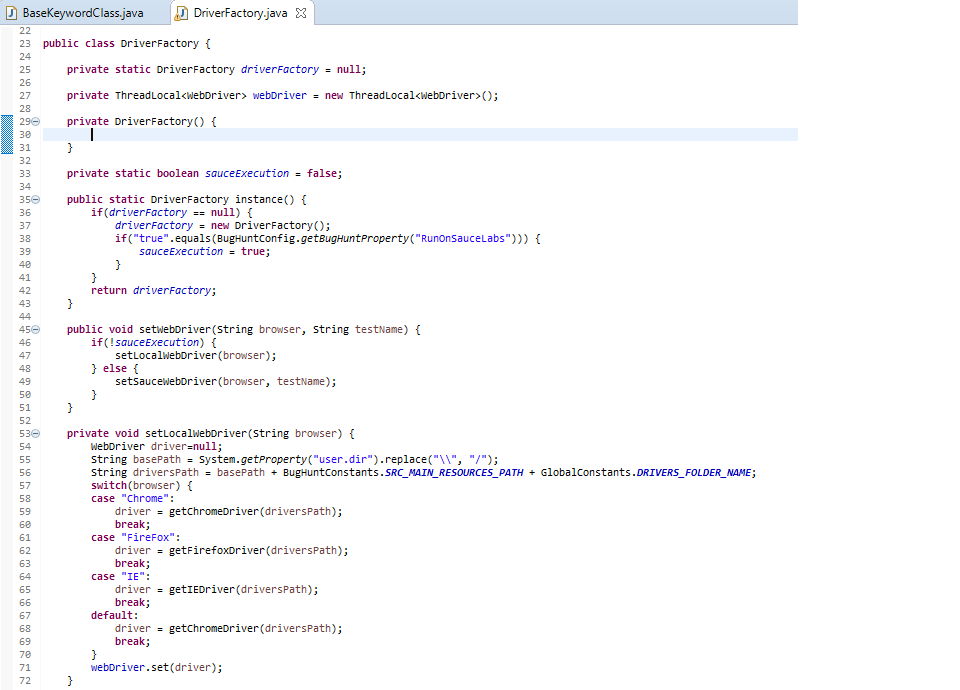


1. Parallel – To run tests in parallel set ExecutionMode = Parallel and ThreadCount to number of threads. This will run tests in parallel.

There are 2 ways you can run tests in parallel.

* Running test cases on same browser reading browser value from bug-hunt.properties file. If in bug-hunt.properties file RunOnSauceLabs=true then tests will run on sauce labs.

All these configurations are in bug-hunt project (not in bug-hunt-lib project). Its fully customizable and can be used with any grid setup.

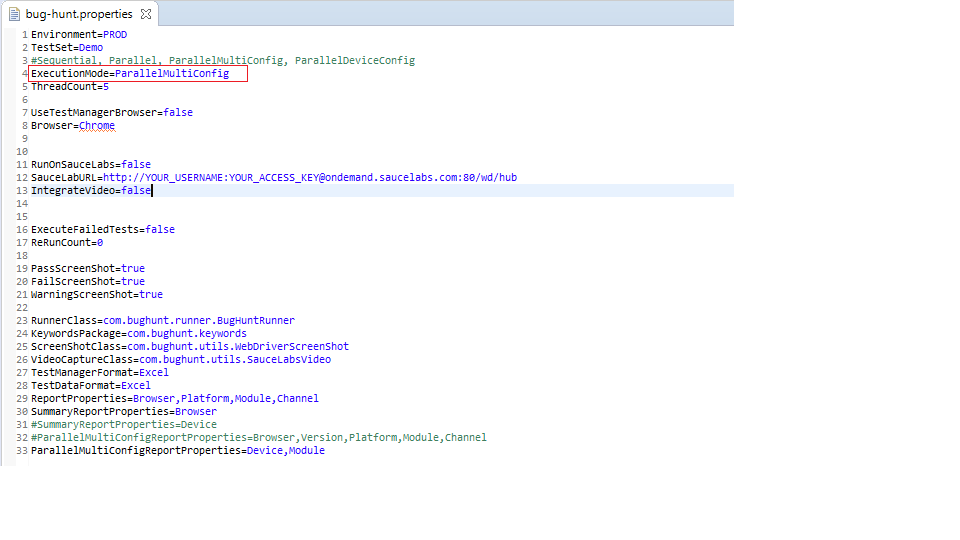


* Running test cases in parallel using browser value in TestManager sheet

by setting UseTestManagerBrowser=true. This way we can run different tests on different browsers.

