EVERYMOBILE – USER MANUAL – ANDROID

Contact:

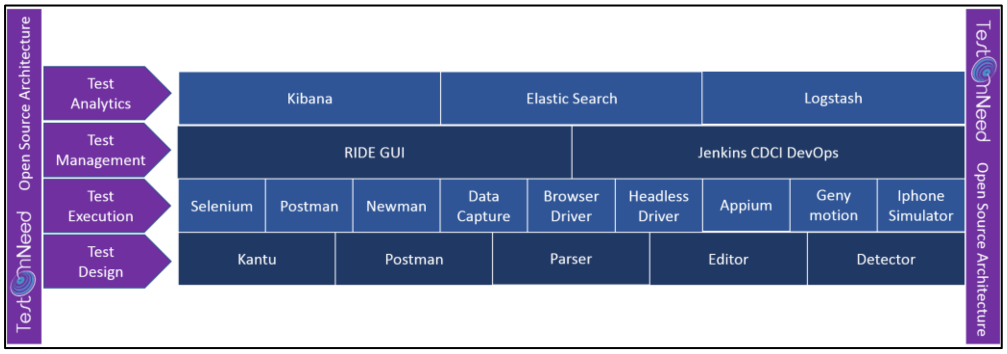
sales@testonneed.com

TABLE OF CONTENTS

1. INTRODUCTION

TestOnNeed, an Open Source Testing Ecosystem, is created for Entrepreneurs, Start-ups and Enterprises to **test** software products and solutions **‘on need’**. We have helped global startups, mid to large enterprises to deliver world-class products and solutions with strategic insights to transform and thrive in this rapidly changing world.

- TestOnNeed Open Source Ecosystem is used in the testing of the frontend GUI, backend API calls, mobile application (android and iOS) functionalities.



- This Open Source Ecosystem consists of **Test Design, Test Execution, Test Management and Test Analytics**.
- Each stage of Open Source Ecosystem uses set of open sources that are useful in their own way to achieve product quality.
- **Test Design**
  - In this step, the development or creating of the test cases are done.
  - Kantu, Postman, Parser applications, Editor and Detector are the open sources that are being used.
- **Test Execution**
  - In the step, the execution of the developed or created test cases take place.
  - Selenium, Postman, Newman, Data Capture, Browser Driver, Headless Driver, Appium, Genymotion and iPhone Simulator are the different open sources that are being used.
- **Test Management**
  - In this step, the test cases are given and modified the arguments and parameters that it produces desired result of the test cases.
  - RIDE GUI and Jenkins are the open sources that are being in this step.
- **Test Analytics**
  - Used in the process of analysis of data.
  - Kibana, Elastic Search and Logstash are the open sources that are used in this step.

2. WHAT WE DO

- We help the user to perform an automated testing process that will simplify the process of testing.
- We give the user the benefits of Test Automation, Mobile Testing and DevOps.
- We help to perform the Android and iOS mobile application testing.
- We perform the testing within the allotted time given by the customer.
- We help to attain success by providing expert advice, using open source testing tools augmented by highly skilled software testing resources.

3. PREREQUISITES

The following Open Sources have to be installed in order to run the application. For installation process, please refer Installation Manual.

- **Notepad++**

  Notepad++ is a text editor and source code editor for use with Microsoft Windows.
- **RIDE**

  It is used to manage the execution of test cases. The user can select one or multiple test cases, provide various parameters for automation execution.
- **Kibana**

  It is used to analyze and display the data in the form of vertical graphs and pie charts.
- **Jenkins**

  Jenkins is a popular open source tool to perform continuous integration and build automation.
- **Appium**

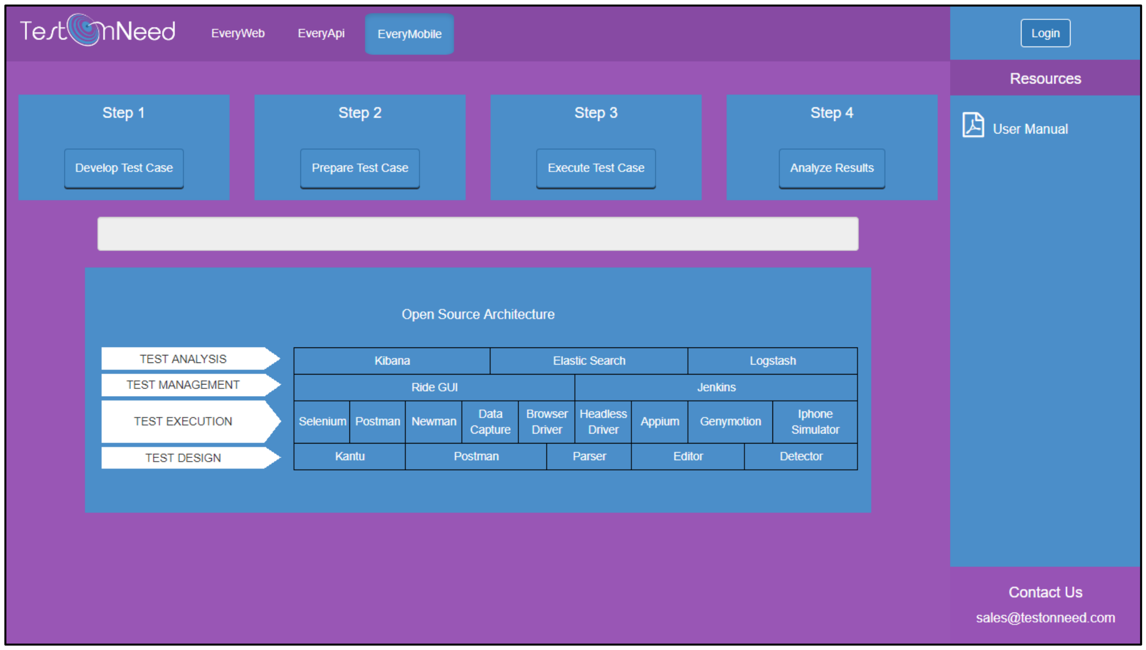
  Appium is an automation tool for running scripts and testing native applications and mobile-web applications on android or iOS using a web driver.
- **Android Studio**

  Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains IntelliJ IDEA software and designed specifically for Android development.
- **Emulator**

  An emulator enables the host system to run software or use peripheral devices designed for the guest system.

4. DESCRIPTION OF THE WEBSITE

- In order to start the project, there are certain steps to be followed by the user. To know how to start the project, please refer **Quick Start Guide**.
- The outline of the Website looks like below:

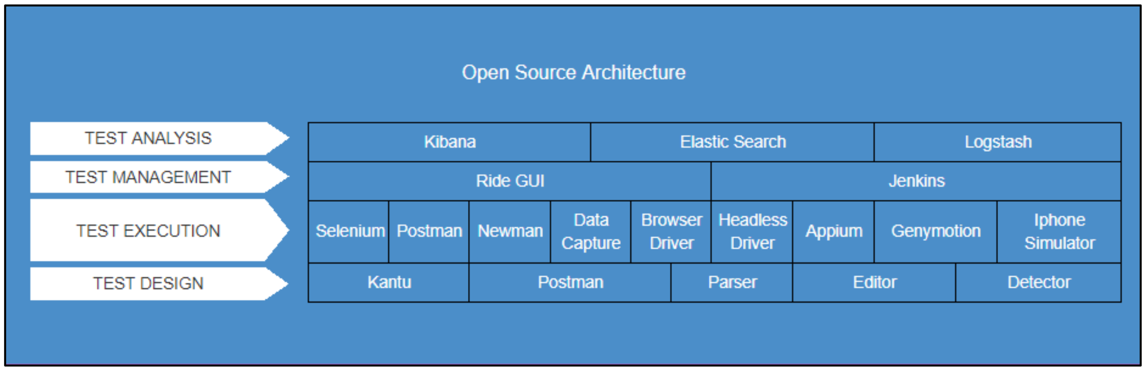


TEST ANALYSIS	Kibana, Elastic Search, Logstash
TEST MANAGEMENT	Ride GUI, Jenkins
TEST EXECUTION	Selenium, Postman, Newman, Data Capture, Browser Driver, Headless Driver, Appium, Genymotion, Iphone Simulator
TEST DESIGN	Kantu, Postman, Parser, Editor, Detector

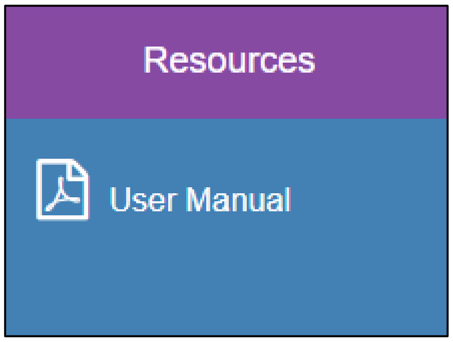
- The application has three tabs namely – EveryWeb, EveryAPI and EveryMobile.

- Things that are common to all the tabs are the stack which contains all the open sources, a status bar and the user manual.
- In the status bar, the user will get know what is being done next by clicking any button.

- The stack of open sources will contain all the applications being 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.



- The user manual guides the user through the web application. It will also change from tab to tab since working of each tab is different. It is a hyperlink, on clicking it, will open the respective document in pdf format in a new tab in the browser.



5. EVERYMOBILE PAGE

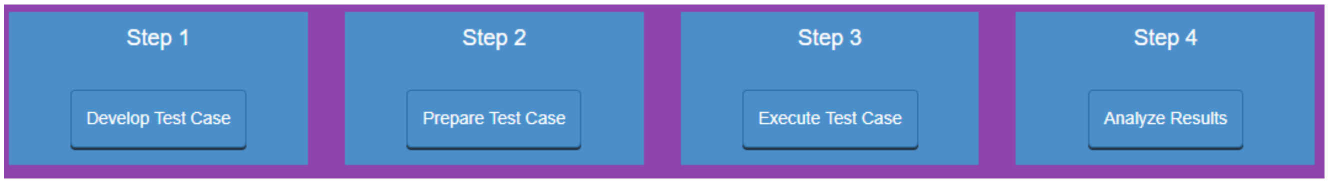
EveryMobile is a web application testing ecosystem for mobile using open sources. It is used to develop and execute the test cases for the applications using both Android and iOS.

The **Prerequisites of EveryMobile** are the following:

» The user should know Appium scripting.

» The basic of Android (Appium Automation Scripting) and iOS (XC Test Scripting).

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



- **Develop Test Case**

  » The user can write a test case in the editor and execute it in the upcoming steps.
- **Prepare Test Case**

  » In the background, the downloaded JSON file gets converted to its corresponding Python file.
- **Execute Test Case**

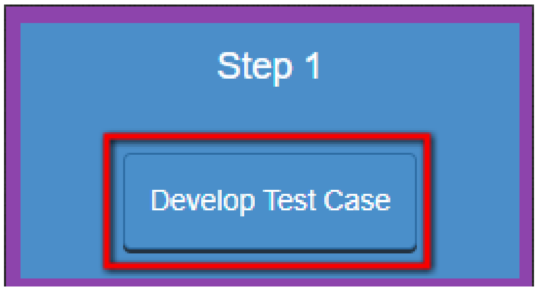
  » The user can execute the automated test cases that was recorded in the first step.
- **Analyze Results**

  » The user can analyze the results visually with the help of graphs.
- Each step is explained below

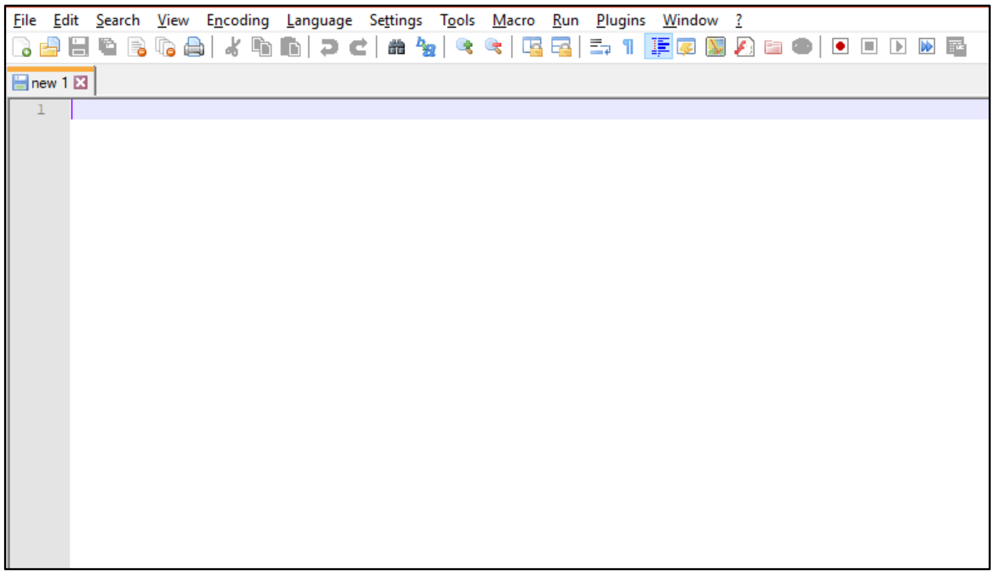
5.1 DEVELOP TEST CASE

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 reference.

