E.C.I.NETWORKS

<

WIFI AUTOMATION – ROBOT AUTOMATION – USER MANUAL

Bell Canada; ATL Lab

Bell

Contact:

Angelo Virgilio

Angelo.virgilio@ecin.ca

Mobile: 416.605.1296

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
0.1	Arijit Saha	First Draft version	May 30, 2017
0.2	Angelo Virgilio	Added template, formatted and updated various sections	June 2, 2017
0.3	Angelo Virgilio	Added New Section - Next steps	June 7, 2017
1.0	Arijit Saha	User Manual – Complete description for MySQL Project	Aug 15, 2017
2.0	Arijit Saha	Final Version – Robot Automation Project (Integrated)	Oct 18. 2017

Review & Approval

Requirements Document Approval History

Approving Party	Version Approved	Signature	Date
Bell Canada: **Bertrand Camus** **Intissar Harrabi**	0.3		

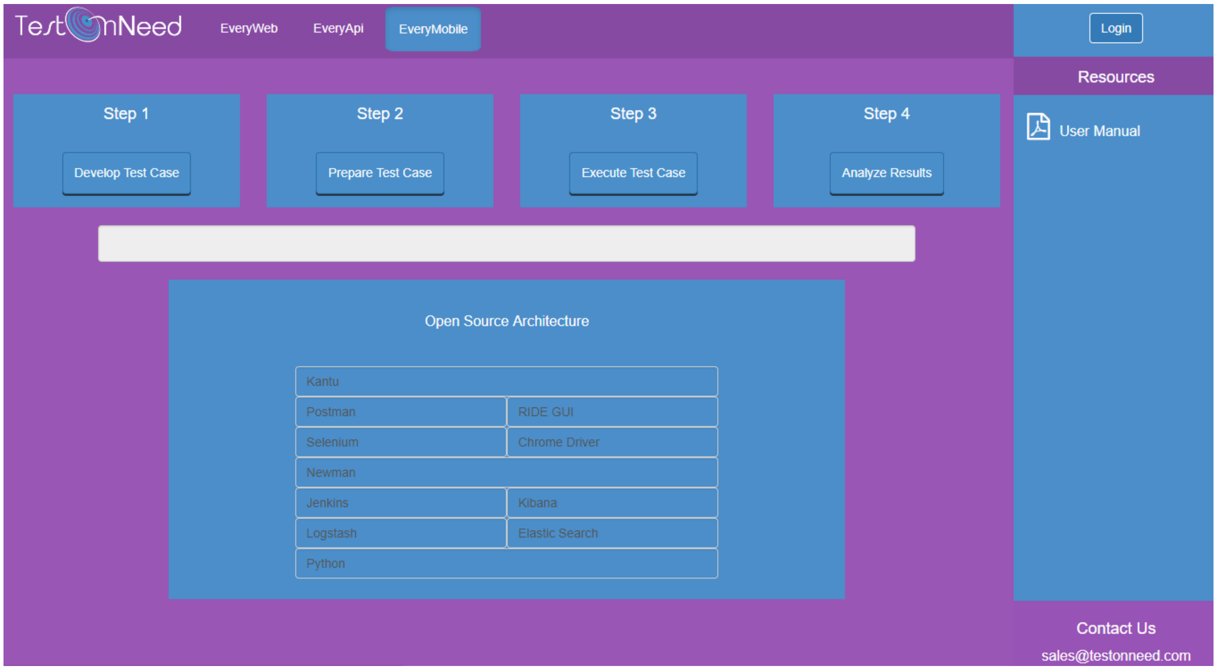
Requirements Document Review History

Reviewer	Version Reviewed	Signature	Date
Angelo Virgilio	0.3		June 9, 2017

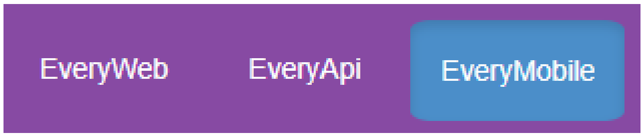
TABLE OF CONTENTS

1. DESCRIPTION OF THE WEB PAGE

- The outline of the Website looks like below:

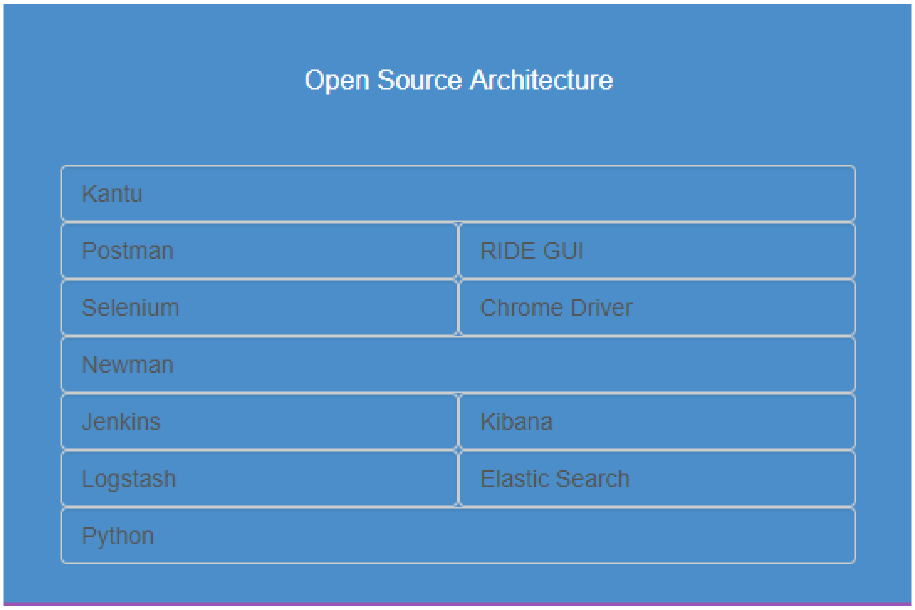


- As the web page opens up, the user will find three tabs namely – EveryWeb, EveryAPI and EveryMobile.



- Things that are common to all the tabs are:
  - In the status bar, the user will get know what is being done next by clicking of any button

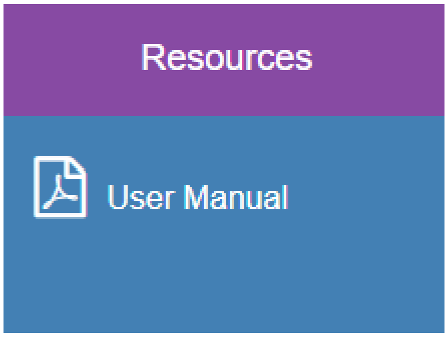
  - The stack of open sources will contain all the extension and application that are used. On mouse hover on any open source, a pop-up window comes up with a brief description of what it is and how it is being used.



Open Source Architecture

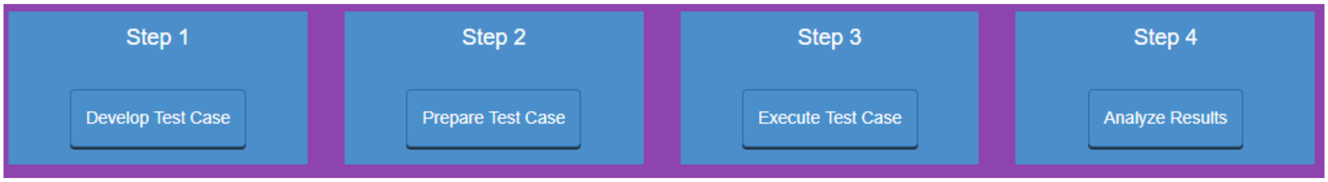
Kantu	
Postman	RIDE GUI
Selenium	Chrome Driver
Newman	
Jenkins	Kibana
Logstash	Elastic Search
Python	

- The user manual guides the user through the web application. The content for the user manual will change based on the current tab being selected. It is a linked text, clicking on which, the respective document opens in .pdf format in a new tab in the browser



2. EVERYMOBILE PAGE

- The EveryMobile tab is the third and the final tab of the website~~.~~
- This tab has 4 buttons on the top namely –
    1. Develop Test Case
    2. Prepare Test Case
    3. Execute Test Case
    4. Analyze Results.

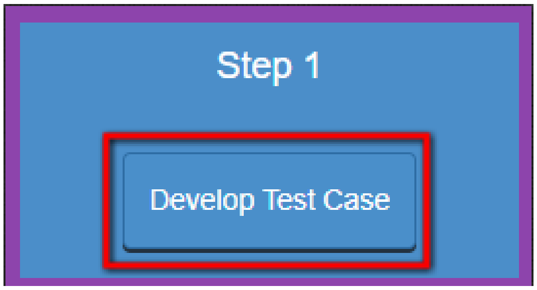


- Each step is explained below

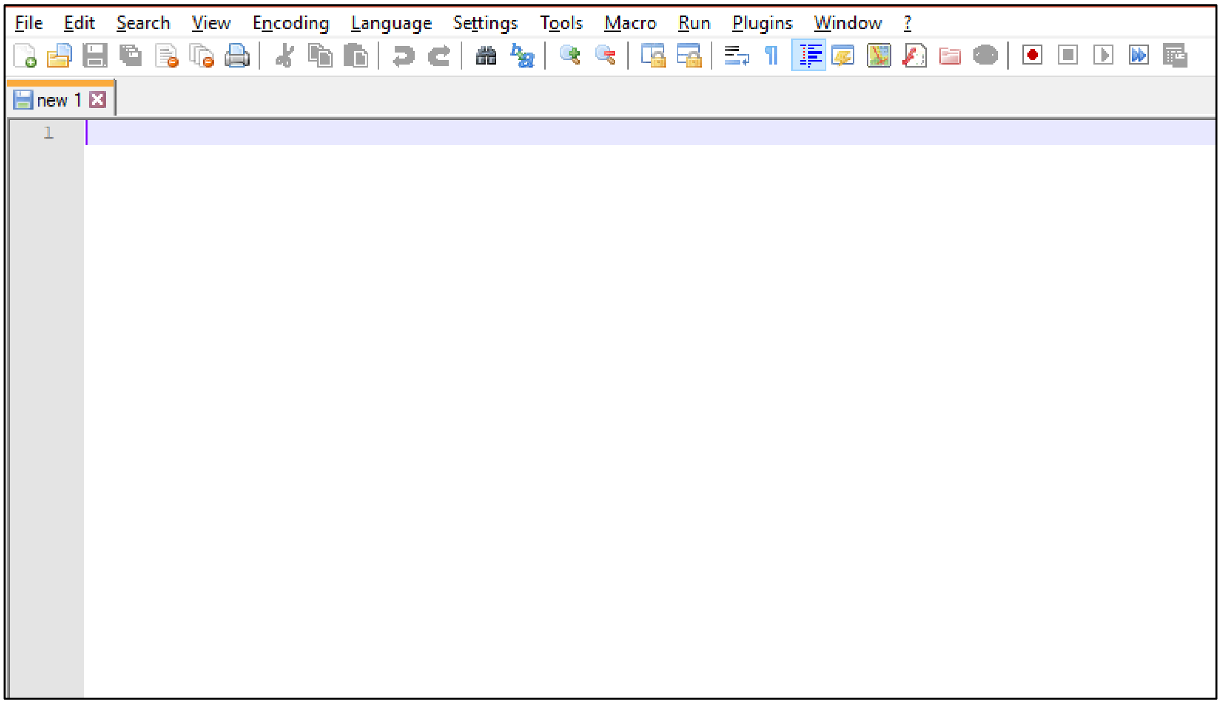
2.1 DEVELOP TEST CASE

The Develop Test Case button when clicked will open Notepad++ for the user. This can be used to write the test scripts for Mobile Application automation by following the guidelines mentioned in the appendix

1. Click the **Develop Test Case**.

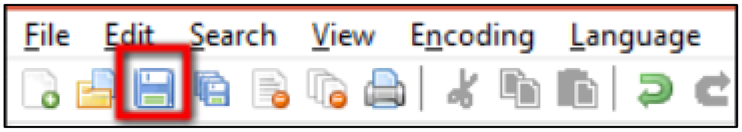


2. On clicking the **Develop Test Case** button, notepad++ opens up.


File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
new 1
1

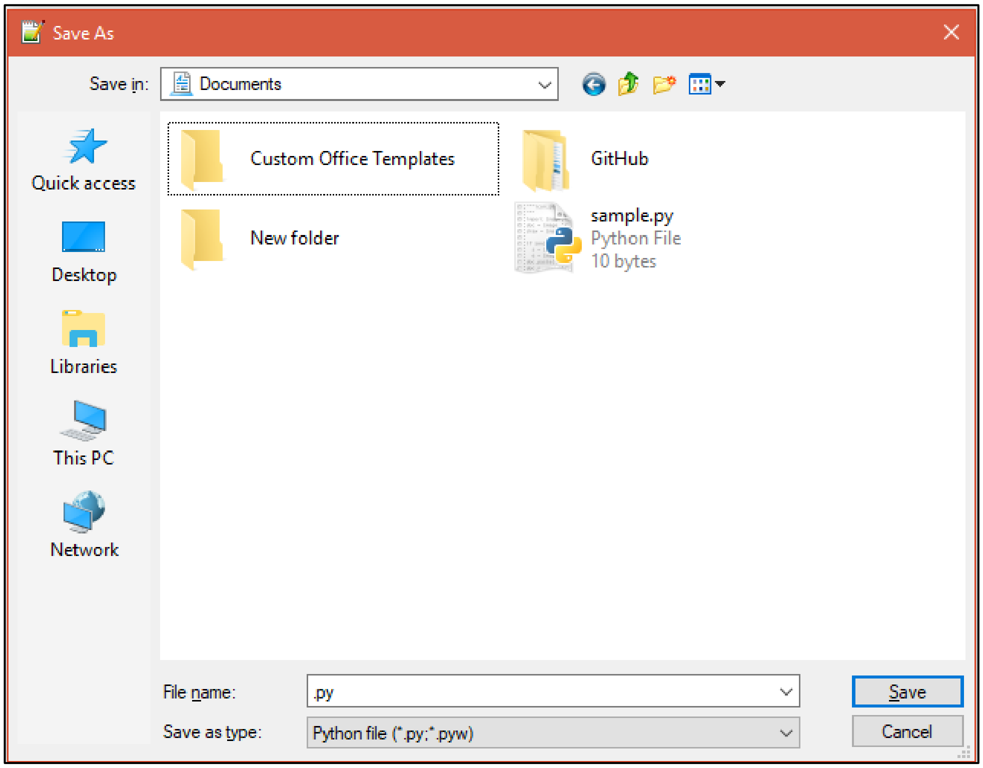
NOTE: The Notepad ++ should be saved in the C: drive then only it will open. For the installation process, please refer the Notepad ++ in the installation manual.