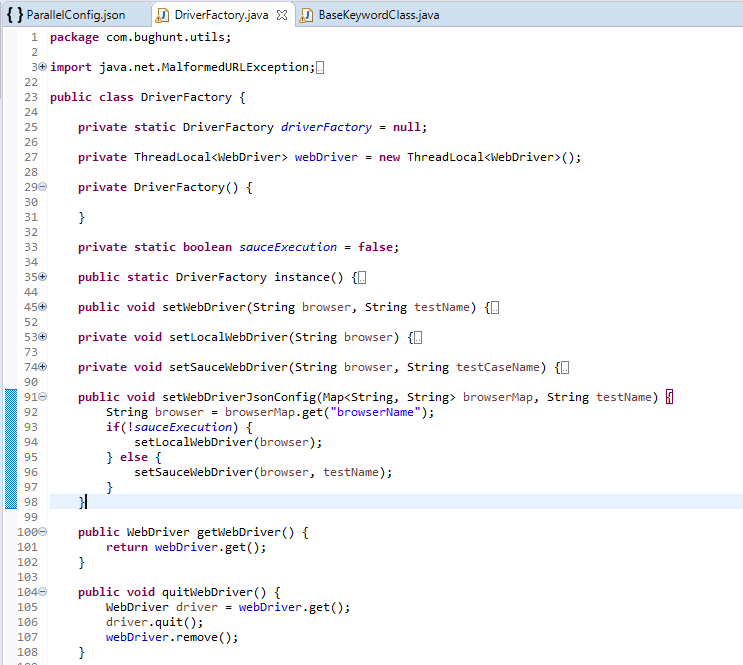
1. ParallelMultiConfig – To run each test case on different browsers set ExecutionMode= ParallelMultiConfig. Using this configuration different tests can run in parallel on different browsers by setting Thread Count more than one.

Different browsers to run can be set in ParallelConfig.json. Respective config item in config array is passed as map to keyword class, which can be used to open appropriate browser.