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

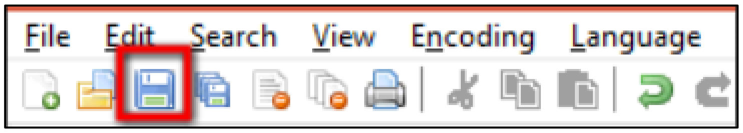


2. On clicking the **Develop Test Case** button, Notepad++ opens up.


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

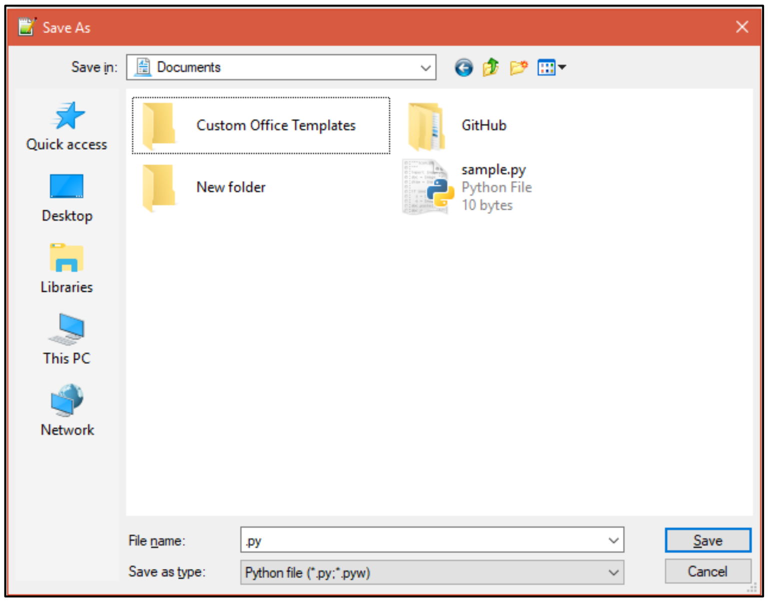
NOTE: The Notepad ++ should be saved in the C: drive then only it will open. For the installation process, please refer the installation manual.

3. Write the Appium code for executing the test case. For writing the code, please <u>refer</u> in the Reference section.
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.

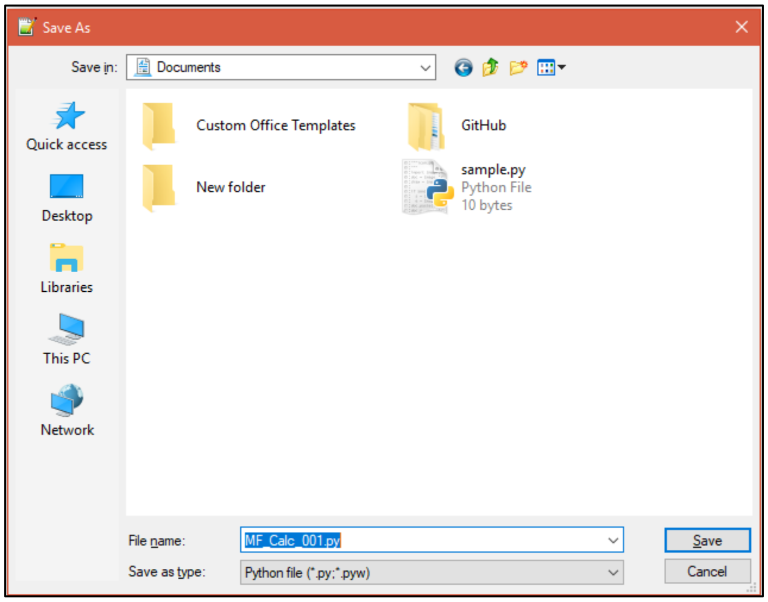
6. In the **file name** text, give the name of the file and in **save as type** drop down, select **Python file**.



File name: .py

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

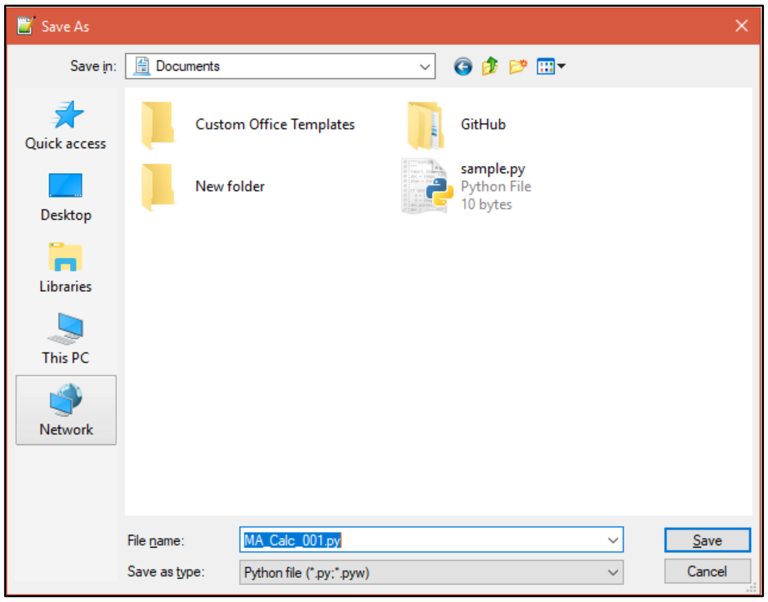
7. Click Save to save the file.
8. Now save the file based on the type of testing functionality. There are 2 types of testing functionality– **Function and Automation.**
9. In the case of **Function**, the test case should be saved as **MF_<TestCase_Name>**. For example, the test case should be saved 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**.



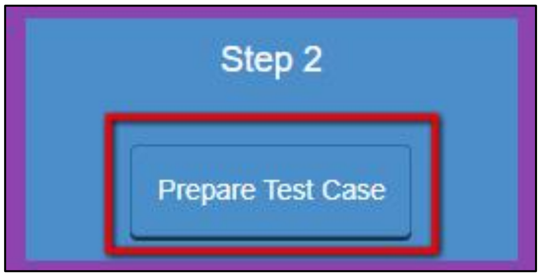
File name: MA_Calc_001.py

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

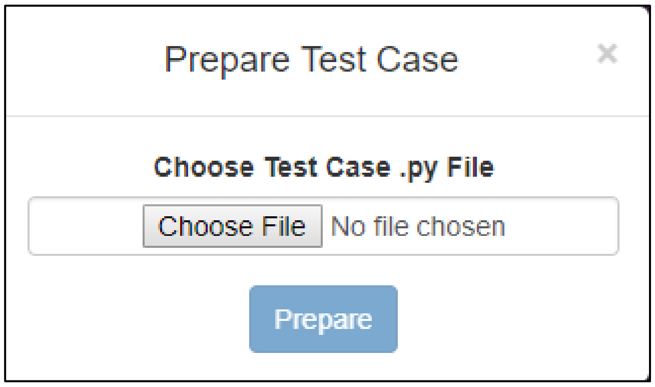
5.2 PREPARE TEST CASE

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

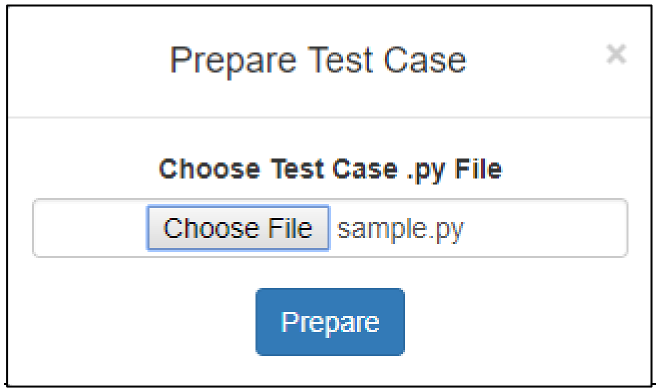
1. Click the **Prepare Test Case** button.



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

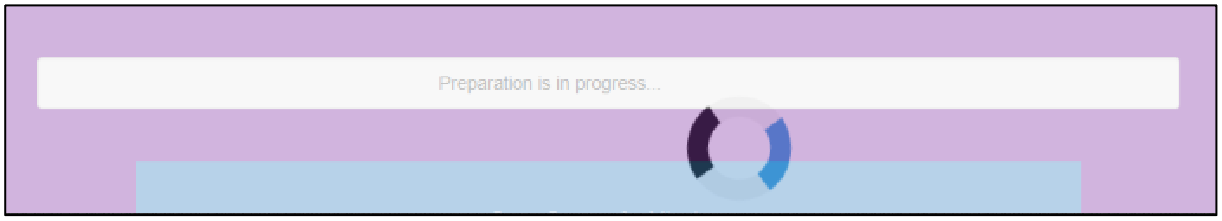


3. Click on the **Choose File** button.
4. Surfing the folder location where saved the Python file that was saved in the previous step.
5. Now upload the Python file.
6. The user can upload **only one file** at a time.
7. On uploading the Python file, the **Prepare** button becomes active.



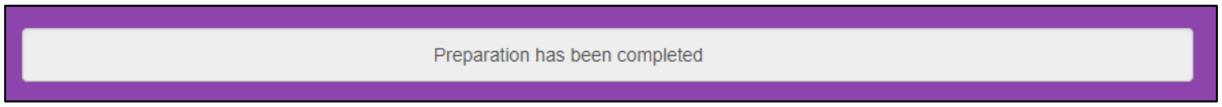
8. On uploading JSON file and clicking the **Prepare** button, the following will happen at the backend.
   - It creates a folder in the name of the test case that was mentioned in the previous step.
   - The folder gets stored in:

     > 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.

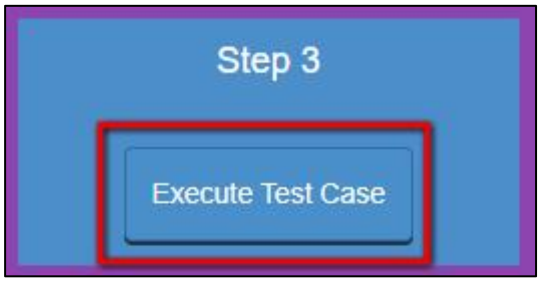


Preparation has been completed

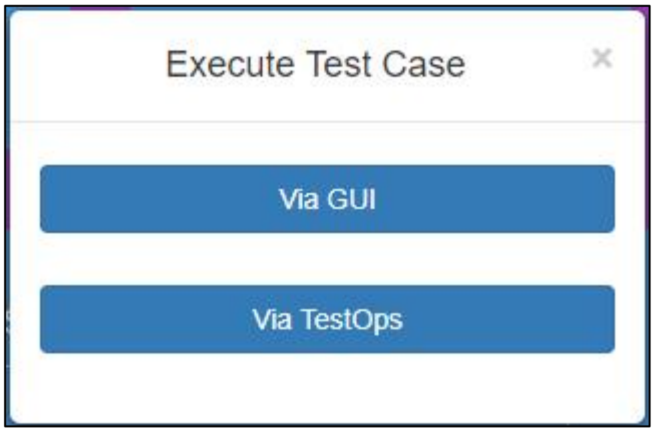
5.3 EXECUTE TEST CASE

The Execute Test Case button is used for the execution process. The execution process takes place using RIDE and TestOps. For the execution of the test cases of mobile, the user need to install Genymotion, Appium, Android Studio in the local system.