3. Write the Appium code for executing the test case.
4. After writing the code, click Save button.



5. Surf to the location where file has to be stored. Example shows folder location as Documents.

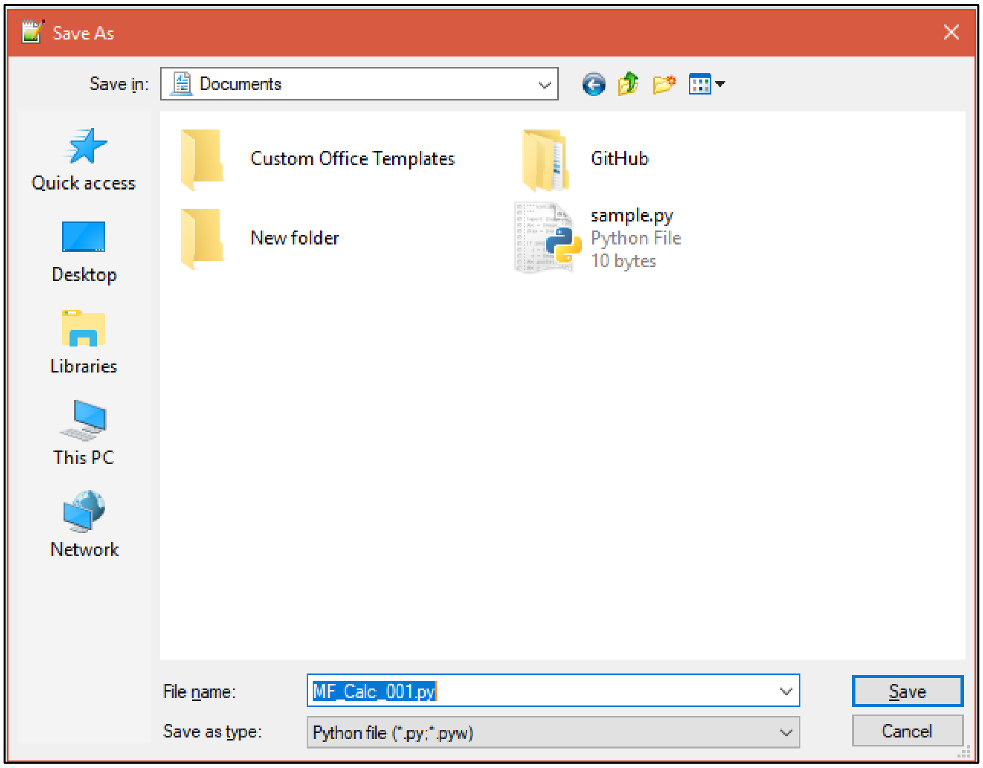
6. In the **file name** text, give the name of the file and in **save as type** drop down, select **Python file**.



File name: .py

Save as type: Python file (*.py;*.pyw)

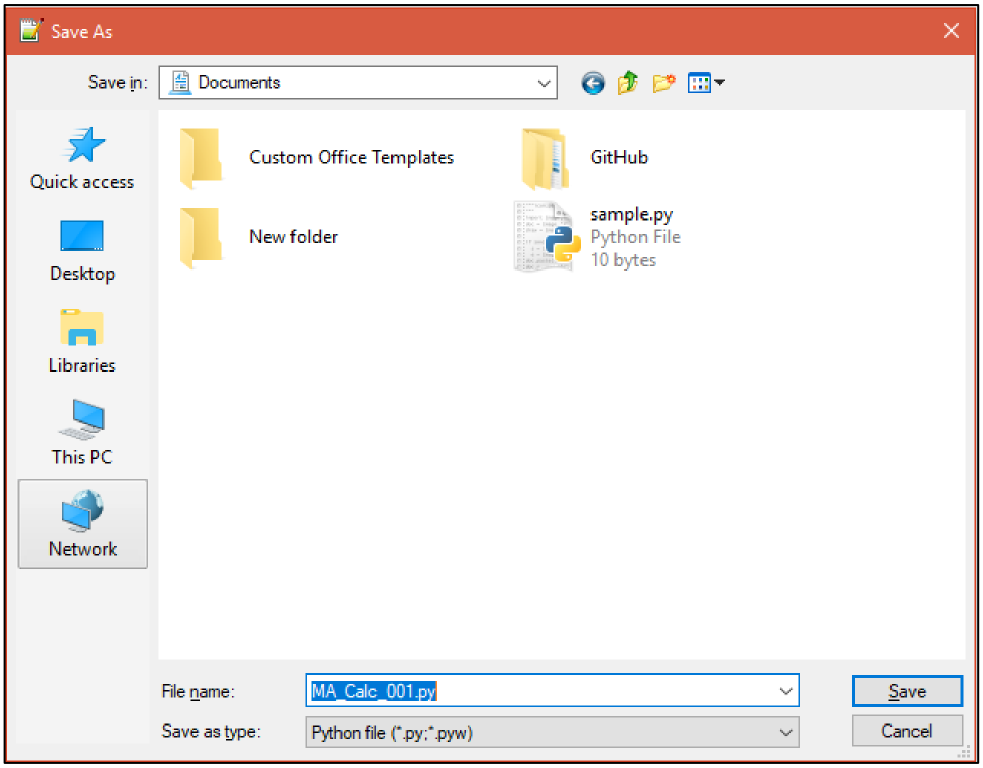
7. Click Save to save the file.
8. Now save the file based on the type of testing functionality. There are 3 types of testing functionality– **Function, Automation and Load.**
9. In the case of **Function**, the test case should be saved as **MF_<TestCase_Name>**. For example, the test case should be saved as **MF_Calc_001.**



File name: MF_Calc_001.py

Save as type: Python file (*.py;*.pyw)

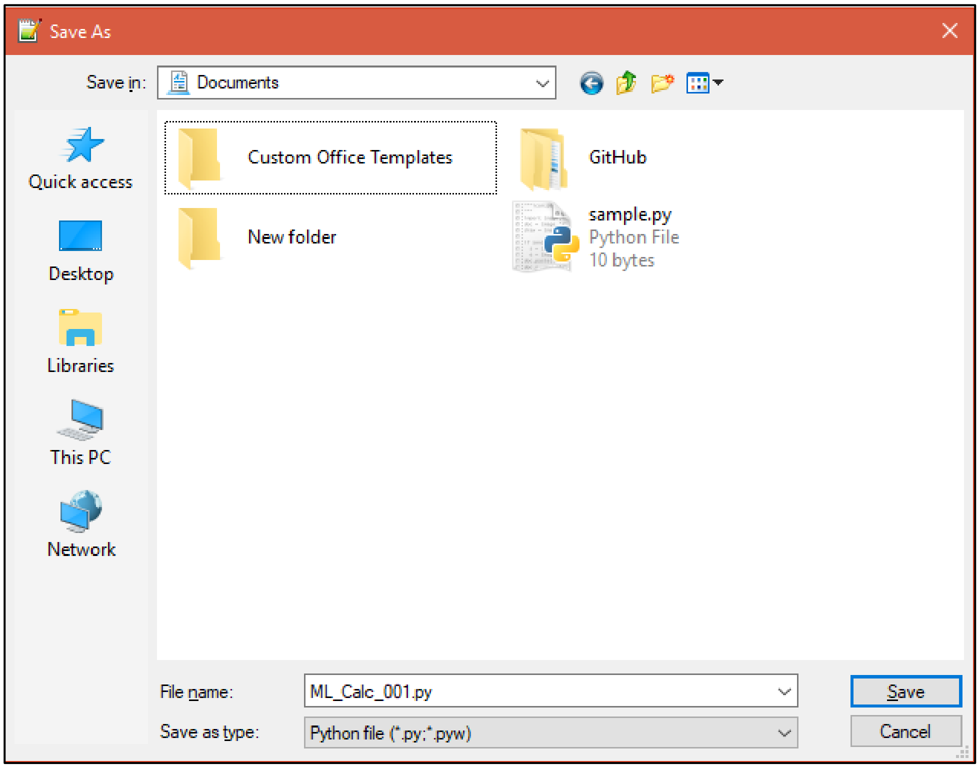
10. In the case of **Automation**, the test case should be saved as **MA_<TestCase_Name>**. For example, the test case should be saved as **MA_Calc_001**.



File name: MA_Calc_001.py

Save as type: Python file (*.py;*.pyw)

11. In the case of **Load**, the test case should be saved as **ML_<TestCase_Name>**. For example, the test case should be saved as **ML_Calc_001**.



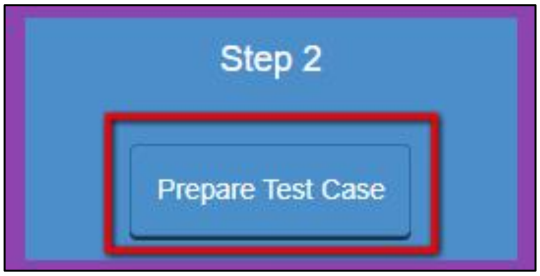
File name: ML_Calc_001.py

Save as type: Python file (*.py;*.pyw)

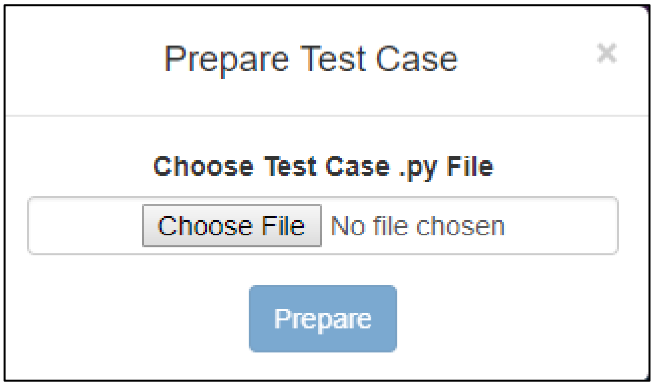
2.2 PREPARE TEST CASE

The Prepare Test Case button is used to convert the file uploaded into Python file and save in the auto created folder.

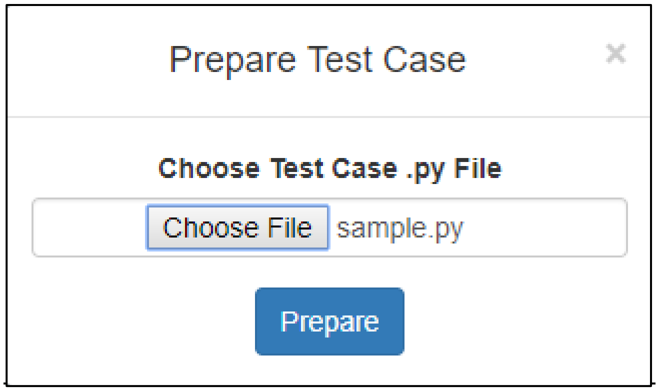
1. Click the **Prepare Test Case** button.



2. On clicking on the button, a pop up appears.



3. Click on the **Choose File** button.
4. Surfing the folder location where saved the Python file that was saved in the previous step.
5. Now upload the Python file.
6. The user can upload **only one file** at a time.
7. On uploading the Python file, the **Prepare** button becomes active.

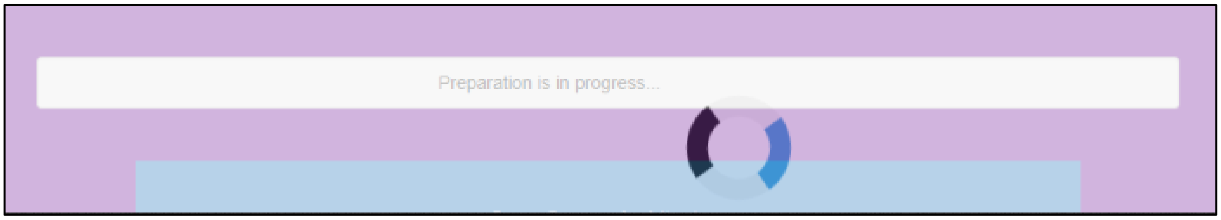


8. On uploading JSON file and clicking the **Prepare** button, the following will happen at the backend.

- It creates a folder in the name of the test case that was mentioned in the previous step.
- The folder gets stored in:

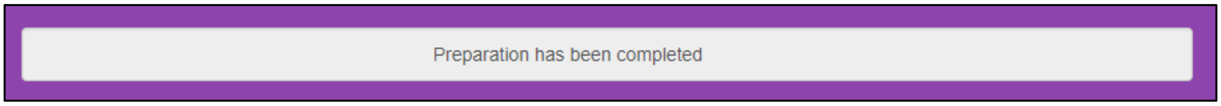
  > TON > MobileTesting > GUI > Demo_TON > <TestCase_Name_Folder>.
- In case if the test case name was saved as **MF_Calc _001** in the previous step then it will create a folder with same name **MF_ Calc _001**.
- In this folder, the Python file that was uploaded will be found.

9. On clicking of the **Prepare** button, a message will be shown as **"Preparation is in Progress"** in the status bar. And until the preparation is complete, a spinner will be loading stating that the preparation is still on.



Preparation is in progress...