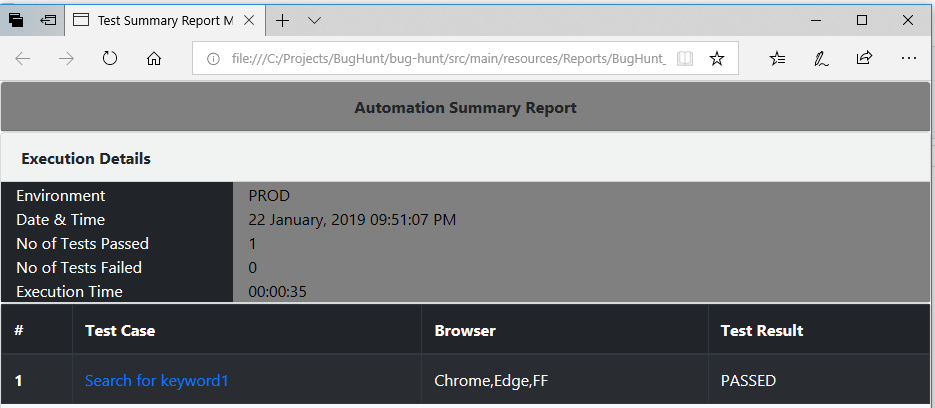


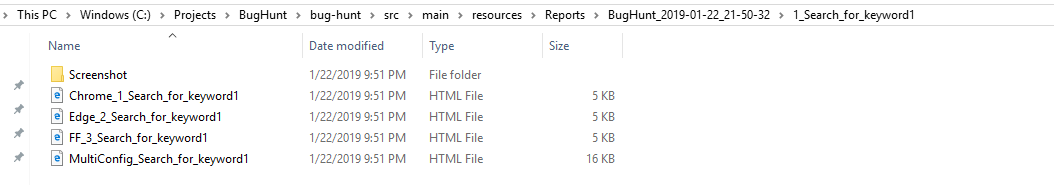


/

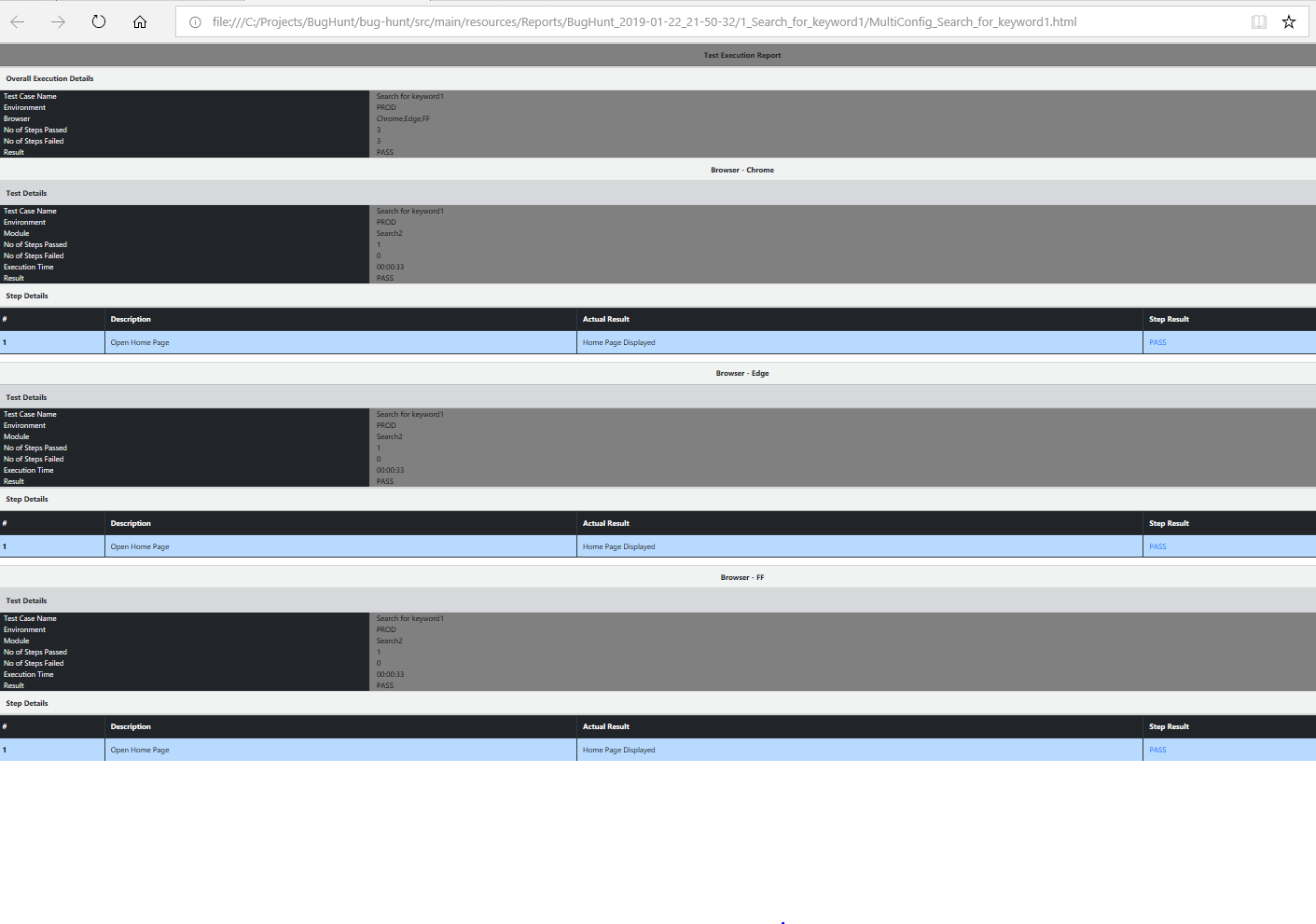


Summary Report generated for multi browser setting

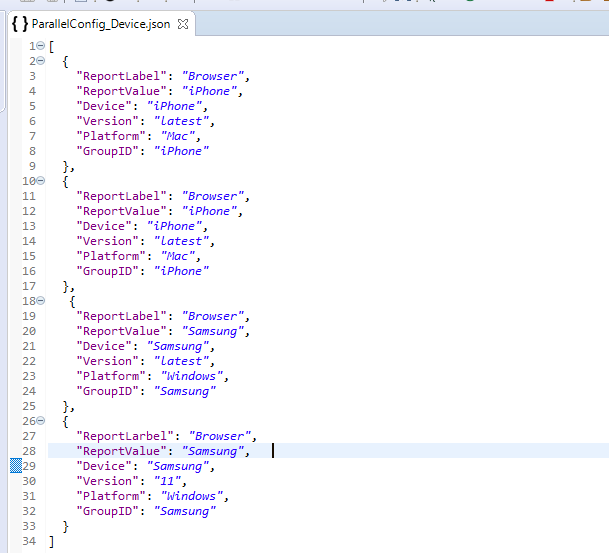




MultiConfig merged test case report is generated for each test case.



1. ParallelDeviceConfig – Using this configuration tests can be run in parallel on multiple devices. If device configuration has same group id then tests will run for grouped config in parallel.