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

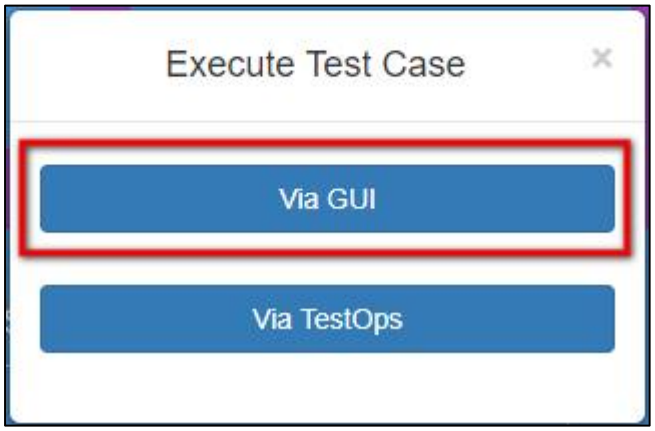


2. On clicking on the button, a pop up appears.
3. From the pop-up, select the button either as **Via GUI, Via TESTOPS.**

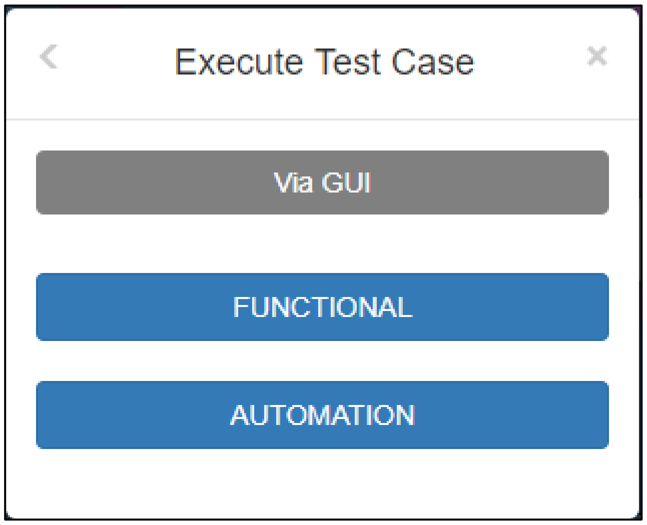


5.3.1 EXECUTE – VIA GUI

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

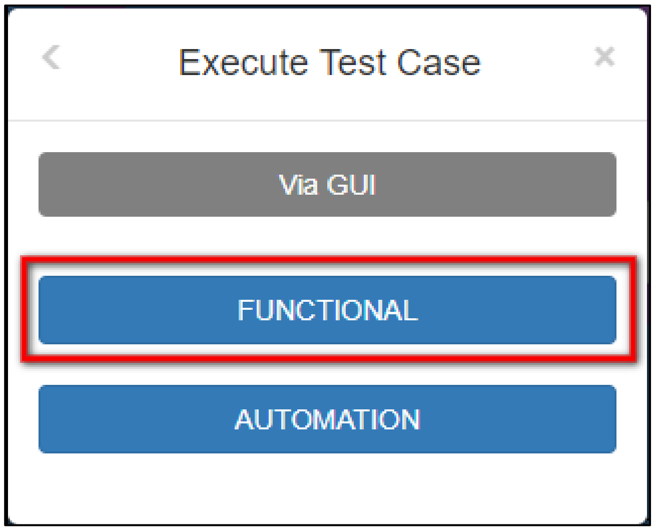


2. On clicking the button, 2 buttons appear – **Function and Automation**.

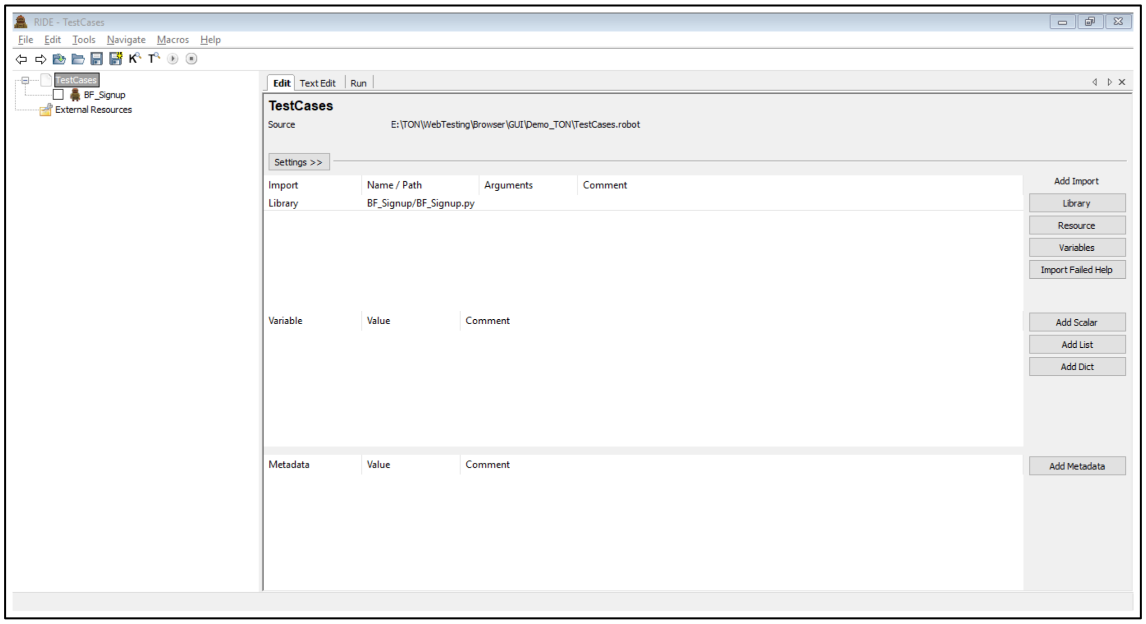


5.3.1.1 VIA GUI – FUNCTIONAL

1. To perform functional testing, press Functional Button.



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



TestCases

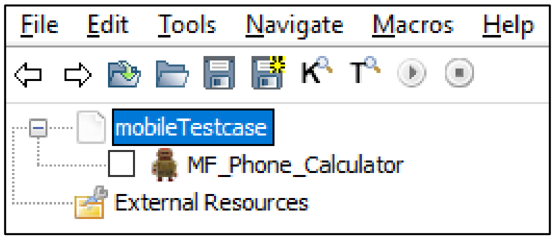
Source E:\TON\WebTesting\Browser\GUI\Demo_TON\TestCases.robot

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

Variable	Value	Comment

Metadata	Value	Comment

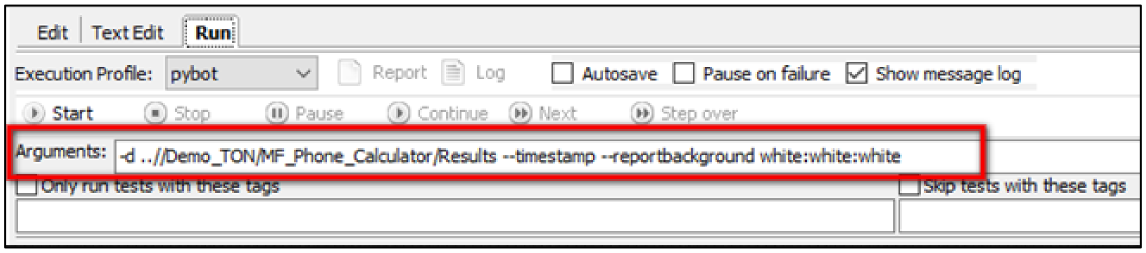
3. On the left panel, select one test case to execute.



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

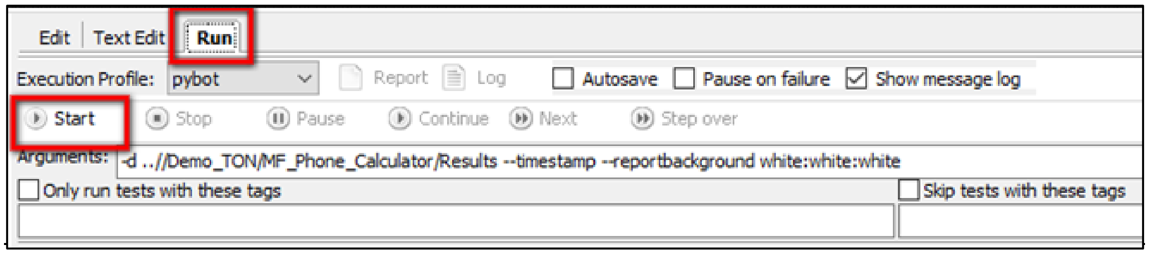
5. On click of the test case name, the **Edit** tab opens.
6. On the top, click **Run** Tab.
7. At the top of the Run tab, configure the **Arguments** textbox by entering:
8. Configure the **Arguments** textbox by entering:

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



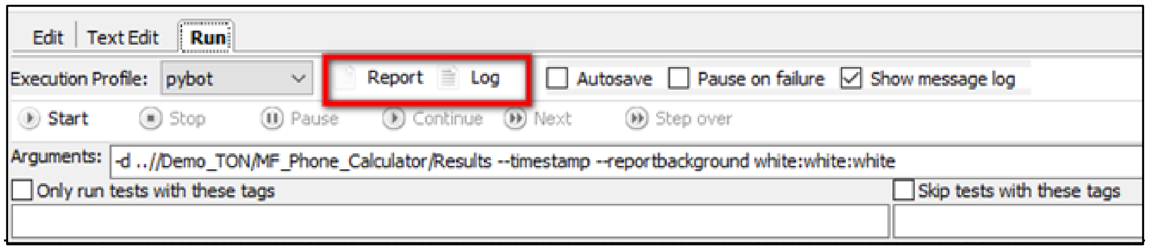
Execution Profile: pybot
Autosave ☐ Pause on failure ☐ Show message log ☑
Arguments: -d ..//Demo_TON/MF_Phone_Calculator/Results --timestamp --reportbackground white:white:white
☐ Only run tests with these tags
☐ Skip tests with these tags

9. Before running the code, the user has to start Genymotion or setup the real phone. To setup the real phone device, click Here. To start the Genymotion, click Here.
10. Under Run tab, click **Start** button to start the execution.



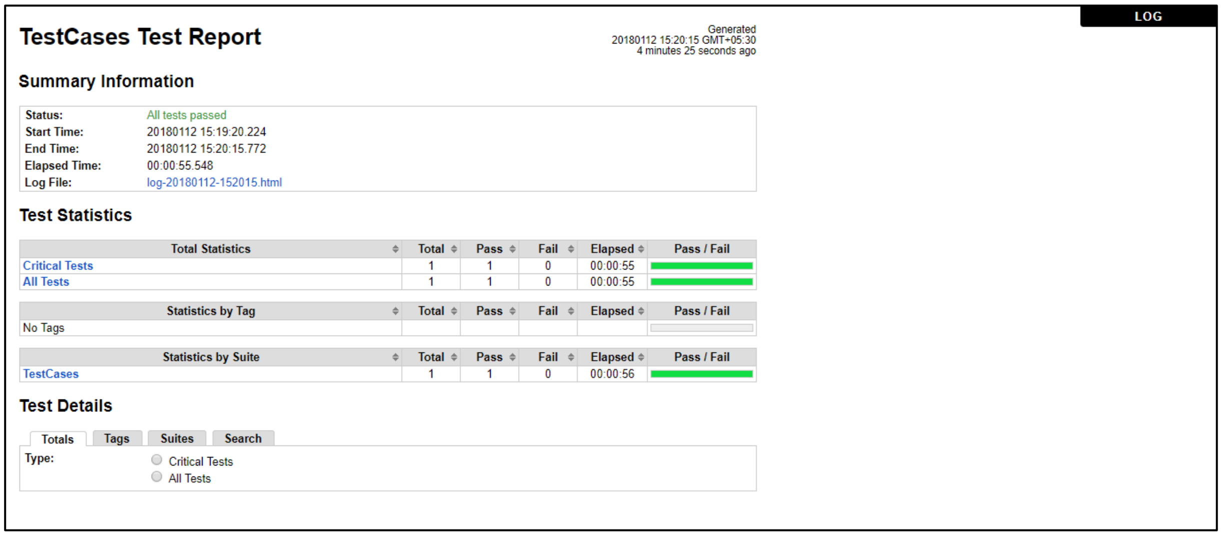
Execution Profile: pybot
Autosave ☐ Pause on failure ☐ Show message log ☑
Arguments: -d ..//Demo_TON/MF_Phone_Calculator/Results --timestamp --reportbackground white:white:white
☐ Only run tests with these tags
☐ Skip tests with these tags

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/MF_Phone_Calculator/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.



LOG

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					

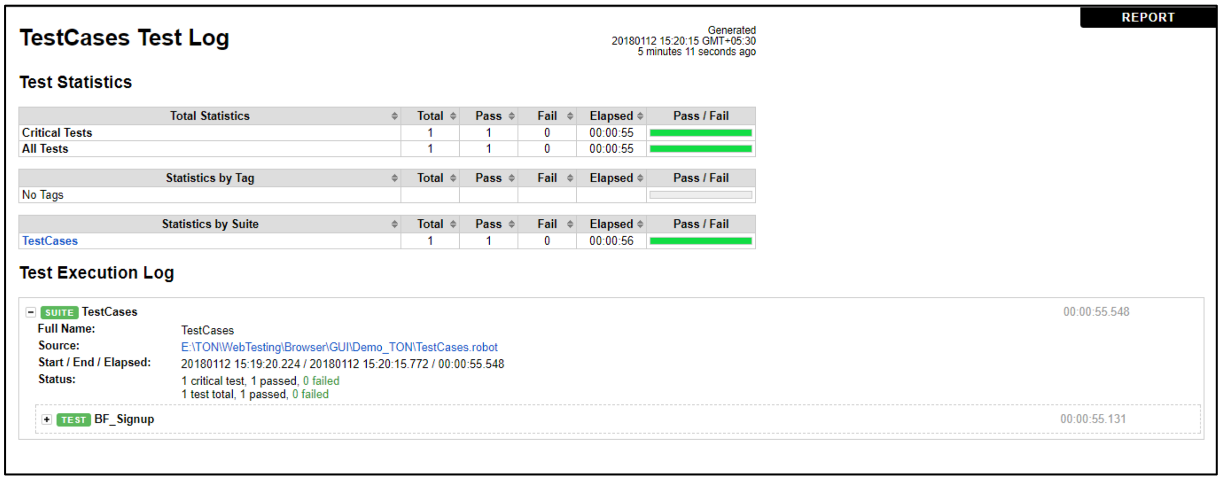
Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
TestCases	1	1	0	00:00:56	🟩

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.