10. Once preparation is finished, a message will be shown as **"Preparation has been completed"** in the status bar.

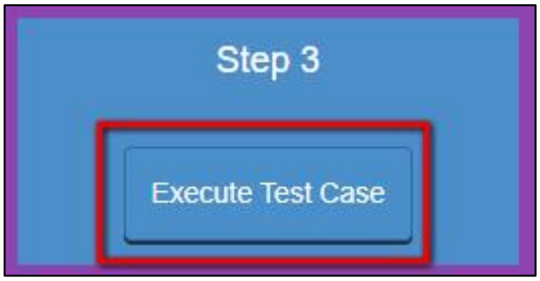


Preparation has been completed

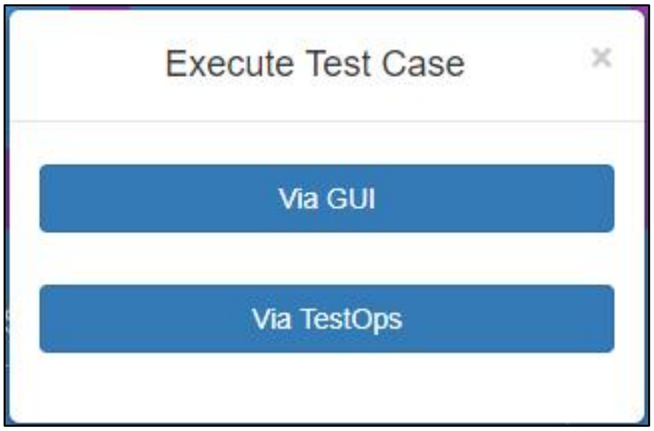
2.3 EXECUTE TEST CASE

The Execute Test Case button is used for the execution process. The execution process takes place using RIDE.

1. Click on the **Execute Test Case** button.

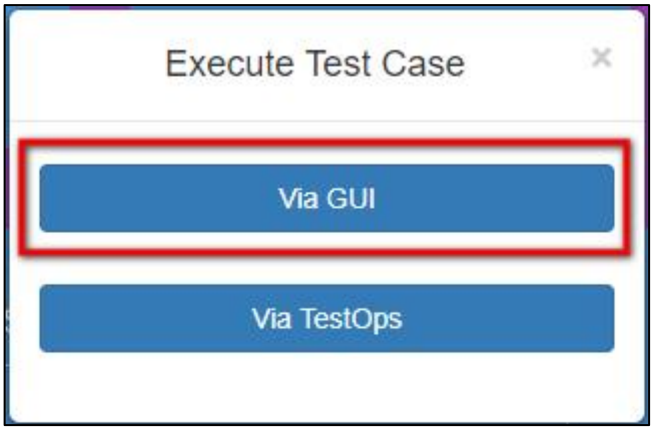


2. On clicking on the button, a pop up appears.
3. From the pop-up, select the button either as **Via GUI, Via TESTOPS.**

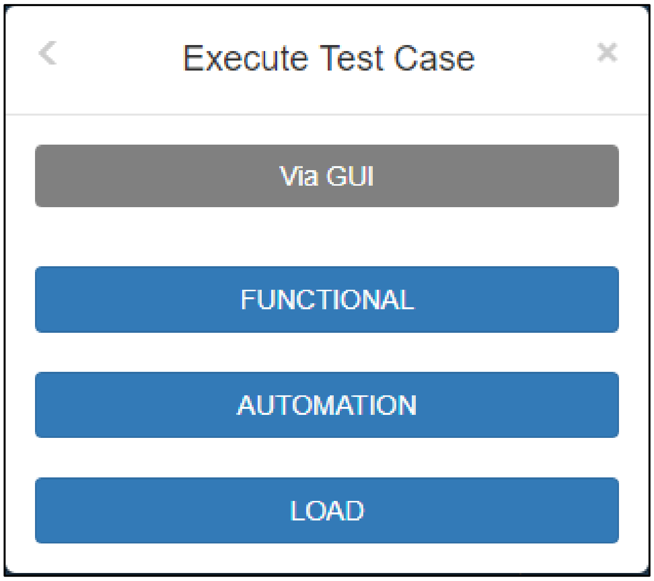


2.3.1 EXECUTE – VIA GUI

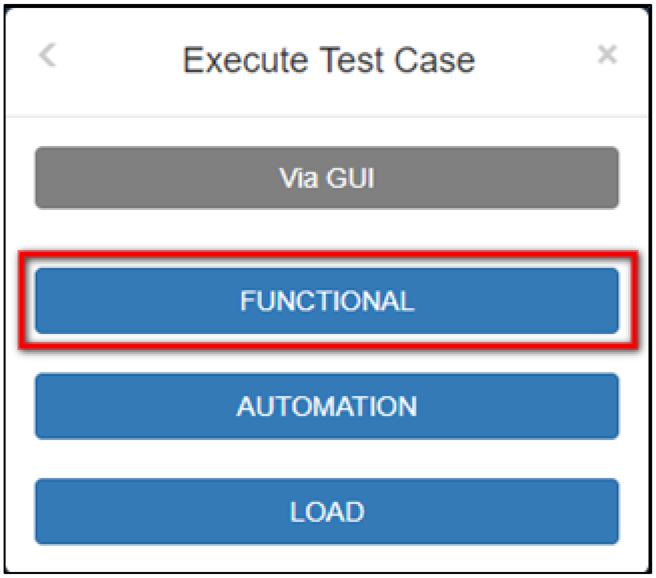
1. Click the **Via GUI** button.



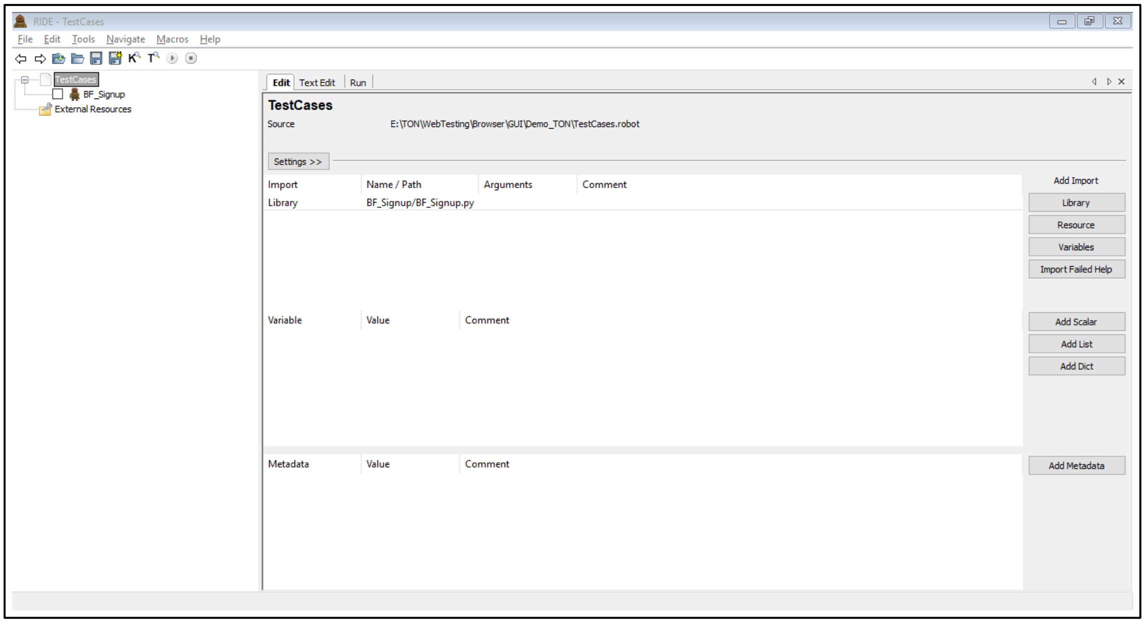
2. On clicking the button, 3 buttons appear – **Functional, Automation and Load**.



3. To perform functional testing, press Functional Button.



4. On click of the button, **RIDE** will open.



TestCases

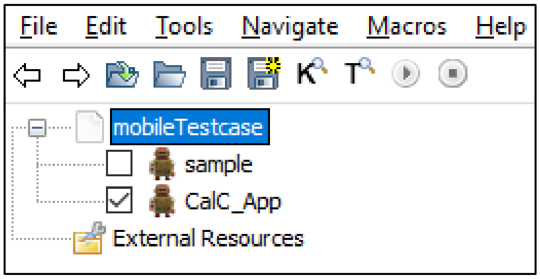
Source E:\TON\WebTesting\Browser\GUI\Demo_TON\TestCases.robot

Import	Name / Path	Arguments	Comment
Library	BF_Signup/BF_Signup.py		

Variable	Value	Comment

Metadata	Value	Comment

5. On the left panel, select one test case to execute.



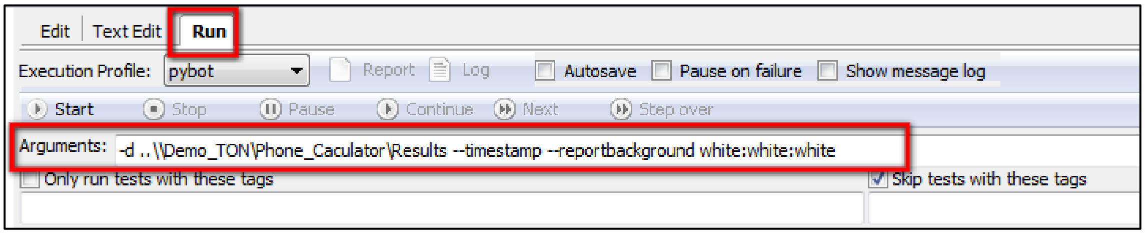
6. To configure the parameters, click on the test case name.

7. On click of the test case name, the **Edit** tab opens.

8. On the top, click **Run** Tab.

9. At the top of the Run tab, configure the **Arguments** textbox by entering:

   -d ..\\Demo_TON\<test_case_name>\Results --timestamp --reportbackground white:white:white -t



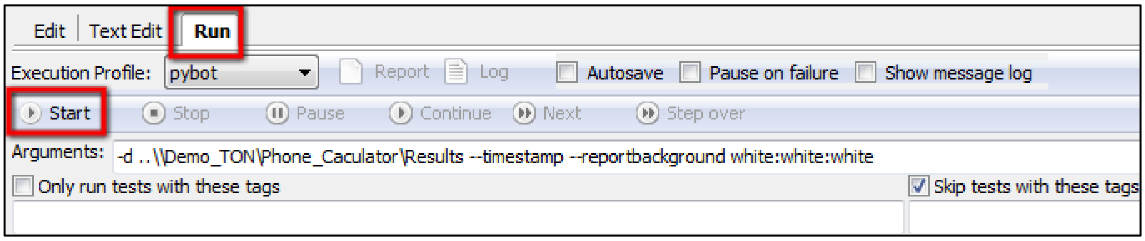
Execution Profile: pybot

□ Autosave □ Pause on failure □ Show message log

Arguments: -d ..\\Demo_TON\Phone_Caculator\Results --timestamp --reportbackground white:white:white

□ Only run tests with these tags ☑ Skip tests with these tags

10. Under Run tab, click **Start** button to start the execution.



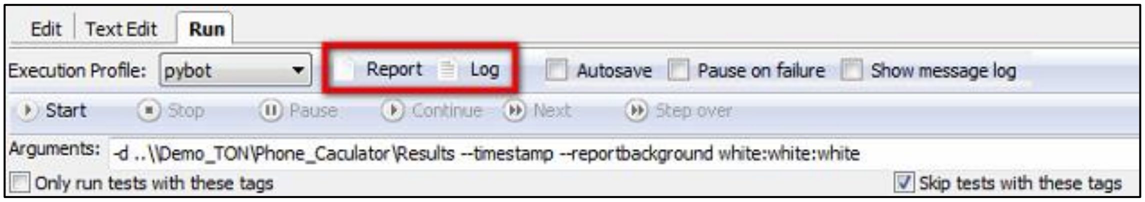
Execution Profile: pybot

□ Autosave □ Pause on failure □ Show message log

Arguments: -d ..\\Demo_TON\Phone_Caculator\Results --timestamp --reportbackground white:white:white

□ Only run tests with these tags ☑ Skip tests with these tags

11. Click on the **Log / Report** button to view the result. The button becomes active on executing the test case.



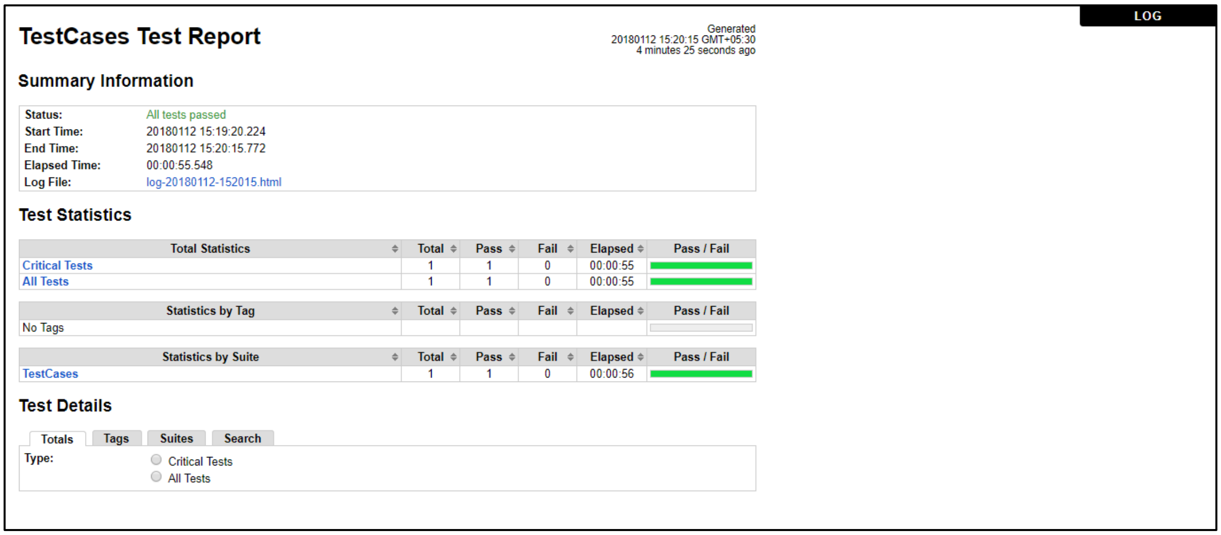
Execution Profile: pybot

□ Autosave □ Pause on failure □ Show message log

Arguments: -d ..\\Demo_TON\Phone_Caculator\Results --timestamp --reportbackground white:white:white