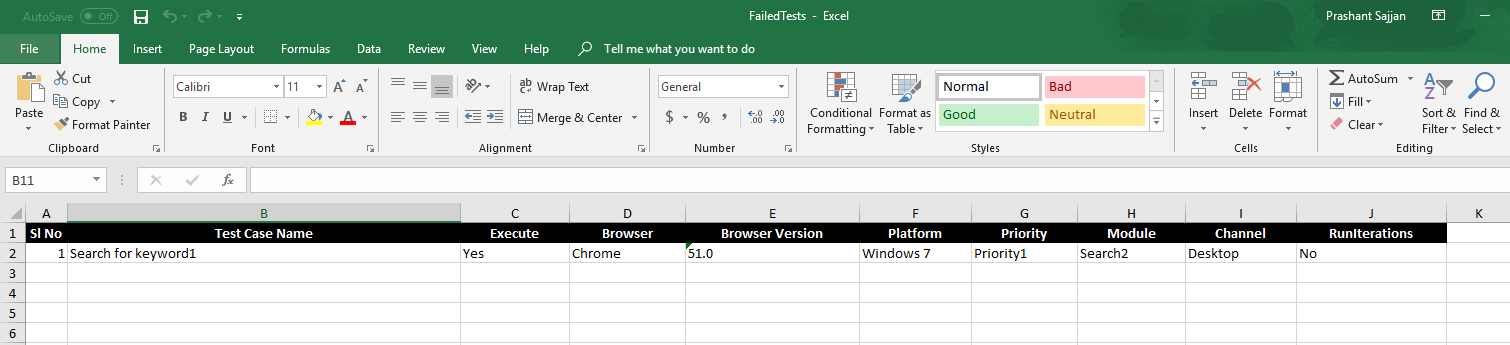


With above configuration, tests will run in parallel on two iPhone devices i.e. TestCase1 on one iPhone device and TestCase2 on another iPhone device as GroupID is same. Similarly, tests will run in parallel on Samsung device.

Running Failed Test Cases

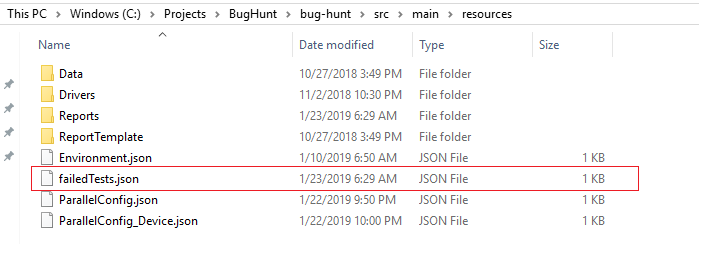
Set ExecuteFailedTests=true in bug-hunt.properties file. Framework will read tests from FailedTests.xls.