REPORT

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

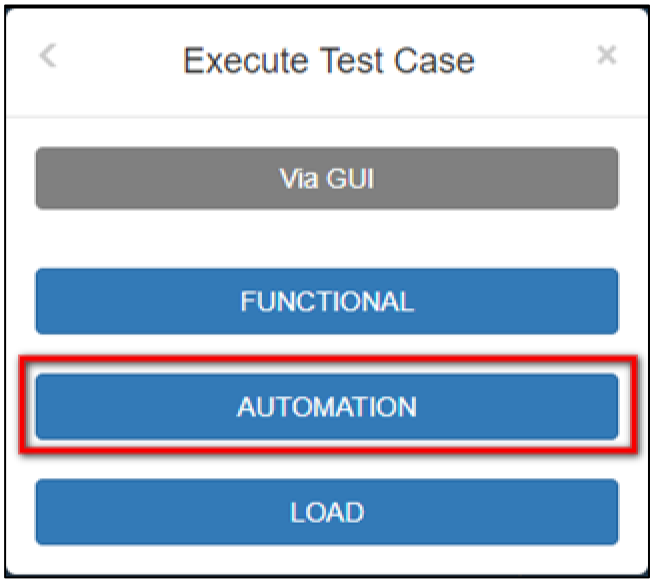
SUITE TestCases 00:00:55.548

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

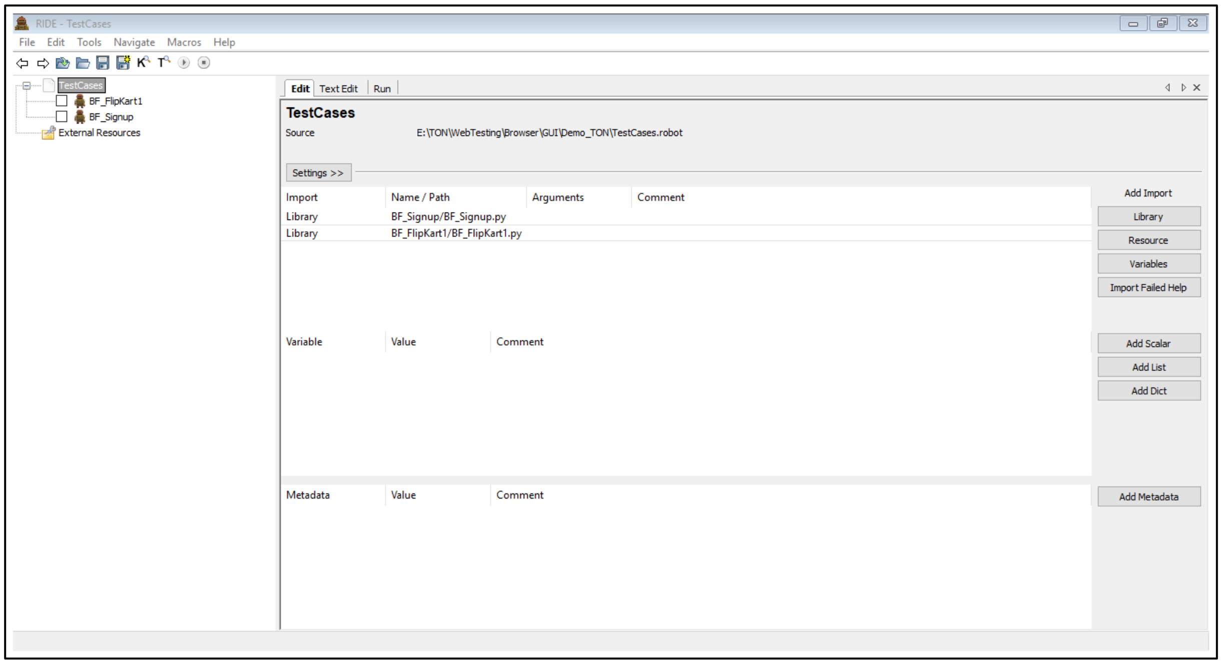
TEST BF_Signup 00:00:55.131

5.3.1.2 VIA GUI – AUTOMATION

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



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



TestCases

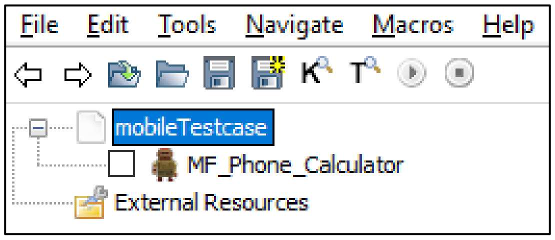
Source E:\TON\WebTesting\Browser\GUI\Demo_TON\TestCases.robot

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

Variable	Value	Comment

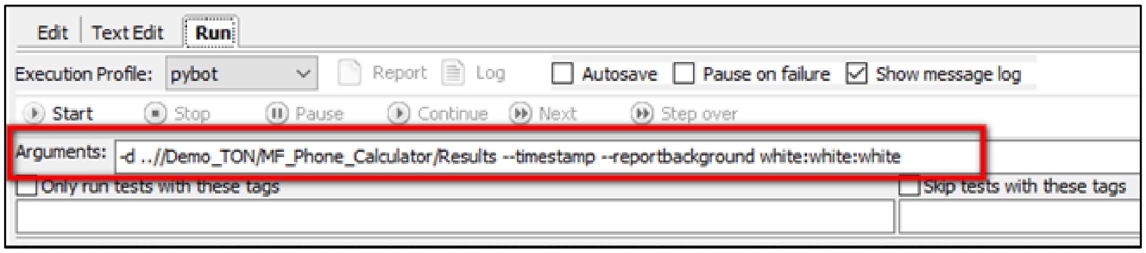
Metadata	Value	Comment

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



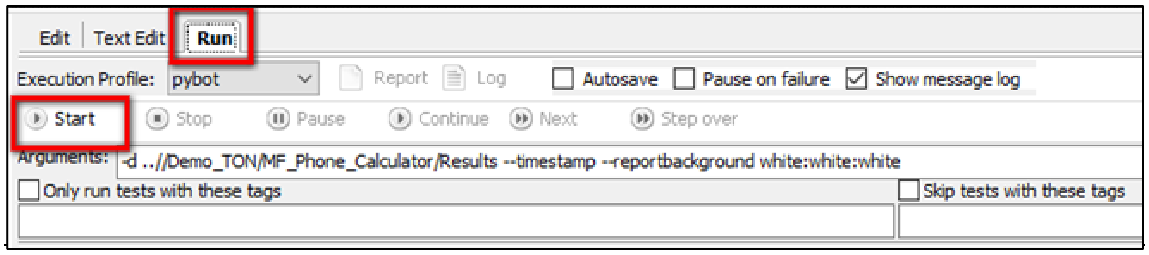
4. To configure the parameters, click on the test case name.
5. On click of the test case name, the **Edit** tab opens.
6. On the top, click **Run** Tab.
7. At the top of the Run tab, configure the **Arguments** textbox by entering:
8. Configure the **Arguments** textbox by entering:

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



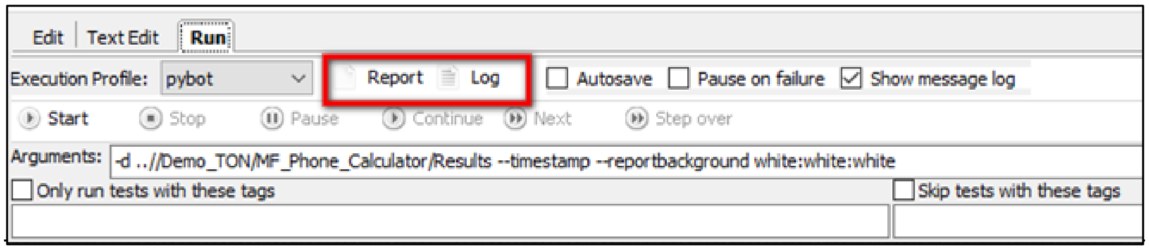
Execution Profile: pybot
Autosave [ ] Pause on failure [ ] Show message log [✓]
Arguments: -d ../Demo_TON/MF_Phone_Calculator/Results --timestamp --reportbackground white:white:white
Only run tests with these tags [ ]
Skip tests with these tags [ ]

9. Before running the code, the user has to start Genymotion or setup the real phone. To setup the real phone device, click Here. To start the Genymotion, click Here.
10. Under Run tab, click **Start** button to start the execution.



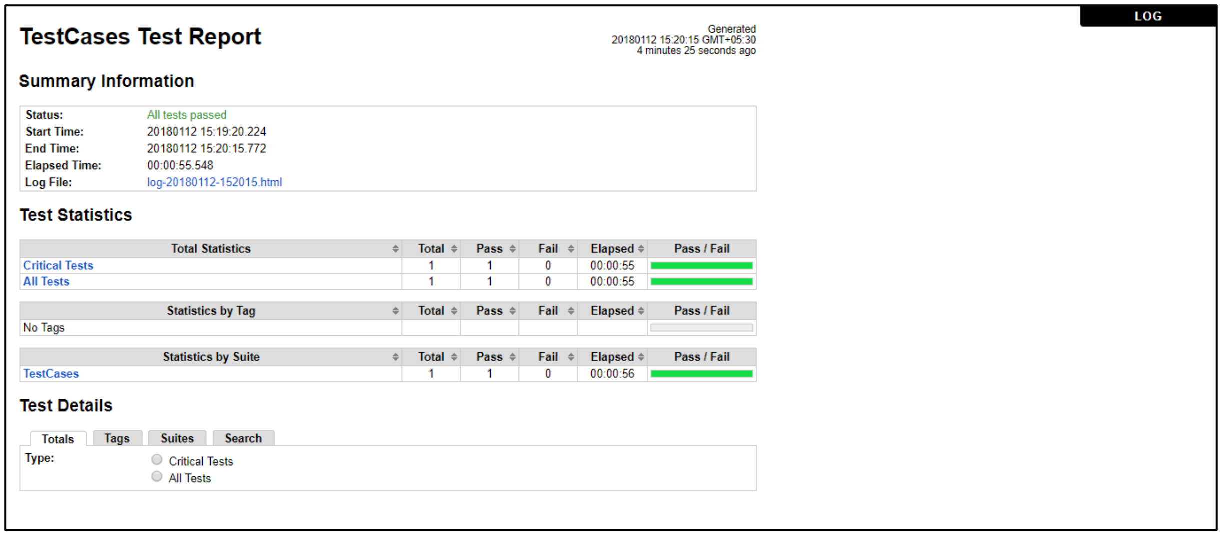
Execution Profile: pybot
Autosave [ ] Pause on failure [ ] Show message log [✓]
Arguments: -d ../Demo_TON/MF_Phone_Calculator/Results --timestamp --reportbackground white:white:white
Only run tests with these tags [ ]
Skip tests with these tags [ ]

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



Execution Profile: pybot
Autosave [ ] Pause on failure [ ] Show message log [✓]
Arguments: -d ../Demo_TON/MF_Phone_Calculator/Results --timestamp --reportbackground white:white:white
Only run tests with these tags [ ]
Skip tests with these tags [ ]

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



LOG

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					

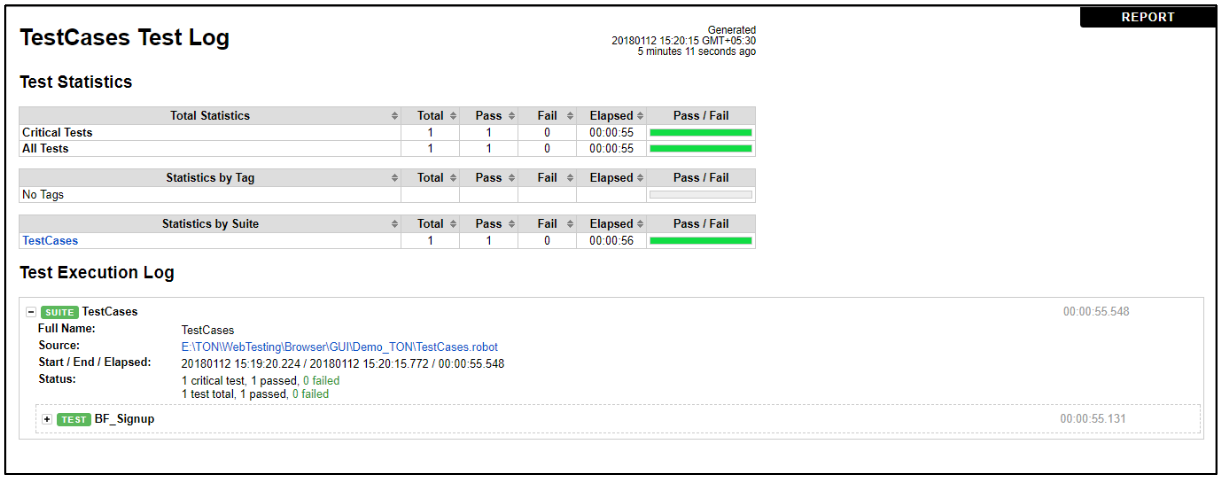
Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
TestCases	1	1	0	00:00:56	🟩

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.