□ Only run tests with these tags ☑ Skip tests with these tags

12. On click of the Report button, the report opens in the browser.



TestCases Test Report

Generated
20180112 15:20:15 GMT+05:30
4 minutes 25 seconds ago

Summary Information

Status:	All tests passed
Start Time:	20180112 15:19:20.224
End Time:	20180112 15:20:15.772
Elapsed Time:	00:00:55.548
Log File:	log-20180112-152015.html

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:00:55	🟩
All Tests	1	1	0	00:00:55	🟩

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

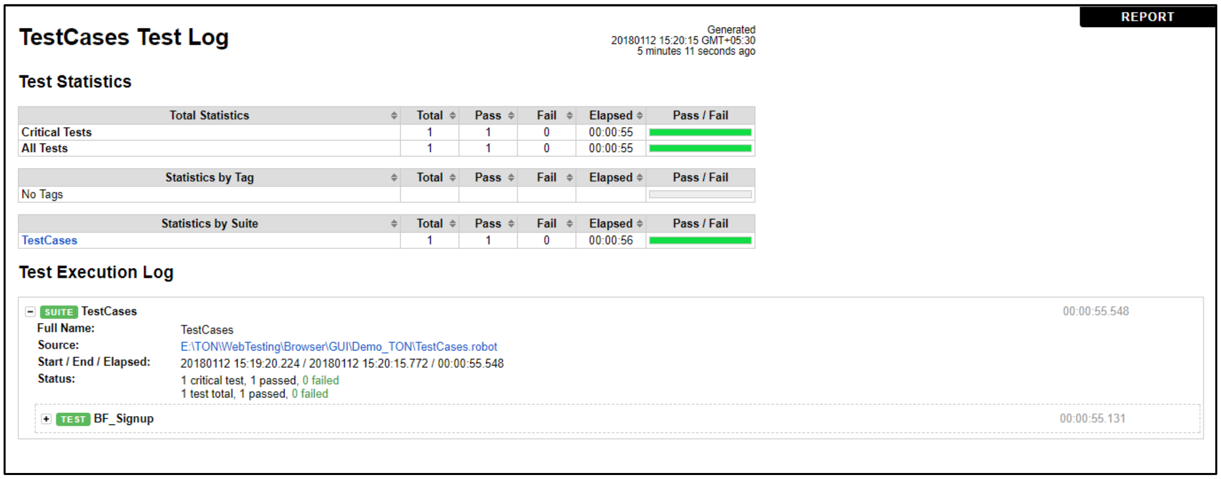
Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
TestCases	1	1	0	00:00:56	🟩

Test Details

Totals | Tags | Suites | Search

Type: ○ Critical Tests ○ All Tests

13. In the browser on the top right corner, the log button is there. Clicking on it, will show the log report and vice versa.



TestCases Test Log

Generated
20180112 15:20:15 GMT+05:30
5 minutes 11 seconds ago

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:00:55	🟩
All Tests	1	1	0	00:00:55	🟩

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
TestCases	1	1	0	00:00:56	🟩

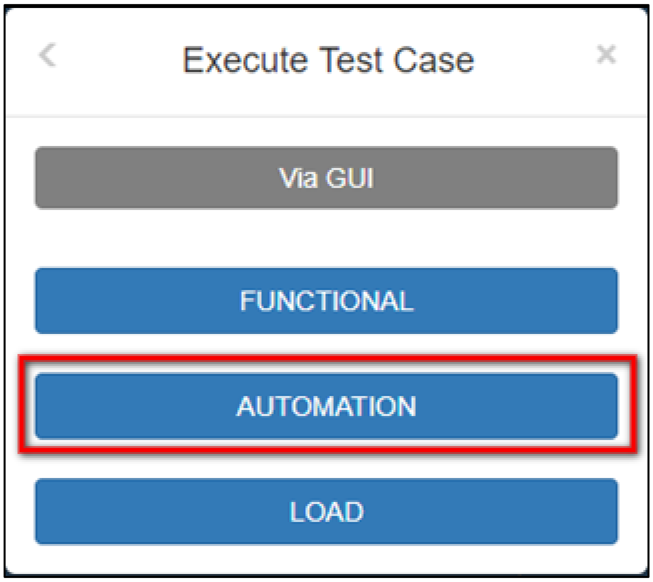
Test Execution Log

− SUITE TestCases 00:00:55.548

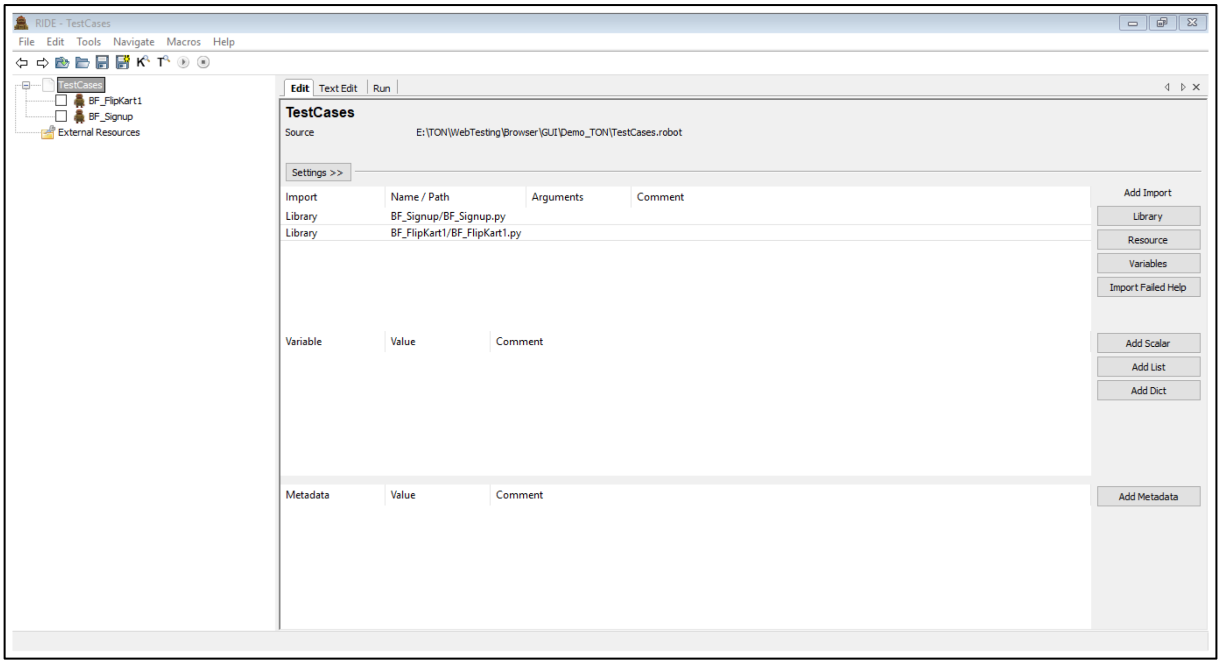
Full Name: TestCases
Source: E:\TON\WebTesting\Browser\GUI\Demo_TON\TestCases.robot
Start / End / Elapsed: 20180112 15:19:20.224 / 20180112 15:20:15.772 / 00:00:55.548
Status: 1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed

+ TEST BF_Signup 00:00:55.131

14. To perform automation testing, click **Automation** button.



15. On click of the button, **RIDE** will open.



TestCases

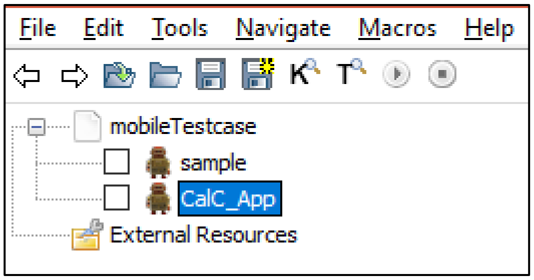
Source E:\TON\WebTesting\Browser\GUI\Demo_TON\TestCases.robot

Import	Name / Path	Arguments	Comment
Library	BF_Signup/BF_Signup.py		
Library	BF_FlipKart1/BF_FlipKart1.py		

Variable	Value	Comment

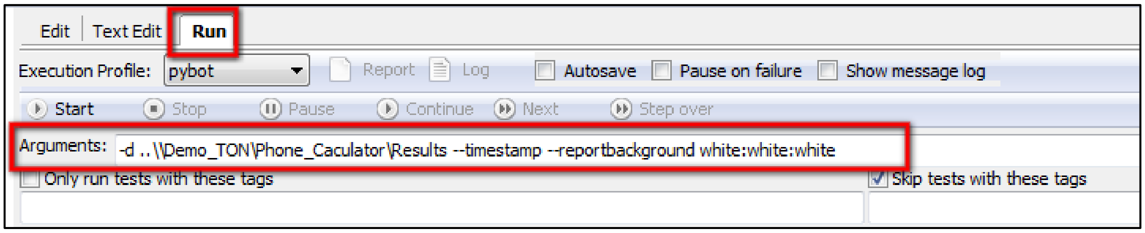
Metadata	Value	Comment

16. On the left panel, select multiple test cases to execute.



17. To configure the parameters, click on the test case name.
18. On click of the test case name, the **Edit** tab opens.
19. On the top, click **Run** Tab.
20. At the top of the Run tab, configure the **Arguments** textbox by entering:

    -d ..\\Demo_TON\<test_case_name>\Results --timestamp --reportbackground white:white:white -t



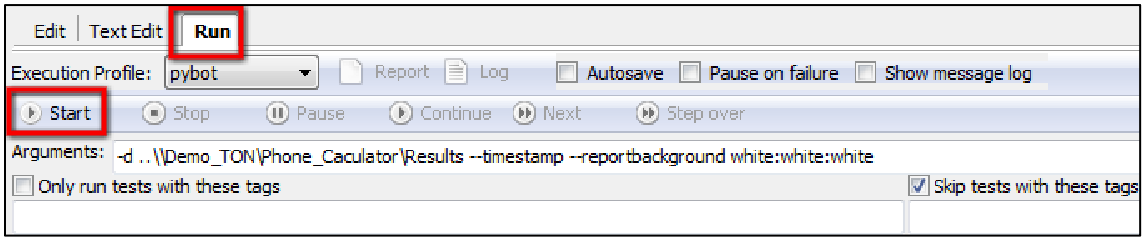
Execution Profile: pybot

□ Autosave □ Pause on failure □ Show message log

Arguments: -d ..\\Demo_TON\Phone_Caculator\Results --timestamp --reportbackground white:white:white

□ Only run tests with these tags ☑ Skip tests with these tags

21. Under Run tab, click **Start** button to start the execution.



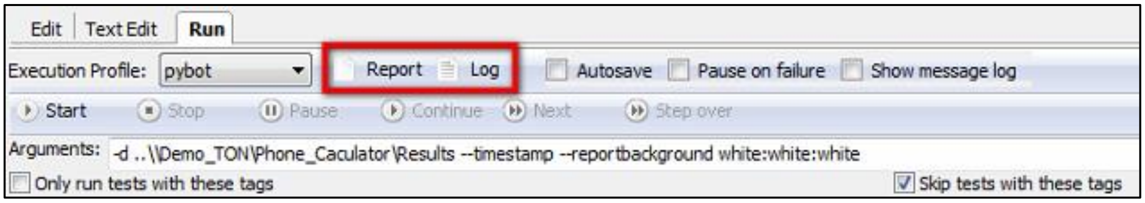
Execution Profile: pybot

□ Autosave □ Pause on failure □ Show message log

Arguments: -d ..\\Demo_TON\Phone_Caculator\Results --timestamp --reportbackground white:white:white

□ Only run tests with these tags ☑ Skip tests with these tags

22. Click on the **Log / Report** button to view the result.



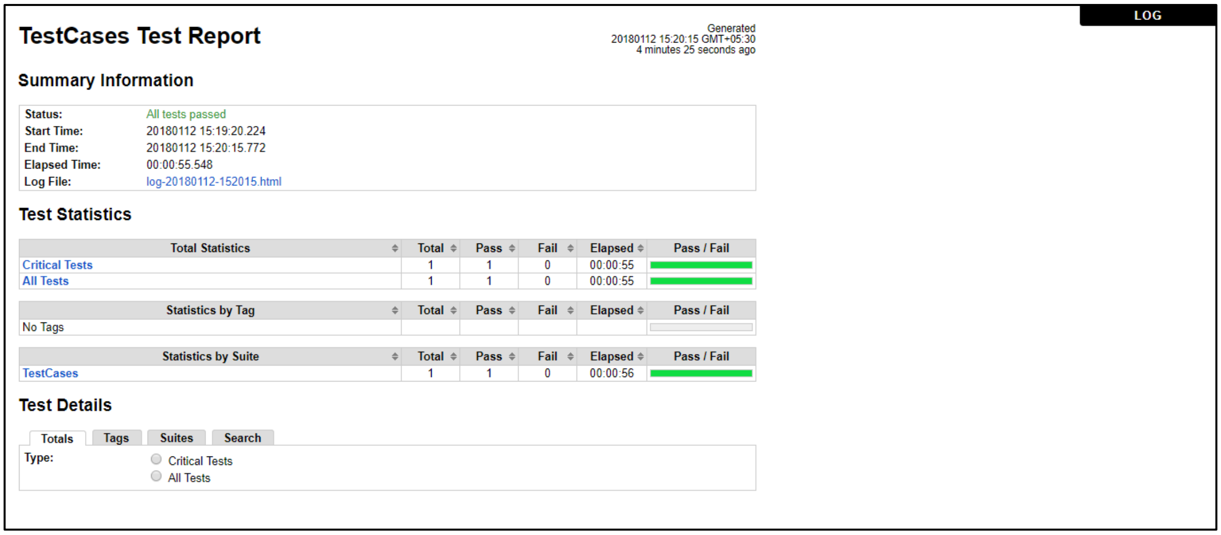
Execution Profile: pybot

□ Autosave □ Pause on failure □ Show message log

Arguments: -d ..\\Demo_TON\Phone_Caculator\Results --timestamp --reportbackground white:white:white