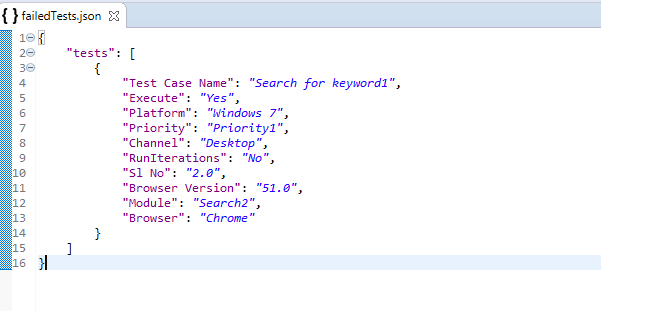
This excel FailedTests.xls is generated after execution is completed. If you stop execution in between or execution gets terminated this file will not be generated. In such cases, failedTests.json file is used to run failed tests, which gets generated after each test case is executed.



Above file will be generated in the root folder. Using this file, we can select tests which you want to execute. As some tests may be actual failures due to application defects.

If this file is not generated tests cases are read from “failedTests.json” in ” \src\main\resources” folder.





If Execute in failedTests.json is set to “No” for any test, then that test will not be executed.

**Re Run TestCases**

If you want to rerun test case if it fails, then set ReRunCount to number of times you want to rerun.

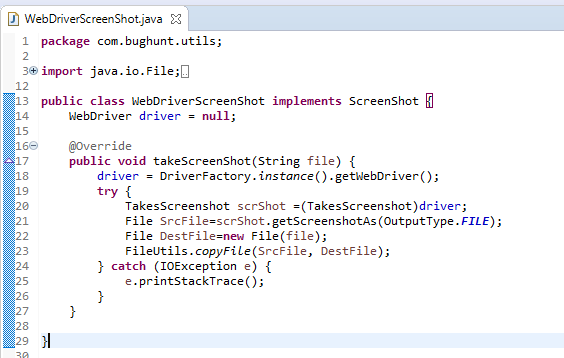
Example: Setting ReRunCount=2 will run test case twice if test fails again after 2nd execution. If test passes before it completed max rerun count, then rerun execution will be stopped.

**Reporting**

Reporting in bug-hunt framework is implemented using mustache templates and bootstrap. These templates are present in “\src\main\resources\ReportTemplate” folder. Based on your needs you can change the styling of report.

To attach screenshot to report in bug-hunt.properties file ScreenShotClass is set to com.bughunt.utils.WebDriverScreenShot

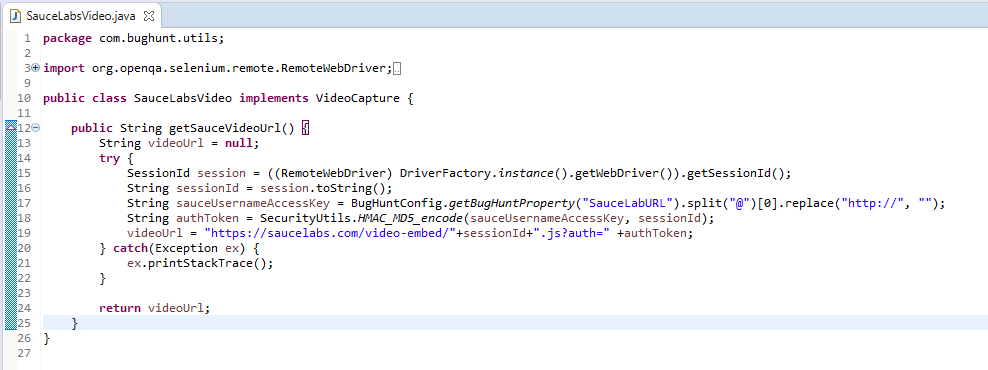
ScreenShotClass=com.bughunt.utils.WebDriverScreenShot



If you are using any tool other than Selenium you can create class that implements ScreenShot interface and provide implementation to save screenshot in takeScreenShot method. ScreenShot path is passed by framework. This way framework is not tightly coupled to any tool.

**SauceLabs Video Integration**

Framework has feature to integrate SauceLab videos into reports. To integrate SauceLabs video set IntegrateVideo=true and VideoCaptureClass=com.bughunt.utils.SauceLabsVideo



If any other Grid service provider provides video feature you can create class that implements VideoCapture interface and implement getSauceVideoUrl.

Framework is completely customizable to work with any tool and grid service.