REPORT

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					

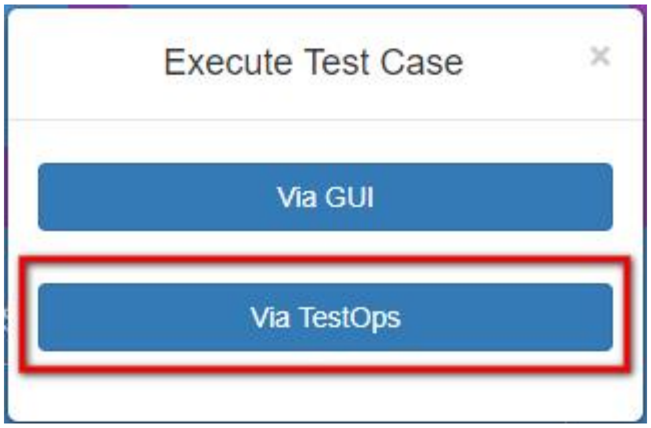
Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
TestCases	1	1	0	00:00:56	🟩

Test Execution Log

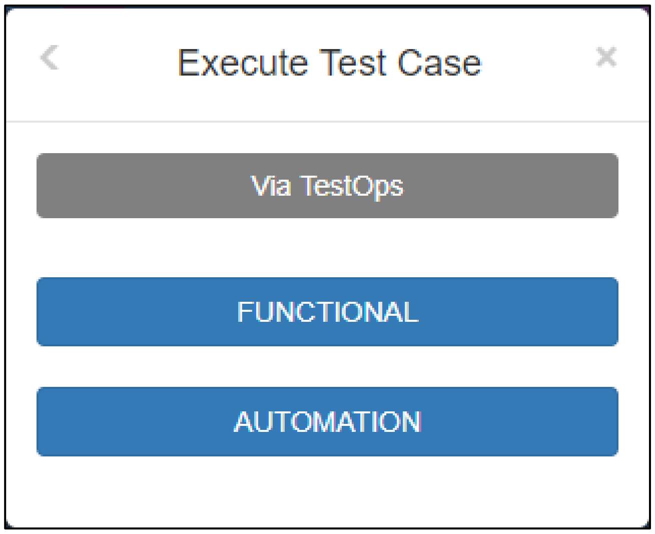
```
− SUITE TestCases                                                      00:00:55.548
    Full Name:             TestCases
    Source:                E:\TON\WebTesting\Browser\GUI\Demo_TON\TestCases.robot
    Start / End / Elapsed: 20180112 15:19:20.224 / 20180112 15:20:15.772 / 00:00:55.548
    Status: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

5.3.2 EXECUTE - VIA TESTOPS

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

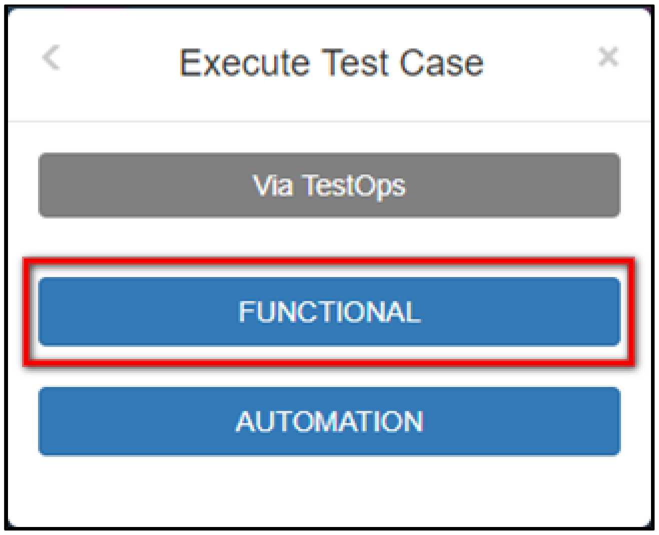


2. On clicking the button, 3 buttons will appear – **Functional and Automation**.

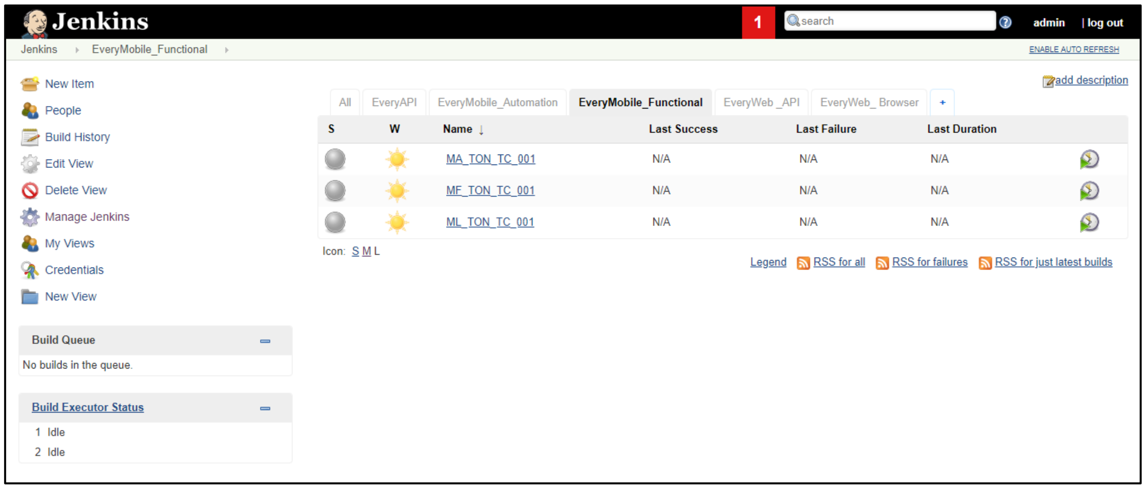


5.3.2.1 VIA TESTOPS – FUNCTIONAL

1. Click on the **Functional** button.



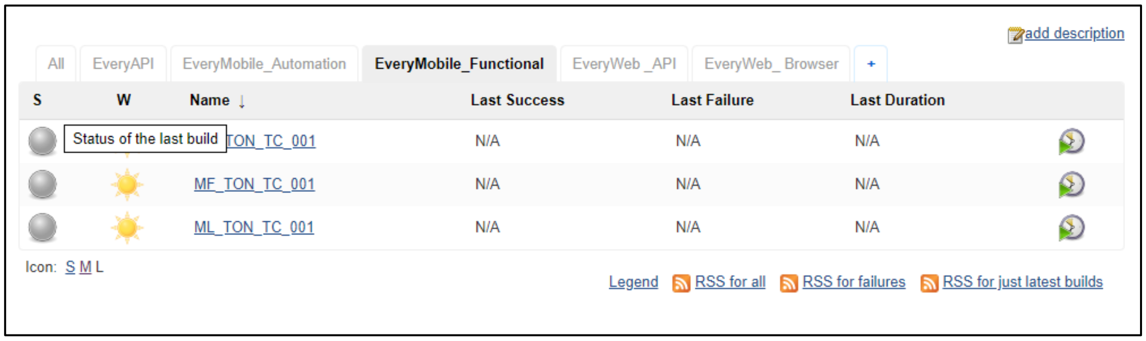
2. On click of the button, the **Jenkins** will open.



S	W	Name ↓	Last Success	Last Failure	Last Duration
		MA_TON_TC_001	N/A	N/A	N/A
		MF_TON_TC_001	N/A	N/A	N/A
		ML_TON_TC_001	N/A	N/A	N/A

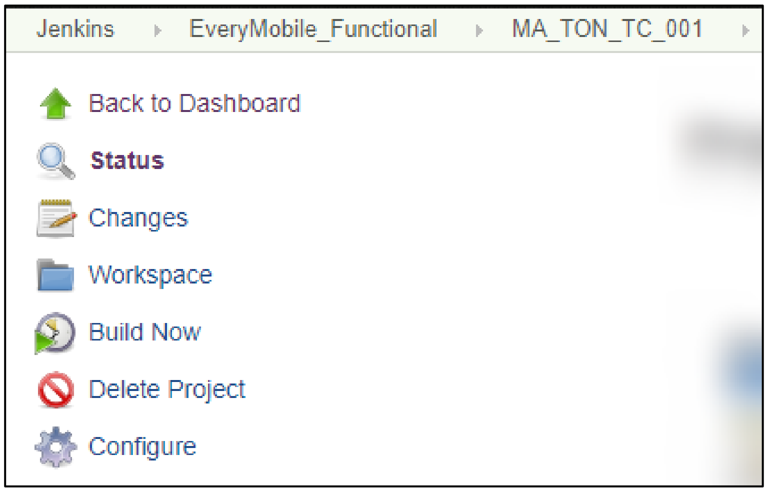
For creation of new <u>View</u> and <u>Job</u>, refer the Reference Section below.

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



S	W	Name ↓	Last Success	Last Failure	Last Duration
		TON_TC_001	N/A	N/A	N/A
		MF_TON_TC_001	N/A	N/A	N/A
		ML_TON_TC_001	N/A	N/A	N/A

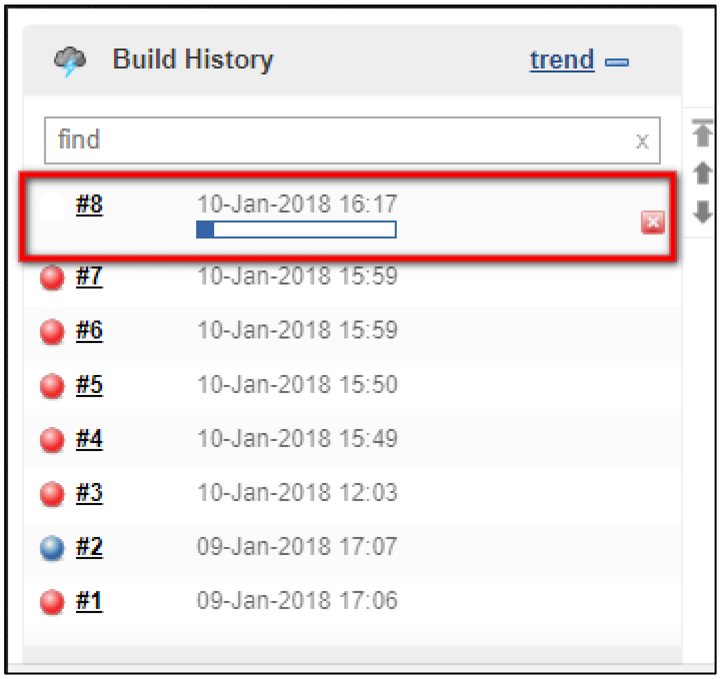
4. On the left side, select Build Now.



Jenkins ▸ EveryMobile_Functional ▸ MA_TON_TC_001 ▸

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

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

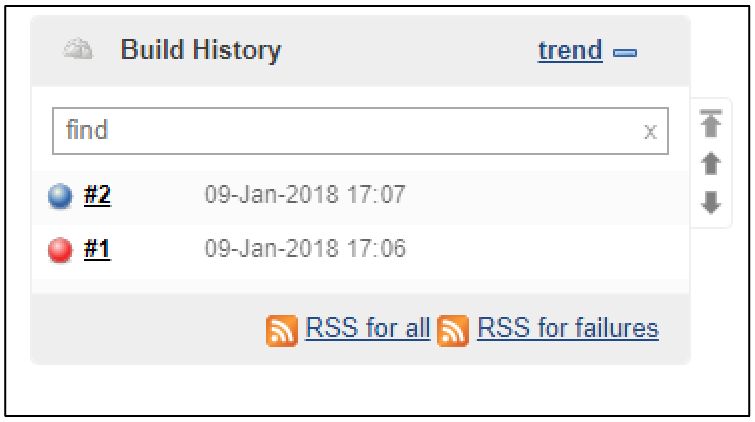


Build History trend