□ Only run tests with these tags ☑ Skip tests with these tags

23. On click of the Report button, in the browser, the report appears.



TestCases Test Report

Generated
20180112 15:20:15 GMT+05:30
4 minutes 25 seconds ago

Summary Information

Status:	All tests passed
Start Time:	20180112 15:19:20.224
End Time:	20180112 15:20:15.772
Elapsed Time:	00:00:55.548
Log File:	log-20180112-152015.html

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:00:55	🟩
All Tests	1	1	0	00:00:55	🟩

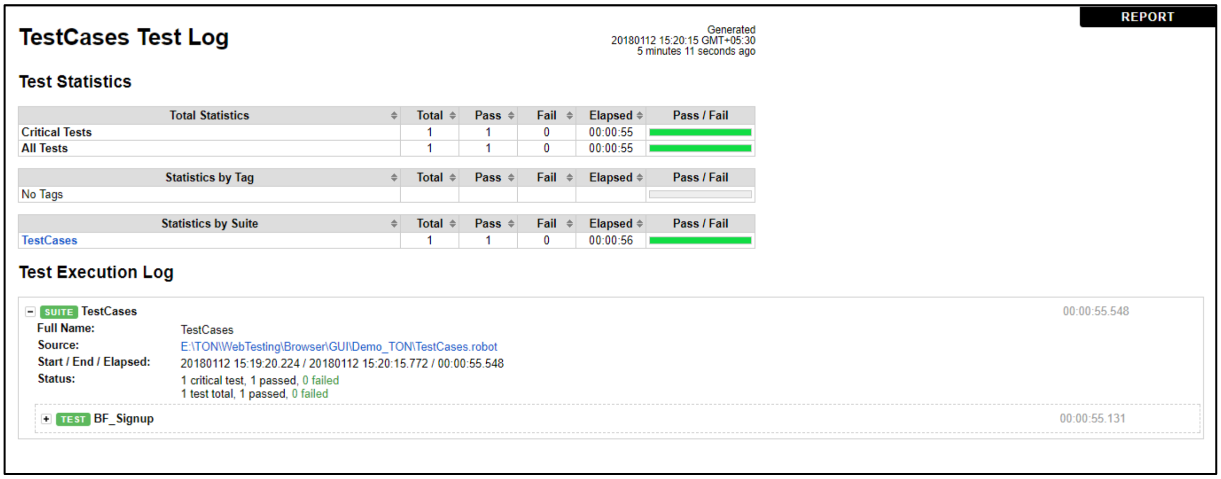
Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
TestCases	1	1	0	00:00:56	🟩

Test Details

Type: ○ Critical Tests ○ All Tests

24. In the browser on the top right corner, the log button is there. Clicking on it, will show the log report and vice versa.



TestCases Test Log

Generated
20180112 15:20:15 GMT+05:30
5 minutes 11 seconds ago

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:00:55	🟩
All Tests	1	1	0	00:00:55	🟩

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
TestCases	1	1	0	00:00:56	🟩

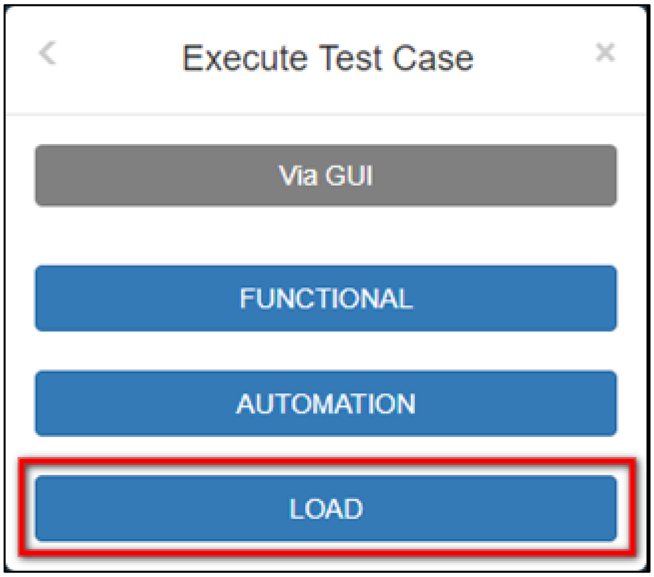
Test Execution Log

− SUITE TestCases 00:00:55.548

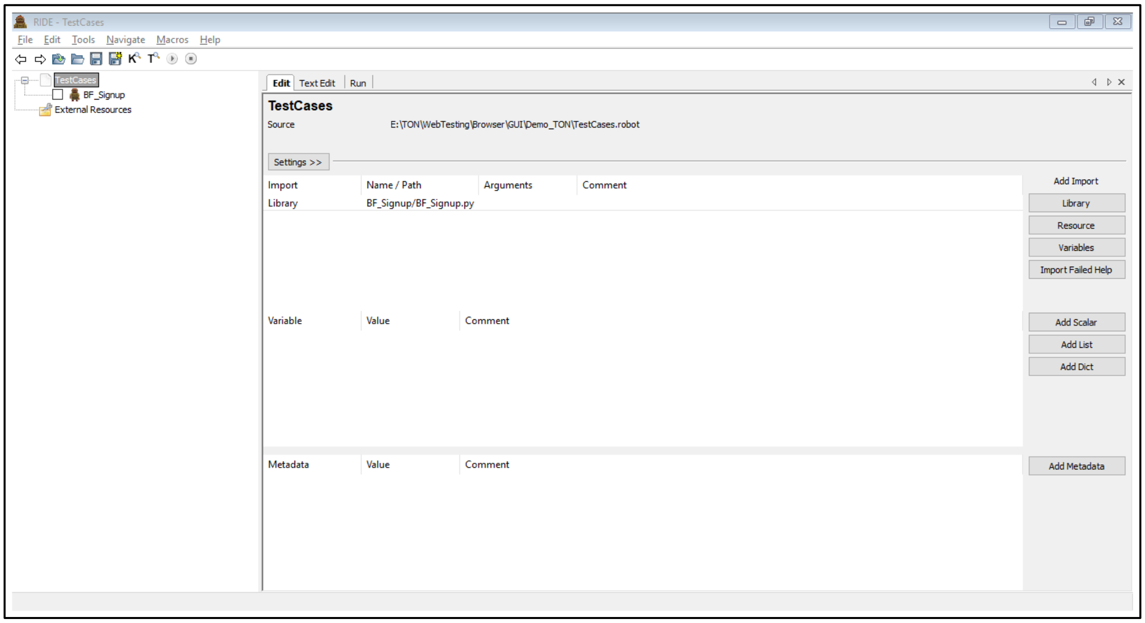
Full Name: TestCases
Source: E:\TON\WebTesting\Browser\GUI\Demo_TON\TestCases.robot
Start / End / Elapsed: 20180112 15:19:20.224 / 20180112 15:20:15.772 / 00:00:55.548
Status: 1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed

+ TEST BF_Signup 00:00:55.131

25. To perform load testing, click **Load** button.



26. On click of the button, **RIDE** will open.



TestCases

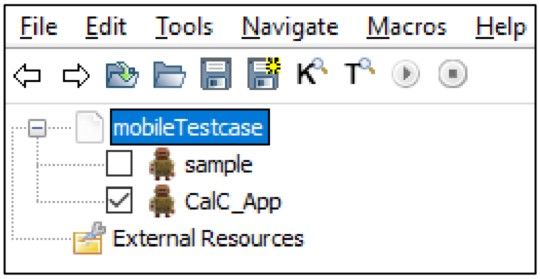
Source E:\TON\WebTesting\Browser\GUI\Demo_TON\TestCases.robot

Import	Name / Path	Arguments	Comment
Library	BF_Signup/BF_Signup.py		

Variable	Value	Comment

Metadata	Value	Comment

27. On the left panel, select one test case to execute.



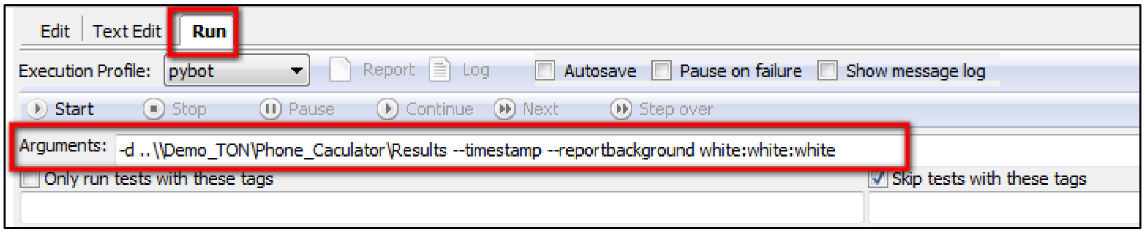
28. To configure the parameters, click on the test case name.

29. On click of the test case name, the **Edit** tab opens.

30. On the top, click **Run** Tab.

31. At the top of the Run tab, configure the **Arguments** textbox by entering:

-d ..\\Demo_TON\<test_case_name>\Results --timestamp --reportbackground white:white:white -t



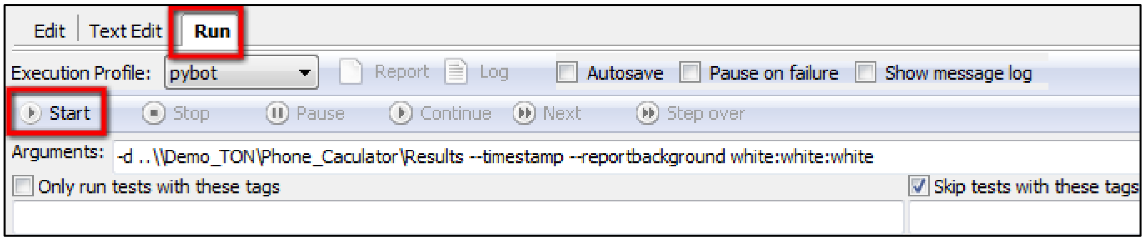
Execution Profile: pybot

☐ Autosave ☐ Pause on failure ☐ Show message log

Arguments: -d ..\\Demo_TON\Phone_Caculator\Results --timestamp --reportbackground white:white:white

☐ Only run tests with these tags ☑ Skip tests with these tags

32. Under Run tab, click **Start** button to start the execution.



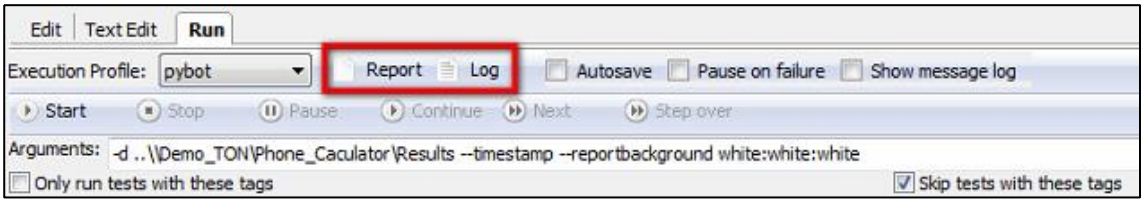
Execution Profile: pybot

☐ Autosave ☐ Pause on failure ☐ Show message log

Arguments: -d ..\\Demo_TON\Phone_Caculator\Results --timestamp --reportbackground white:white:white

☐ Only run tests with these tags ☑ Skip tests with these tags

33. Click on the **Log / Report** button to view the result. The button becomes active on executing the test case.



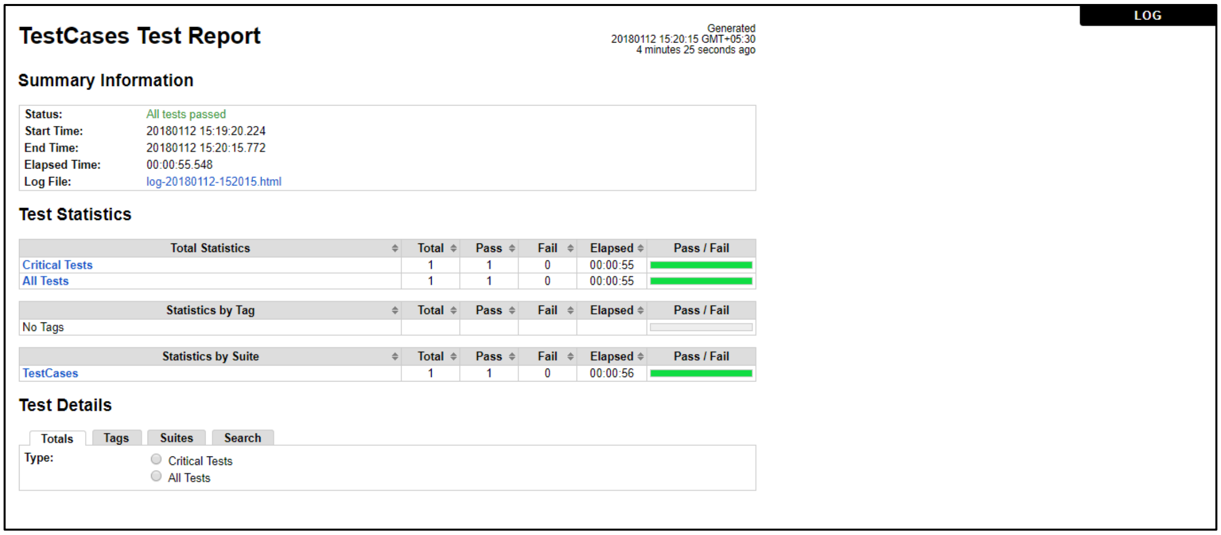
Execution Profile: pybot

☐ Autosave ☐ Pause on failure ☐ Show message log

Arguments: -d ..\\Demo_TON\Phone_Caculator\Results --timestamp --reportbackground white:white:white