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

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



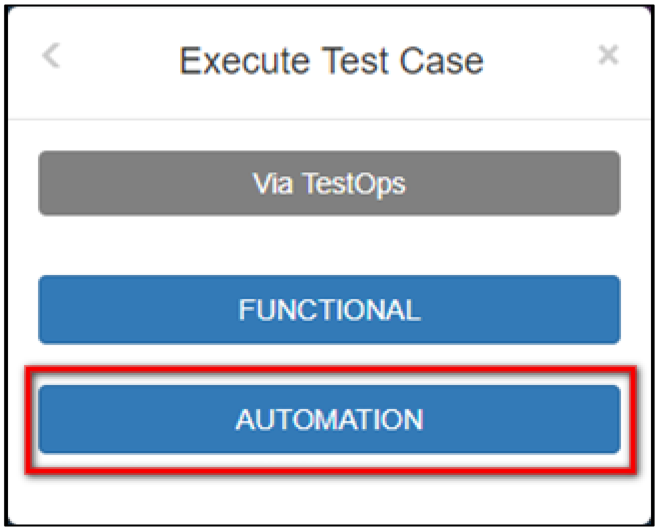
#2	09-Jan-2018 17:07
#1	09-Jan-2018 17:06

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

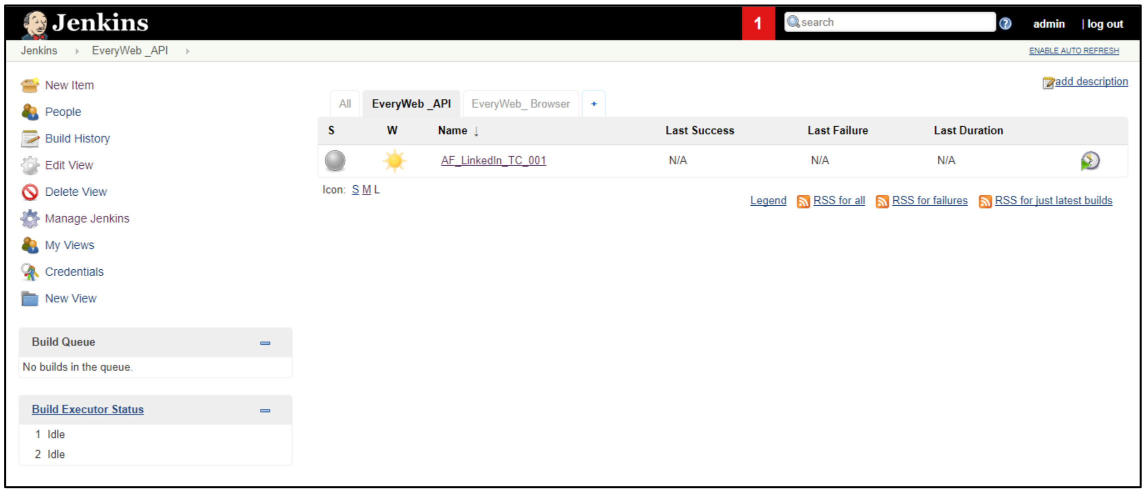
   > TON > MobileTesting > GUI > TestOps > Demo_TON > <TestCase_Name_Folder> > <Job_Name>

5.3.2.2 VIA TESTOPS – AUTOMATION

1. Click on the **Automation** button.



2. On click of the button, the **Jenkins** will open.



S	W	Name ↓	Last Success	Last Failure	Last Duration
●	☀	AF_LinkedIn_TC_001	N/A	N/A	N/A

Build Queue

No builds in the queue.

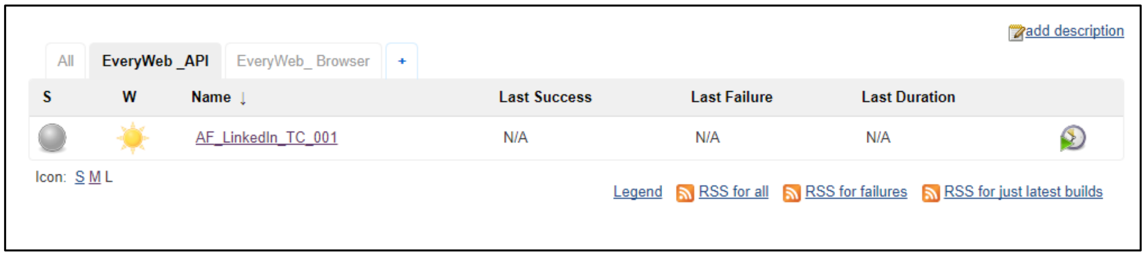
Build Executor Status

1 Idle

2 Idle

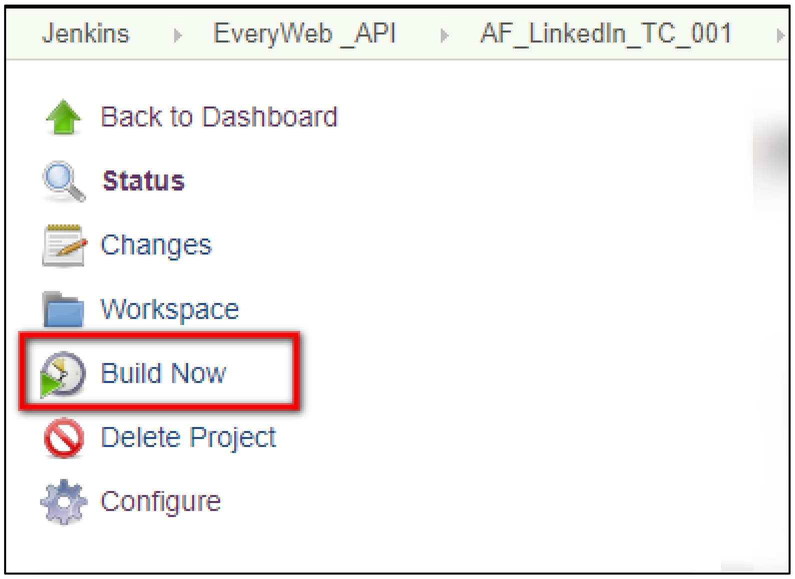
For creation of new View and Job, refer the Reference Section below.

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

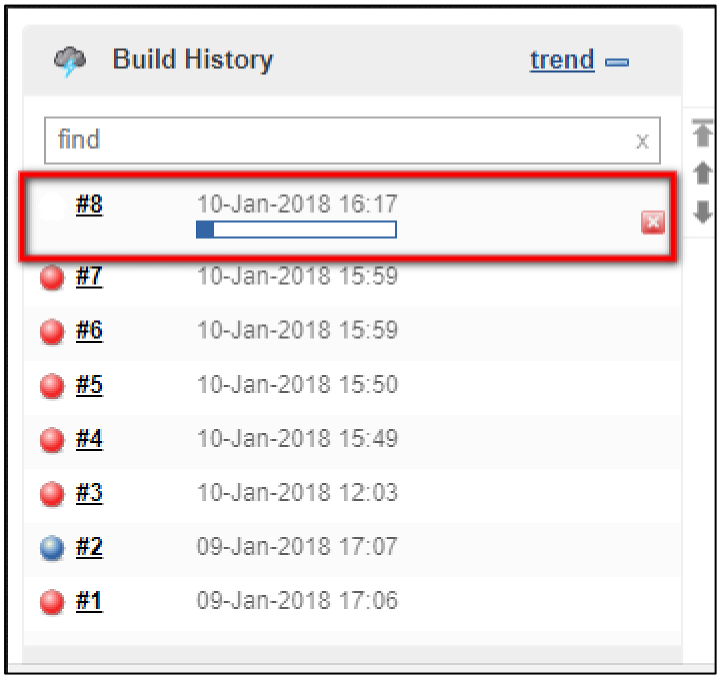


S	W	Name ↓	Last Success	Last Failure	Last Duration
●	☀	AF_LinkedIn_TC_001	N/A	N/A	N/A

4. On the left side, select Build Now.

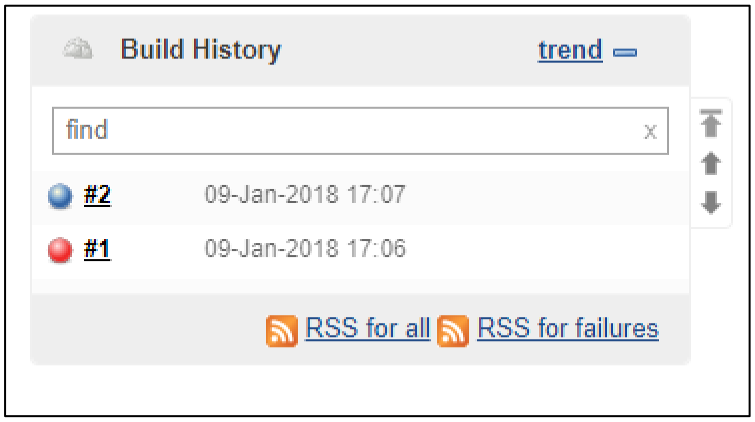


5. On the left side, under Build History, the progress of the job is shown.



#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

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



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

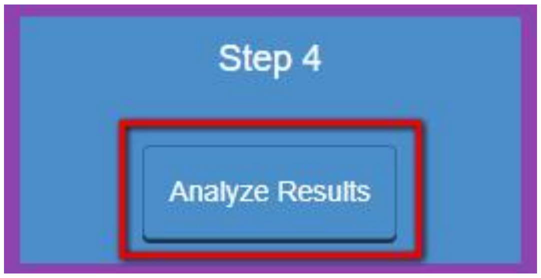
7. All the jobs of the Jenkins will be saved in:

   > TON > WebTesting > API > TestOps > Demo_TON > <TestCase_Name_Folder> > <Job_Name_Folder>

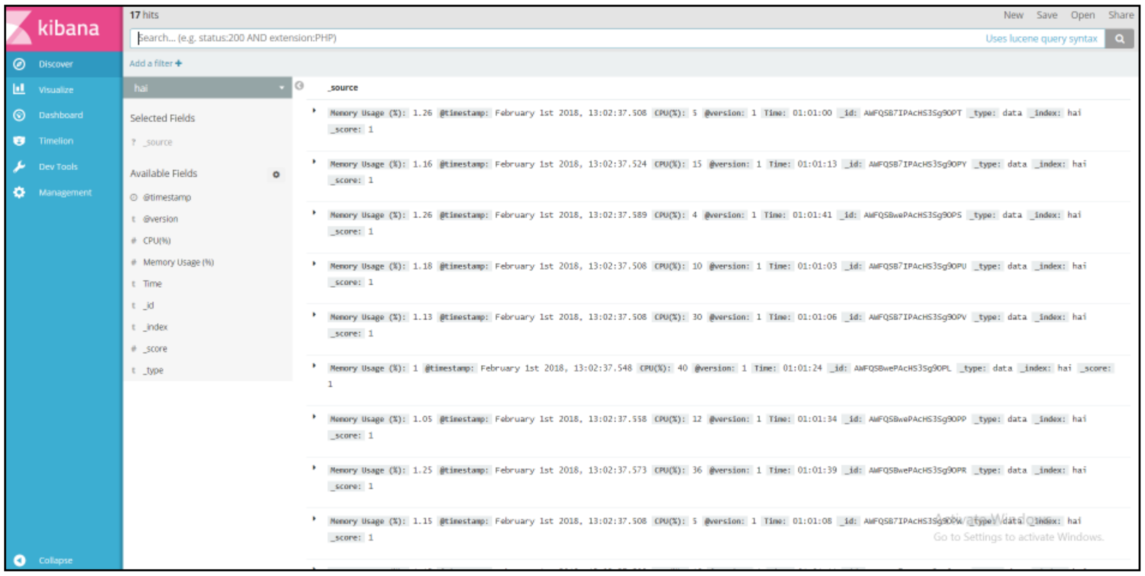
5.4 ANALYZE RESULTS

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**.



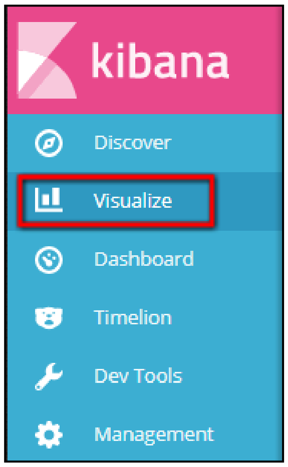
2. On clicking on the **Analyze Results** button, Kibana opens up in a new window.



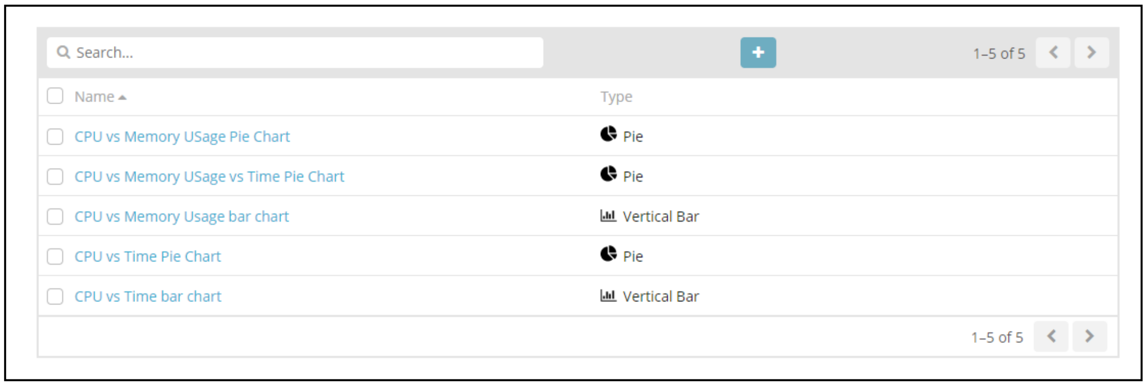
17 hits

```
_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
```

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.

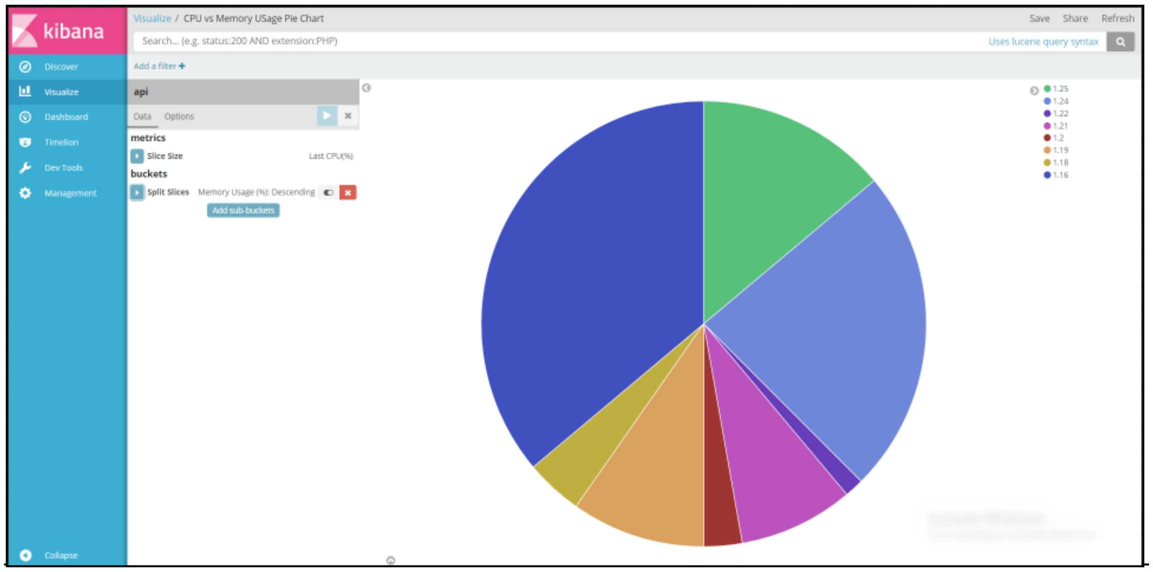


Search...   1–5 of 5

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

1–5 of 5

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)



Visualize / CPU vs Memory USage Pie Chart

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

api

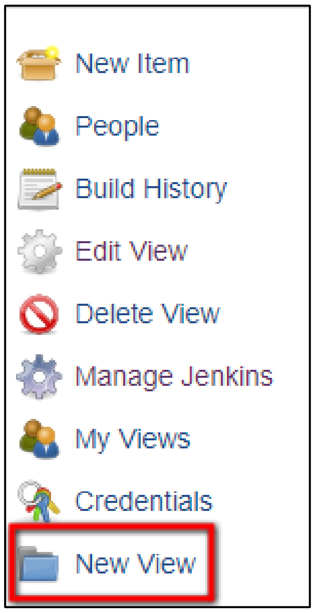
metrics
Slice Size   Last CPU(%)

buckets
Split Slices   Memory Usage (%): Descending

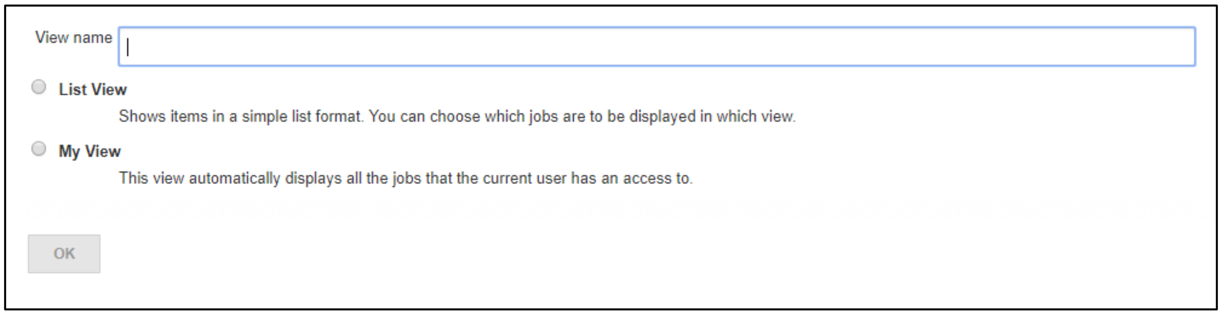
6. REFERENCES

6.1 CREATION OF NEW VIEW

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

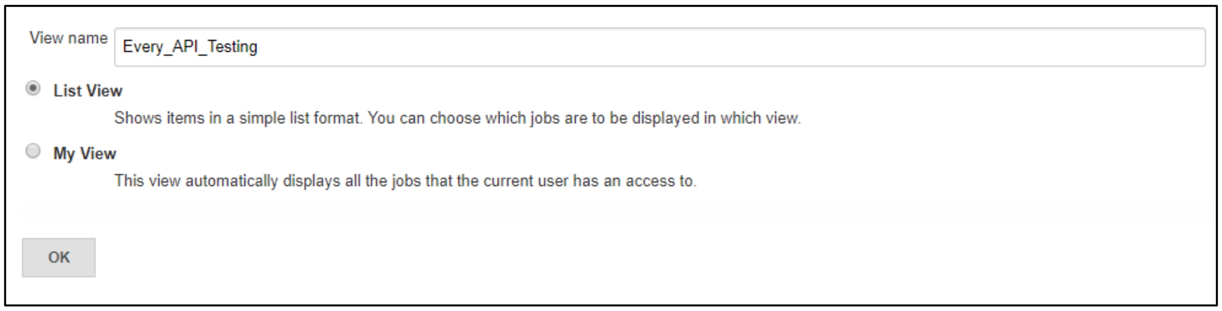


- On clicking, give the name of the tab as **Every_Mobile_Testing** and select the type of view.



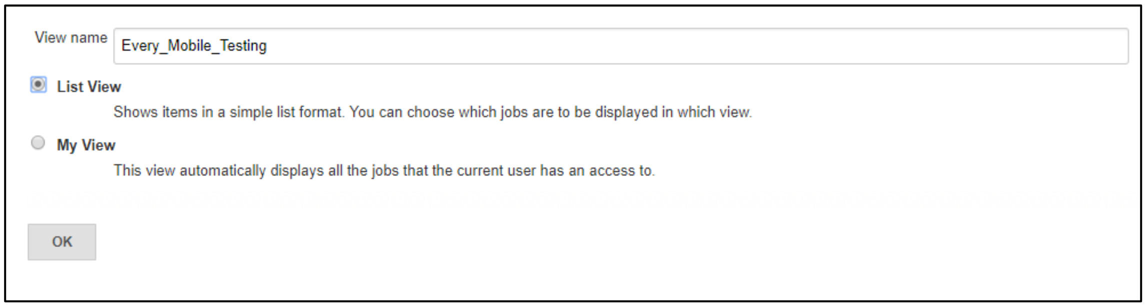
List View
Shows items in a simple list format. You can choose which jobs are to be displayed in which view.
My View
This view automatically displays all the jobs that the current user has an access to.

- 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.

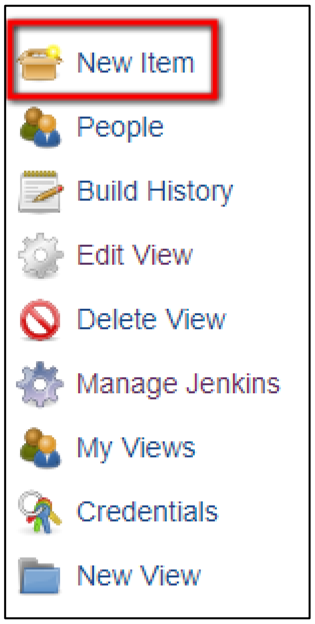
My View

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

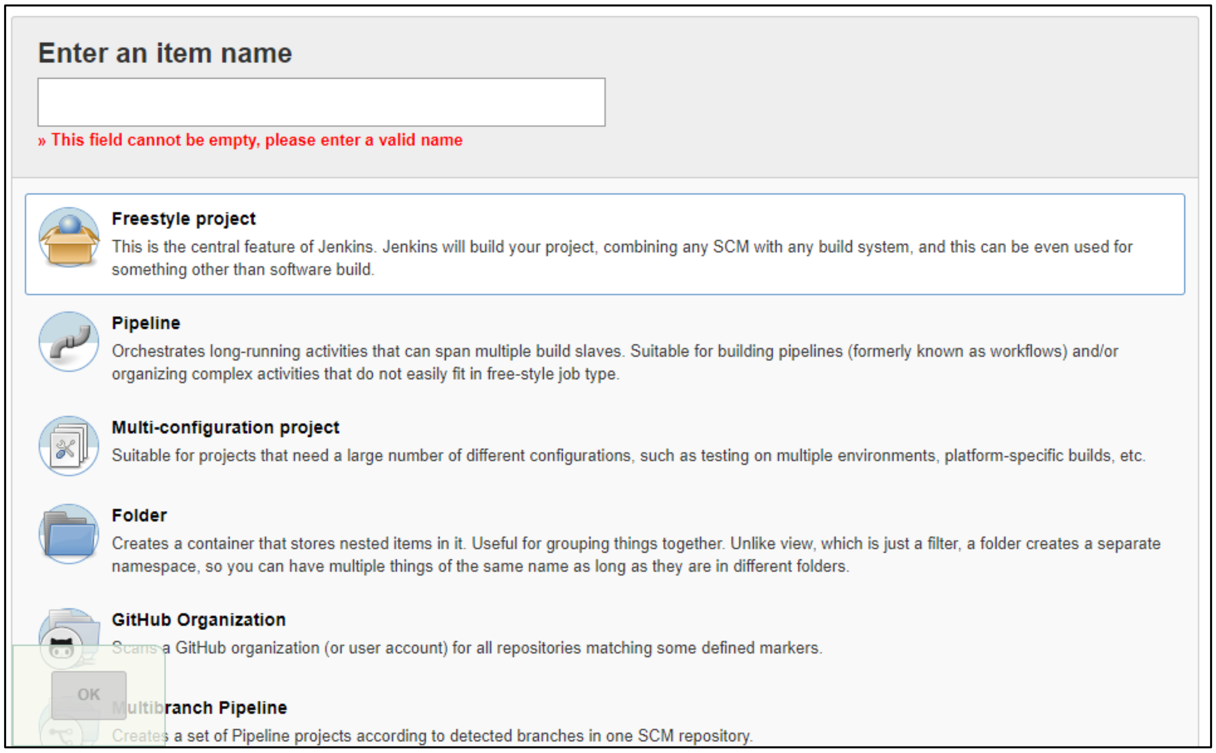
- Once done, click Apply and Ok to save and Close the creation.

6.2 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.

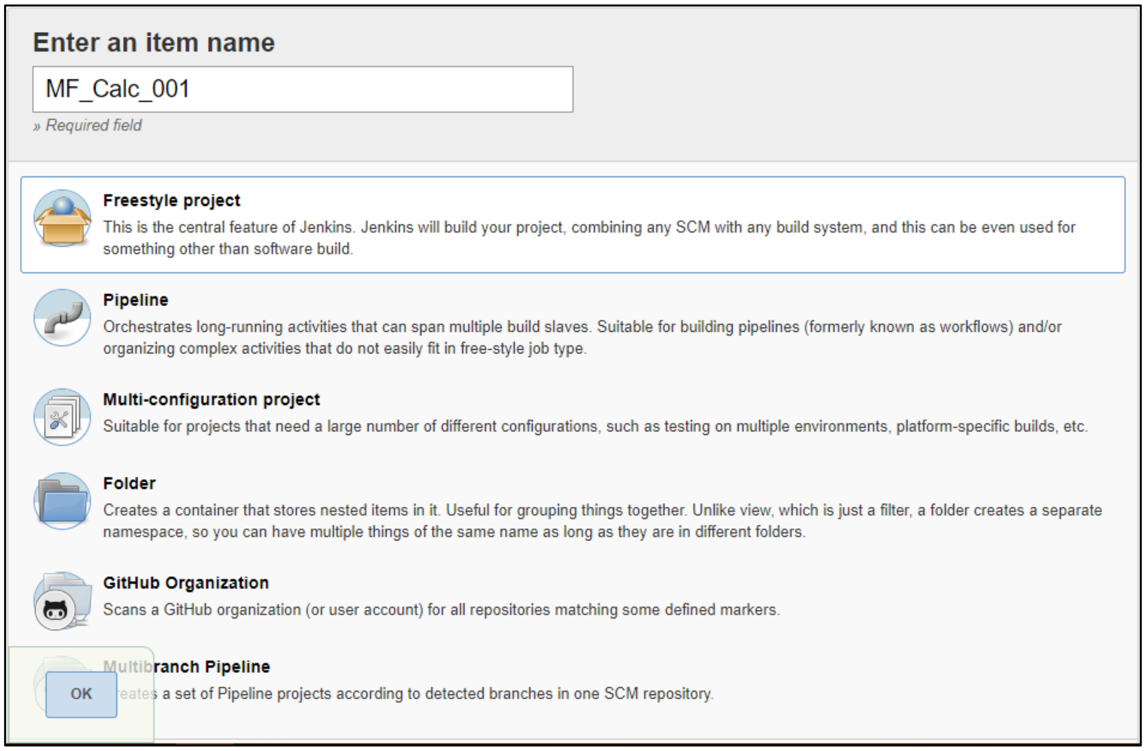
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

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.

OK

- 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.

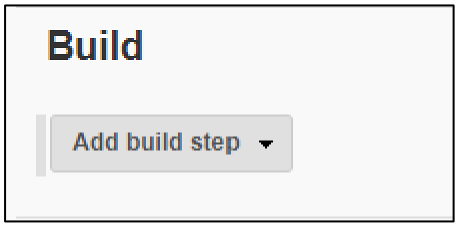
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

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

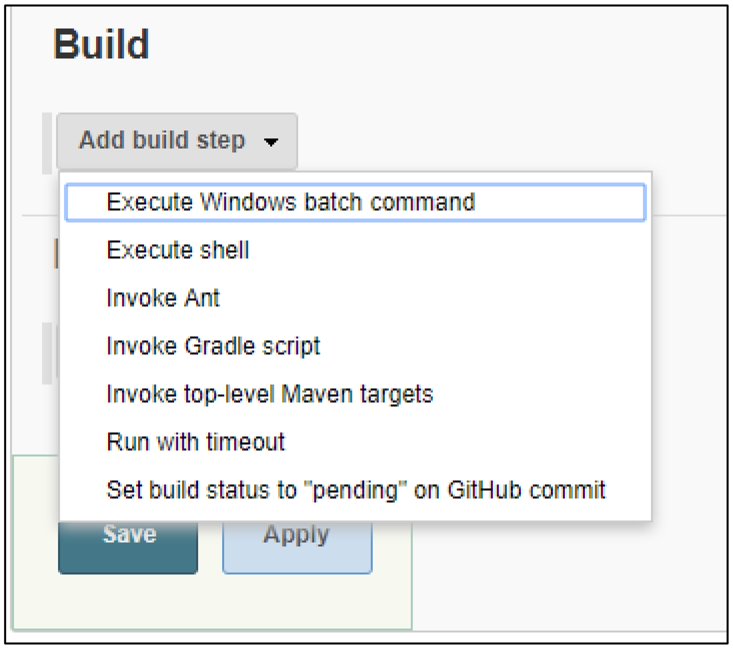
Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

OK

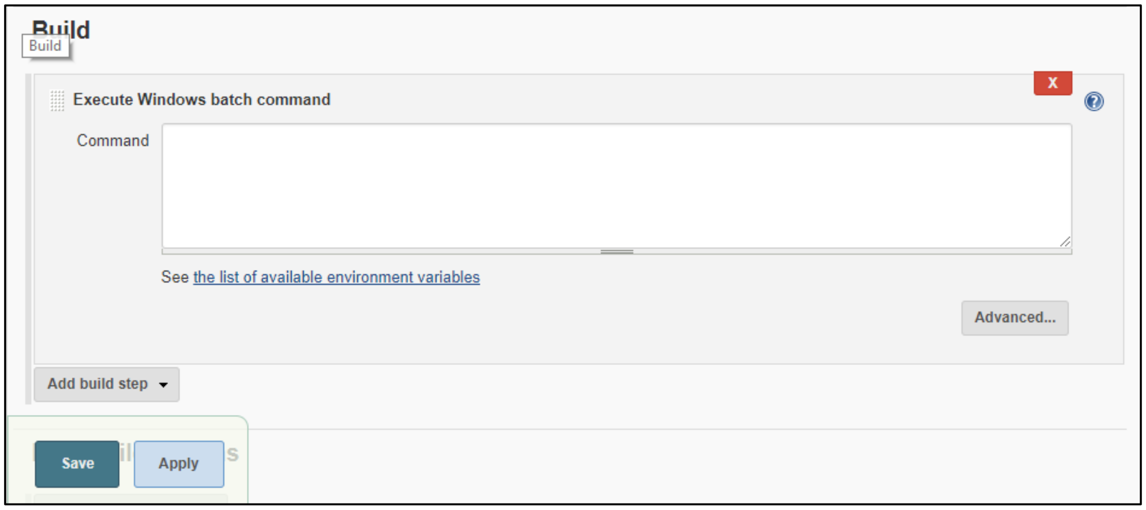
- On click of Ok button, job configuration page will appear.
- Scroll down and look for Build.



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

- In the text area, write the following command.

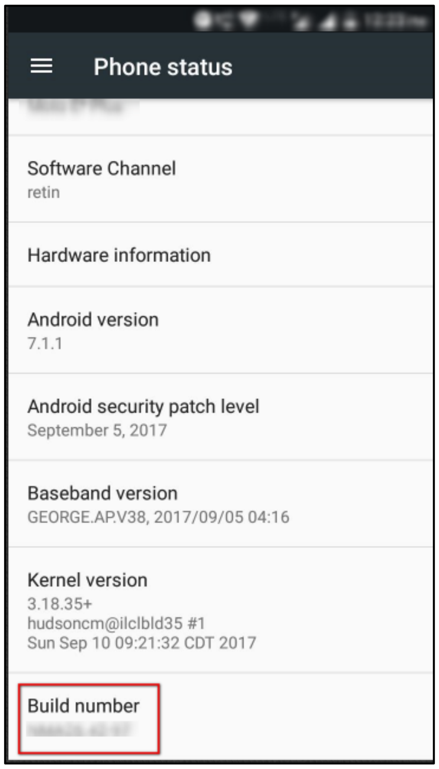
6.3 ANDROID PHONE AND GENYMOTION

There are 2 ways to execute the mobile test cases. One through the real device that is through the user's phone and other through the virtual device that is through genymotion.