☐ Only run tests with these tags ☑ Skip tests with these tags

34. On click of the Report button, in the browser, the report appears.



TestCases Test Report

Generated
20180112 15:20:15 GMT+05:30
4 minutes 25 seconds ago

Summary Information

Status:	All tests passed
Start Time:	20180112 15:19:20.224
End Time:	20180112 15:20:15.772
Elapsed Time:	00:00:55.548
Log File:	log-20180112-152015.html

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:00:55	🟩
All Tests	1	1	0	00:00:55	🟩

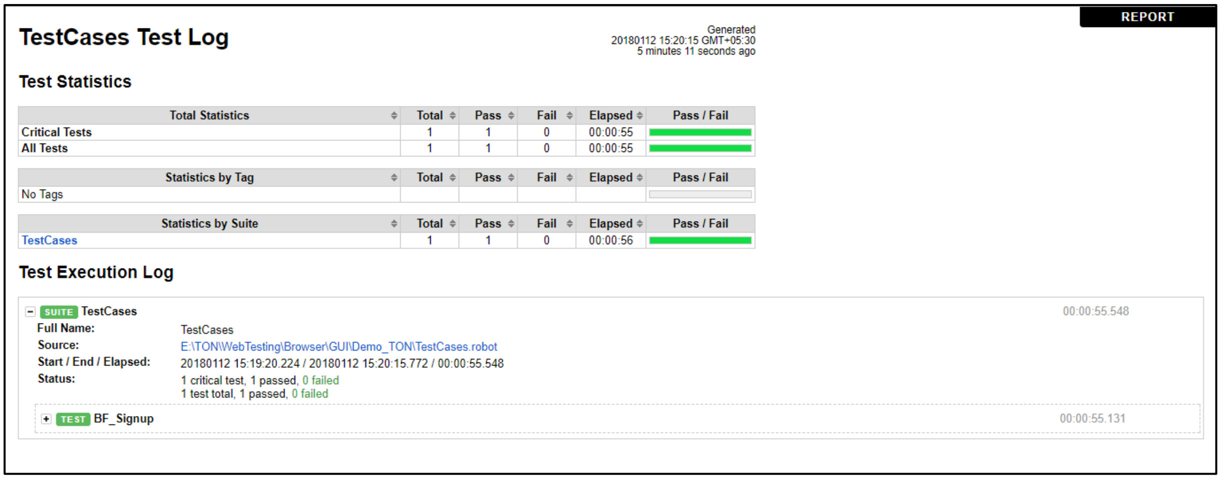
Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
TestCases	1	1	0	00:00:56	🟩

Test Details

Type: ○ Critical Tests
○ All Tests

35. In the browser on the top right corner, the log button is there. Clicking on it, will show the log report and vice versa.



TestCases Test Log

Generated
20180112 15:20:15 GMT+05:30
5 minutes 11 seconds ago

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:00:55	🟩
All Tests	1	1	0	00:00:55	🟩

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

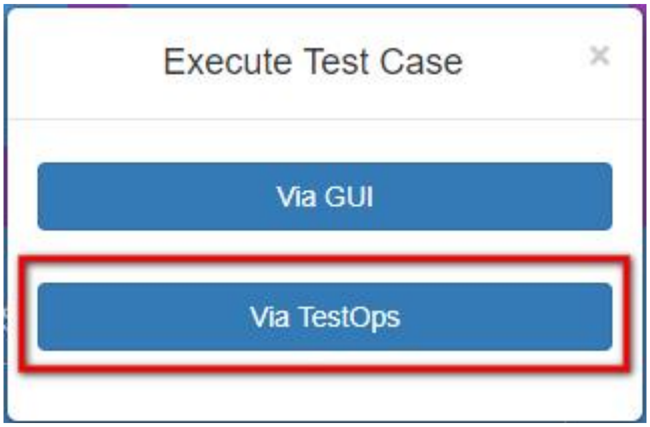
Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
TestCases	1	1	0	00:00:56	🟩

Test Execution Log

```
− SUITE TestCases                                                          00:00:55.548
  Full Name:            TestCases
  Source:               E:\TON\WebTesting\Browser\GUI\Demo_TON\TestCases.robot
  Start / End / Elapsed: 20180112 15:19:20.224 / 20180112 15:20:15.772 / 00:00:55.548
  Status:               1 critical test, 1 passed, 0 failed
                        1 test total, 1 passed, 0 failed
  + TEST BF_Signup                                                         00:00:55.131
```

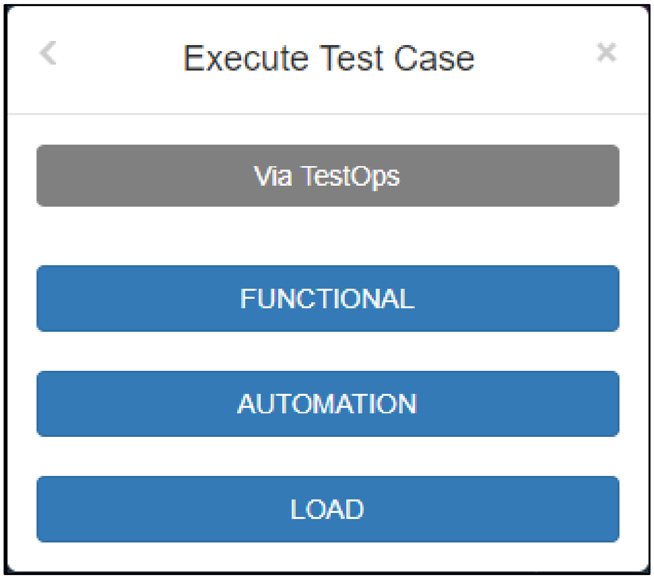
2.3.2 EXECUTE - VIA TESTOPS

1. Click the **Via TestOps** button.



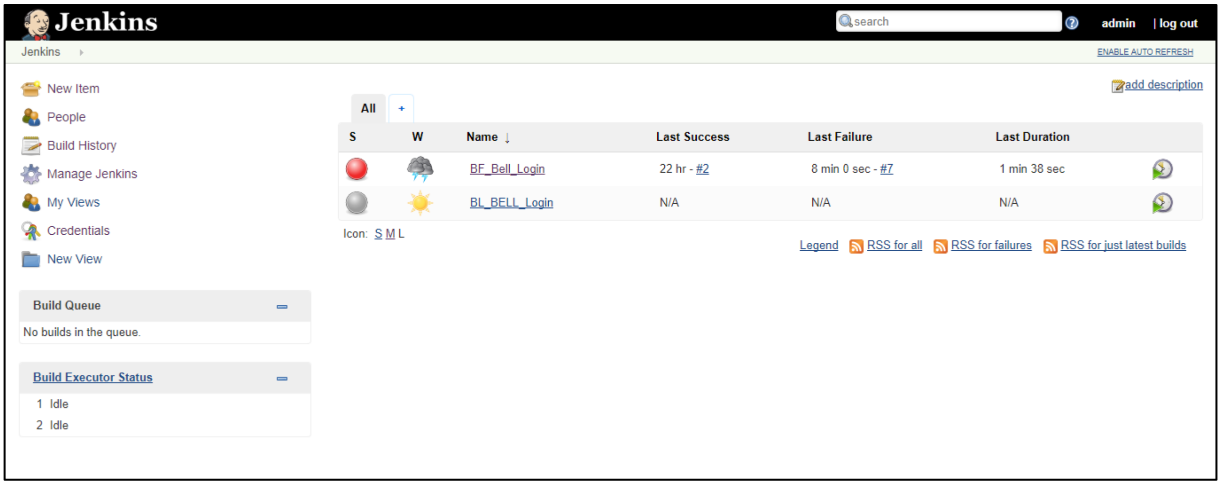
Execute Test Case

2. On clicking the button, 3 buttons will appear – **Functional, Automation and Load**.



Execute Test Case

3. On click of any one button, the **Jenkins** will open.



Jenkins | search | admin | log out

Jenkins ▸ | ENABLE AUTO REFRESH

- New Item
- People
- Build History
- Manage Jenkins
- My Views
- Credentials
- New View

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

add description

All | +

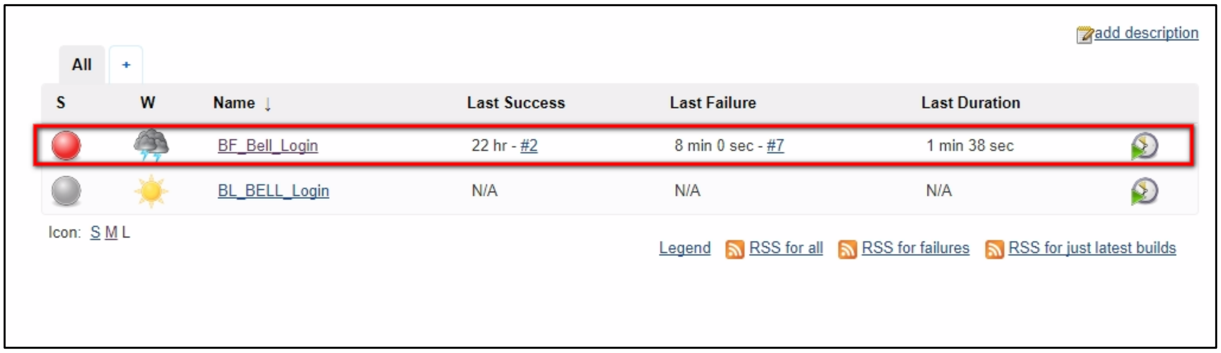
S	W	Name ↓	Last Success	Last Failure	Last Duration	
●		BF_Bell_Login	22 hr - #2	8 min 0 sec - #7	1 min 38 sec	
●		BL_BELL_Login	N/A	N/A	N/A	

Icon: S M L

Legend | RSS for all | RSS for failures | RSS for just latest builds

For creation of new View and Job, refer the Reference Section below.

4. Click on the job that has to be executed.



add description

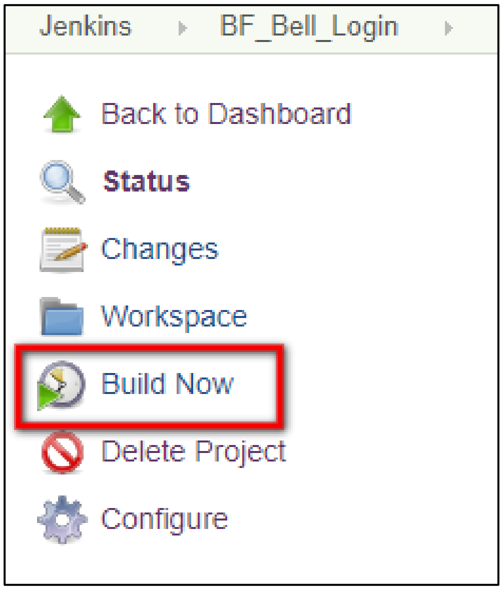
All | +

S	W	Name ↓	Last Success	Last Failure	Last Duration	
●		BF_Bell_Login	22 hr - #2	8 min 0 sec - #7	1 min 38 sec	
●		BL_BELL_Login	N/A	N/A	N/A	

Icon: S M L

Legend | RSS for all | RSS for failures | RSS for just latest builds

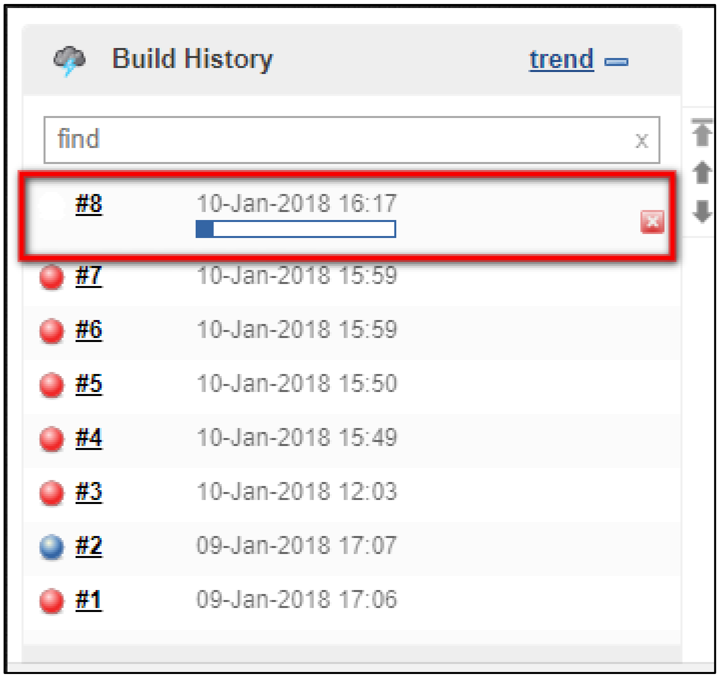
5. On the left side, select Build Now.



Jenkins ▸ BF_Bell_Login ▸

- Back to Dashboard
- Status
- Changes
- Workspace
- Build Now
- Delete Project
- Configure

6. On the left side, under Build History, the progress of the job is shown.

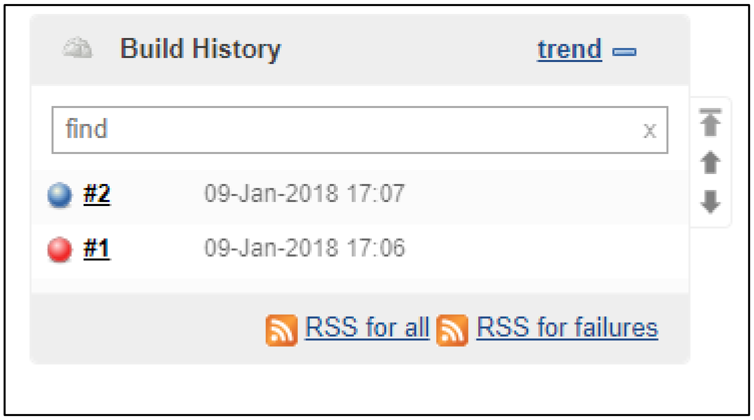


Build History trend

find

	#8	10-Jan-2018 16:17
🔴	#7	10-Jan-2018 15:59
🔴	#6	10-Jan-2018 15:59
🔴	#5	10-Jan-2018 15:50
🔴	#4	10-Jan-2018 15:49
🔴	#3	10-Jan-2018 12:03
🔵	#2	09-Jan-2018 17:07
🔴	#1	09-Jan-2018 17:06

7. If the job is successfully built, it will show the built time, number of times of build and status of the built.



Build History trend

find

🔵	#2	09-Jan-2018 17:07
🔴	#1	09-Jan-2018 17:06

RSS for all RSS for failures

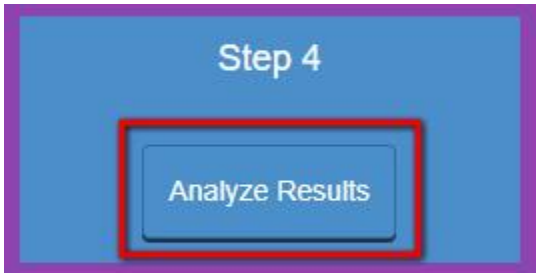
8. All the jobs of the Jenkins will be saved in:

> TON > WebTesting > Browser > TestOps > Demo_TON > <Job_Name_Folder>

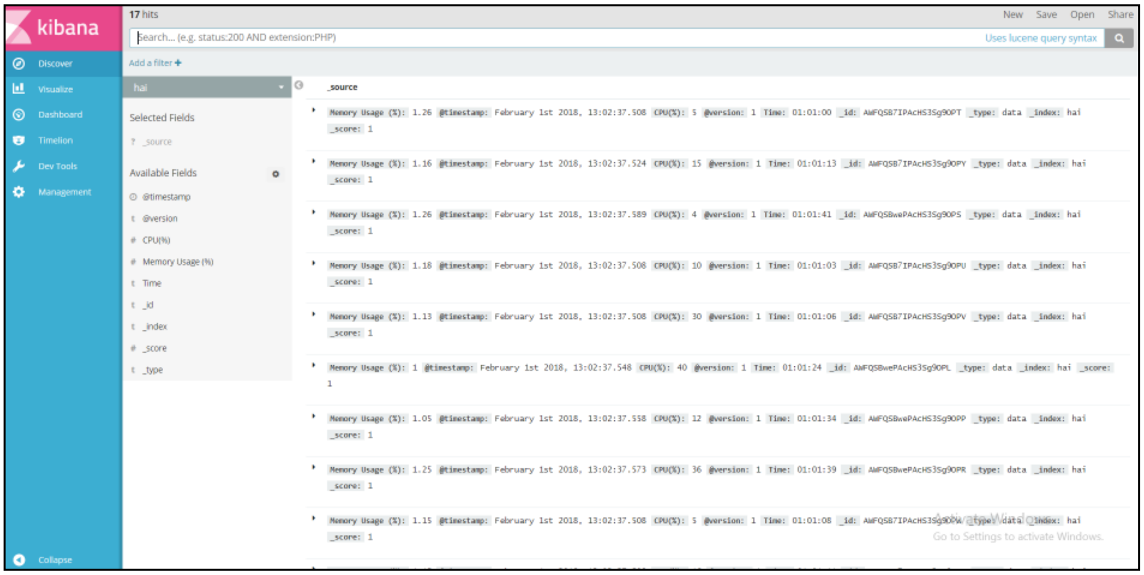
2.4 ANALYZE RESULTS

The **Analyze Results** button is used to analyze the final result of the test case that had been recorded and executed in the previous steps. The final result will be displayed in the form of graphs

1. Click on the **Analyze Results**.



2. On clicking on the **Analyze Results** button, Kibana opens up in a new window.



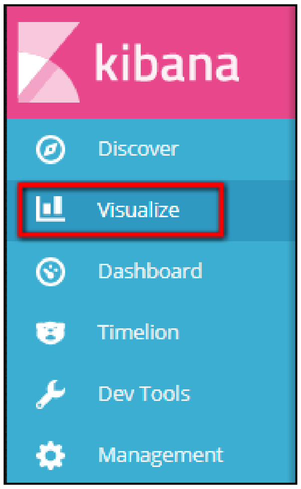
17 hits

Search... (e.g. status:200 AND extension:PHP)

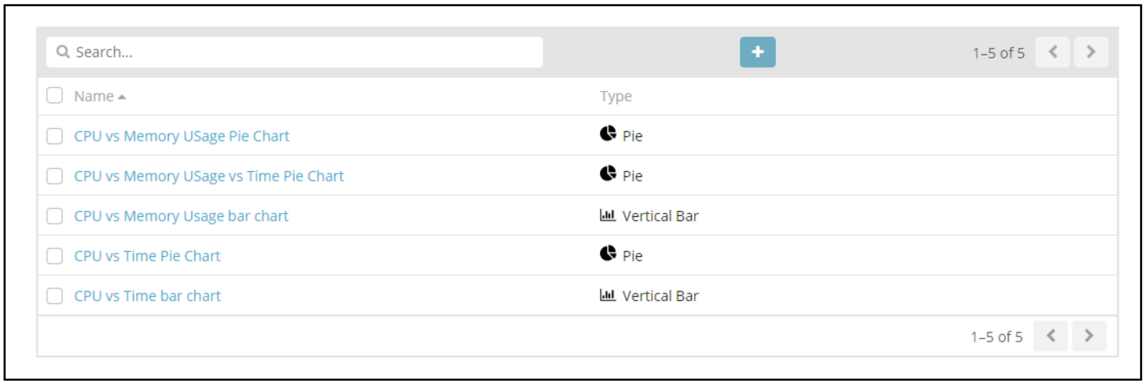
```
_source
Memory Usage (%): 1.26 @timestamp: February 1st 2018, 13:02:37.508 CPU(%): 5 @version: 1 Time: 01:01:00 _id: AWFQSB7IPAcHS3Sg9OPT _type: data _index: hai
_score: 1
Memory Usage (%): 1.16 @timestamp: February 1st 2018, 13:02:37.524 CPU(%): 15 @version: 1 Time: 01:01:13 _id: AWFQSB7IPAcHS3Sg9OPY _type: data _index: hai
_score: 1
Memory Usage (%): 1.26 @timestamp: February 1st 2018, 13:02:37.589 CPU(%): 4 @version: 1 Time: 01:01:41 _id: AWFQSBwePAcHS3Sg9OPS _type: data _index: hai
_score: 1
Memory Usage (%): 1.18 @timestamp: February 1st 2018, 13:02:37.508 CPU(%): 10 @version: 1 Time: 01:01:03 _id: AWFQSB7IPAcHS3Sg9OPU _type: data _index: hai
_score: 1
Memory Usage (%): 1.13 @timestamp: February 1st 2018, 13:02:37.508 CPU(%): 30 @version: 1 Time: 01:01:06 _id: AWFQSB7IPAcHS3Sg9OPV _type: data _index: hai
_score: 1
Memory Usage (%): 1 @timestamp: February 1st 2018, 13:02:37.548 CPU(%): 40 @version: 1 Time: 01:01:24 _id: AWFQSBwePAcHS3Sg9OPL _type: data _index: hai _score:
1
Memory Usage (%): 1.05 @timestamp: February 1st 2018, 13:02:37.558 CPU(%): 12 @version: 1 Time: 01:01:34 _id: AWFQSBwePAcHS3Sg9OPP _type: data _index: hai
_score: 1
Memory Usage (%): 1.25 @timestamp: February 1st 2018, 13:02:37.573 CPU(%): 36 @version: 1 Time: 01:01:39 _id: AWFQSBwePAcHS3Sg9OPR _type: data _index: hai
_score: 1
Memory Usage (%): 1.15 @timestamp: February 1st 2018, 13:02:37.508 CPU(%): 5 @version: 1 Time: 01:01:08 _id: AWFQSB7IPAcHS3Sg9OPW _type: data _index: hai
_score: 1
```

Activate Windows
Go to Settings to activate Windows.

3. On the left panel, click Visualize.

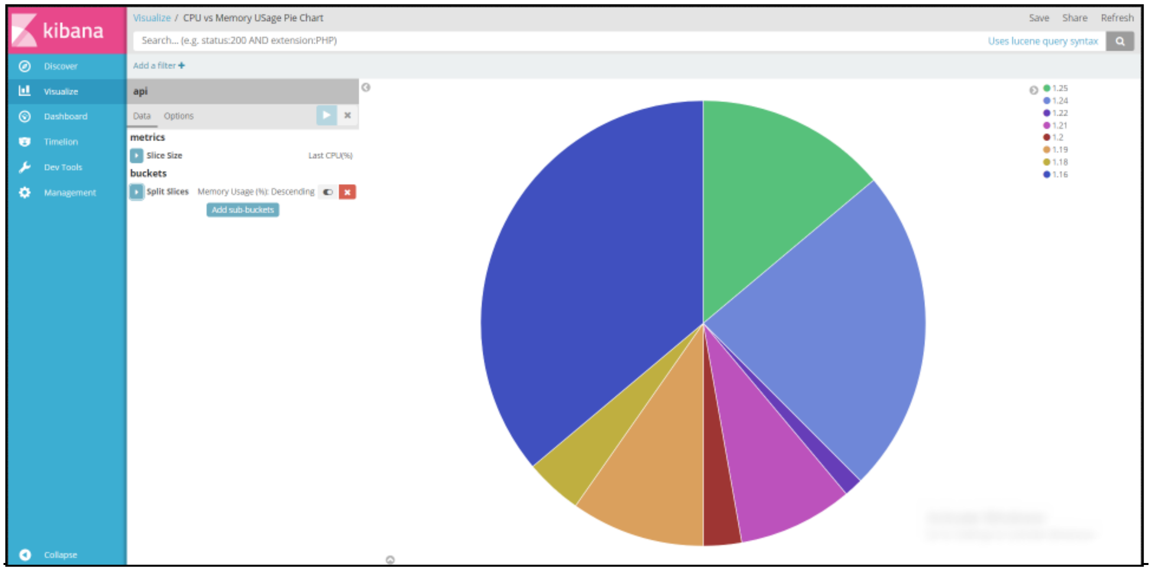


4. The user will find charts of different forms with different combinations like CPU vs Time, CPU vs Memory Usage vs Time etc.



□ Name ▲	Type
□ CPU vs Memory USage Pie Chart	Pie
□ CPU vs Memory USage vs Time Pie Chart	Pie
□ CPU vs Memory Usage bar chart	Vertical Bar
□ CPU vs Time Pie Chart	Pie
□ CPU vs Time bar chart	Vertical Bar

5. On click of the chart name, the user can see the graph based on the type of the chart. (The example shows for Pie Chart)



metrics

Slice Size — Last CPU(%)

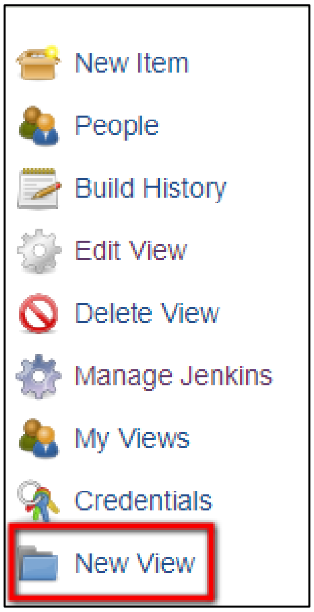
buckets

Split Slices — Memory Usage (%): Descending

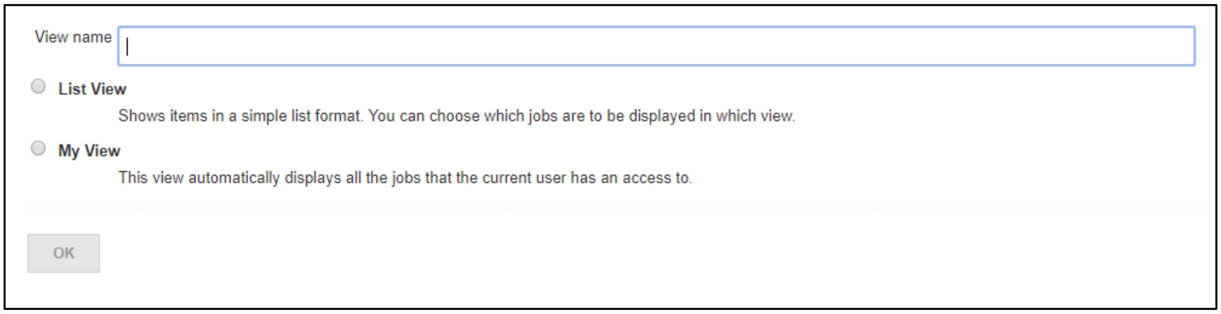
3. REFERENCES

3.1 CREATION OF NEW VIEW

- To create a new view, on the left panel, click **New View**.



- On clicking, give the name of the tab as **Every_Mobile_Testing** and select the type of view.



View name |

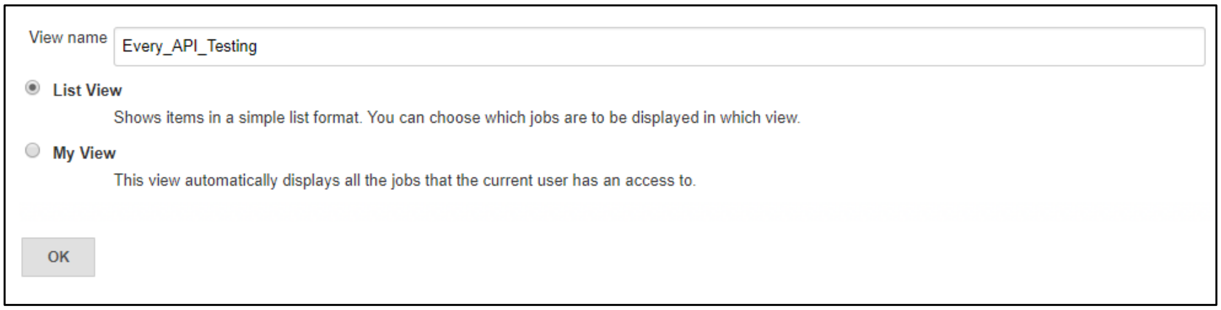
○ **List View**

Shows items in a simple list format. You can choose which jobs are to be displayed in which view.

○ **My View**

This view automatically displays all the jobs that the current user has an access to.

- On entering the name and selecting the view type, the OK button becomes active. Click Ok.



View name Every_API_Testing

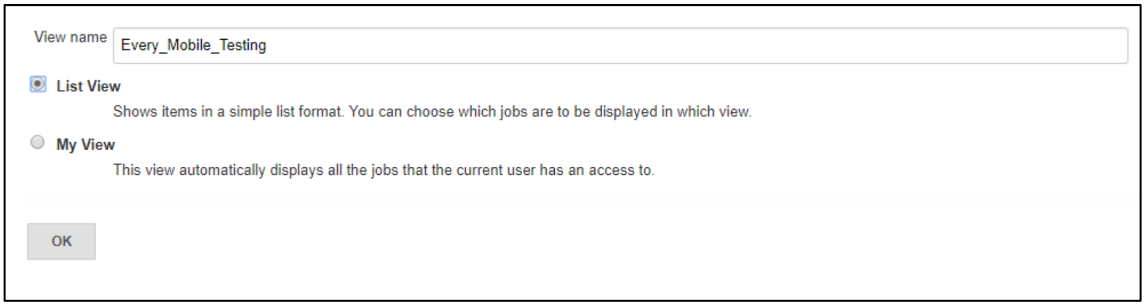
● **List View**

Shows items in a simple list format. You can choose which jobs are to be displayed in which view.

○ **My View**

This view automatically displays all the jobs that the current user has an access to.

- In the next page, select the jobs that are to be added to this tab.



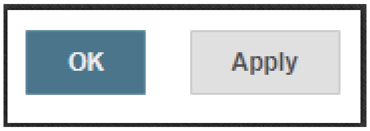
View name Every_Mobile_Testing

● **List View**
Shows items in a simple list format. You can choose which jobs are to be displayed in which view.

○ **My View**
This view automatically displays all the jobs that the current user has an access to.

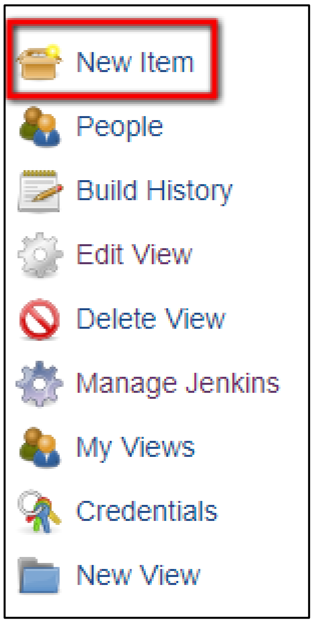
OK

- Once done, click Apply and Ok to save and Close the creation.

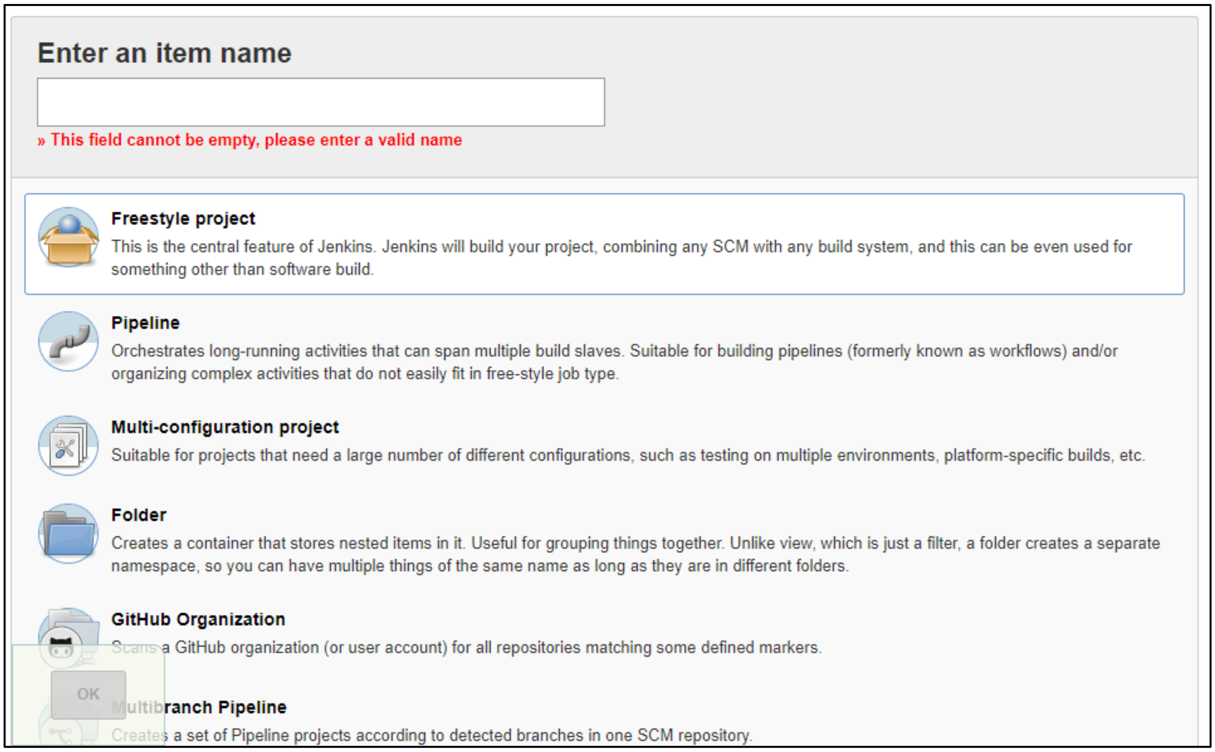


3.2 CREATION OF NEW JOB

- To create a new job, on the left panel, click **New Item**.



- Give a name for the new job that is going to be created and select the tab in which it is to be stored.



Enter an item name

» This field cannot be empty, please enter a valid name

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

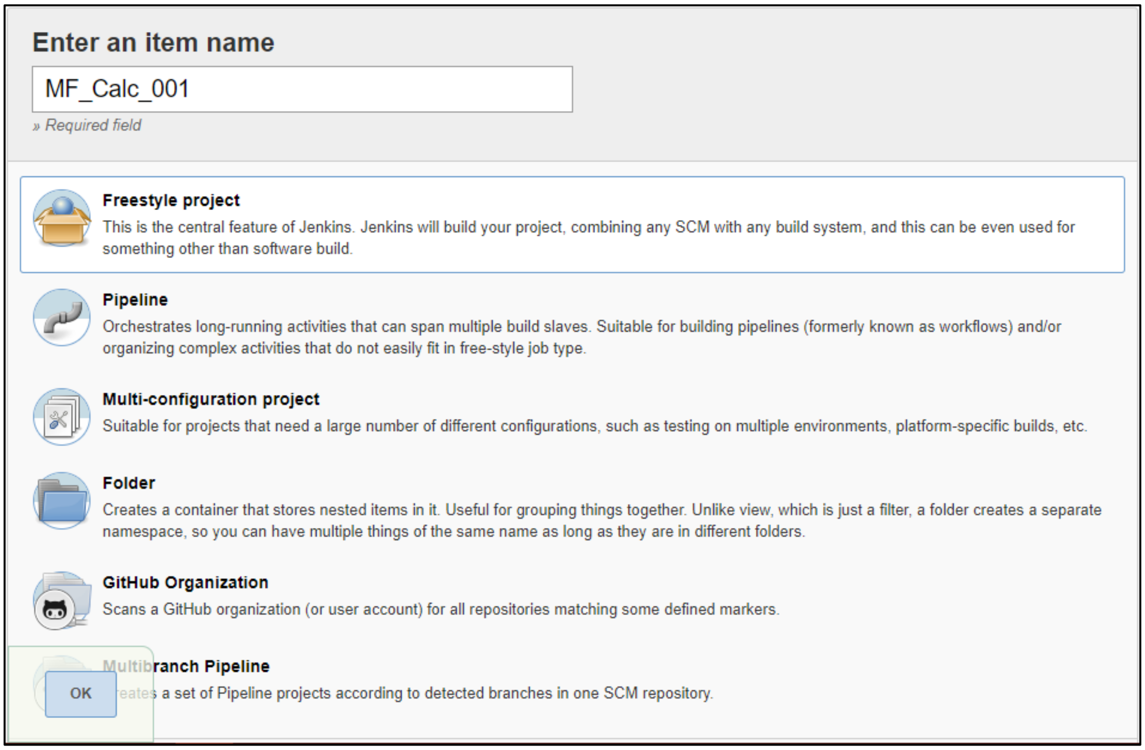
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

- Then select the type of the job that is to be created.
- Click Ok to save the job.



Enter an item name

MF_Calc_001

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

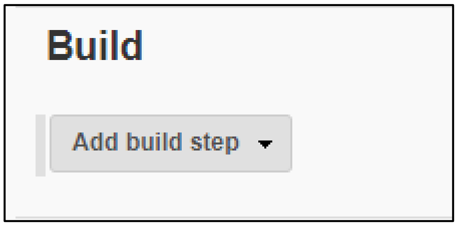
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

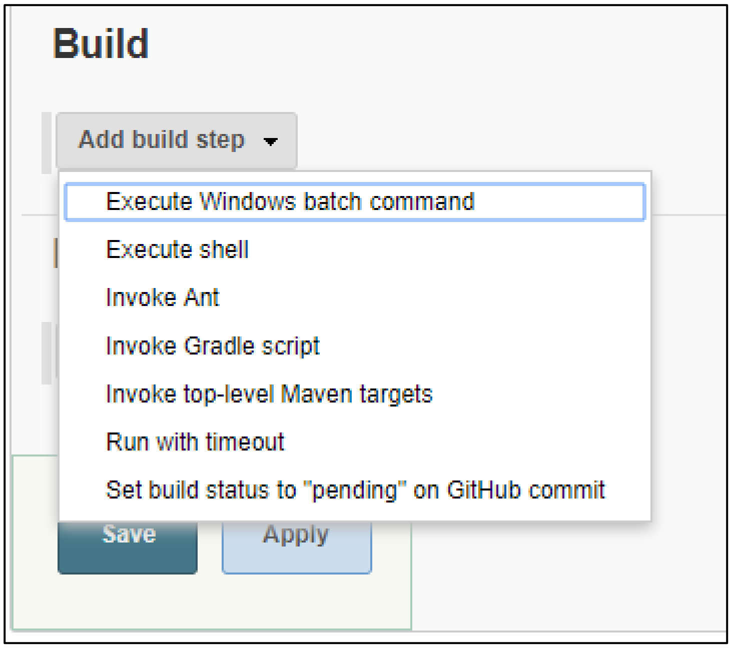
Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

- On click of Ok button, job configuration page will appear.

- Scroll down and look for Build.

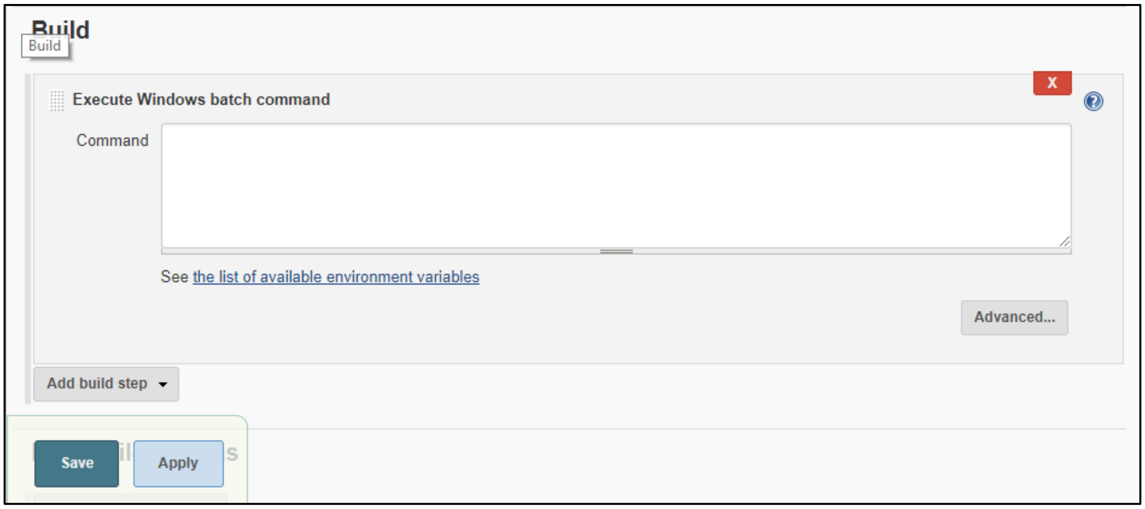


- Under build, click Add Build Setup drop down.



Build
Add build step
Execute Windows batch command
Execute shell
Invoke Ant
Invoke Gradle script
Invoke top-level Maven targets
Run with timeout
Set build status to "pending" on GitHub commit
Save
Apply

- From the drop down, select **Execute Windows Batch Command.**
- On selecting, a text area appears.


Build
Build
Execute Windows batch command
X
Command
See the list of available environment variables
Advanced...
Add build step
Save
Apply

- In the text area, write the following command.