6.3.1 THROUGH ANDROID PHONE

6.3.1.1 CONNECTING ANDROID PHONE

1. Firstly, the user should connect the phone through the USB cable.
2. In the settings of the phone, the user has to enable the developer option.
3. To enable the developer option, open Settings > About Phone > tap on the **Build Number** for 6-7 times, it will display "Now you are a developer".



Phone status

Software Channel
retin

Hardware information

Android version
7.1.1

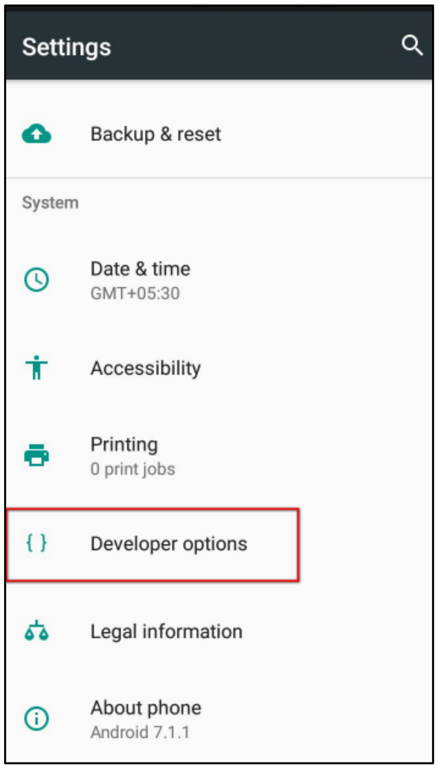
Android security patch level
September 5, 2017

Baseband version
GEORGE.AP.V38, 2017/09/05 04:16

Kernel version
3.18.35+
hudsoncm@ilclbld35 #1
Sun Sep 10 09:21:32 CDT 2017

Build number

4. Now go back to settings of phone, the user will find the **Developer Options** setting.



Settings

Backup & reset

System

Date & time
GMT+05:30

Accessibility

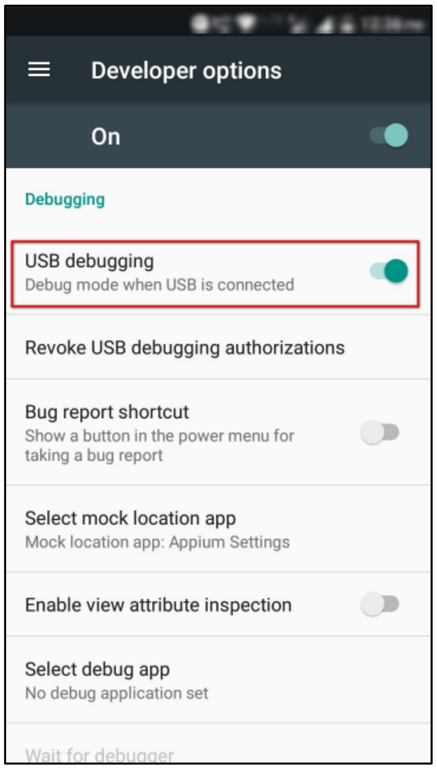
Printing
0 print jobs

Developer options

Legal information

About phone
Android 7.1.1

5. In the Developer Options, scroll down and switch on the **USB debugging**.



Developer options

On

Debugging

USB debugging
Debug mode when USB is connected

Revoke USB debugging authorizations

Bug report shortcut
Show a button in the power menu for taking a bug report

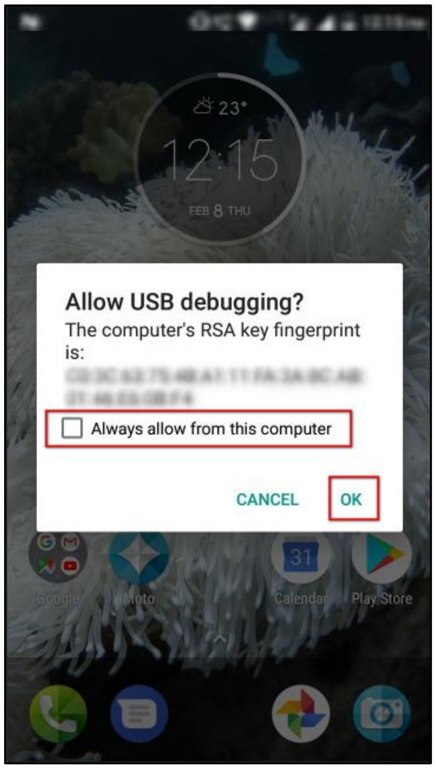
Select mock location app
Mock location app: Appium Settings

Enable view attribute inspection

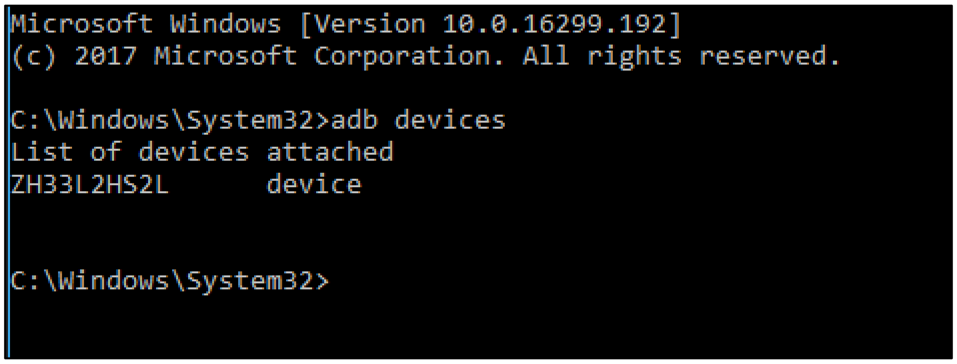
Select debug app
No debug application set

Wait for debugger

6. After enabling the Developer Options, it will ask for permission for USB debugging. Click Ok. If the user wants the system to remember, then tap on **Always allow from this computer**.

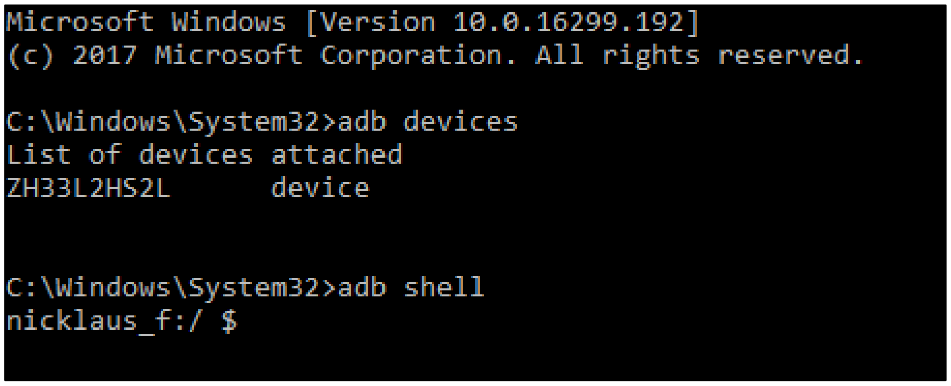


7. Open any app in the phone for which the test case is to be developed.
8. Now open a command prompt.
9. Enter the command "`adb devices`" and press Enter. It will show the devices that are connected to the system.



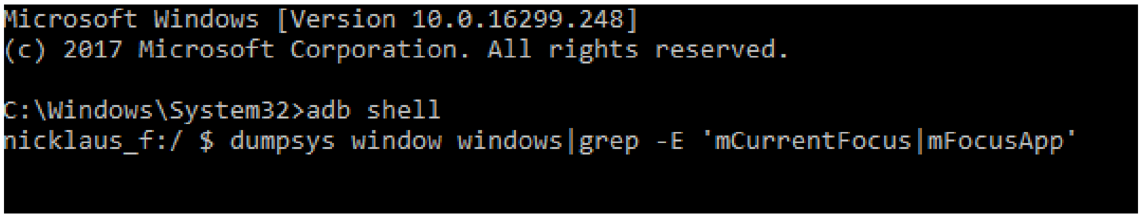
```
Microsoft Windows [Version 10.0.16299.192]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Windows\System32>adb devices
List of devices attached
ZH33L2HS2L      device


C:\Windows\System32>
```

10. Now enter a command "`adb shell`" and press Enter. It will open a shell.



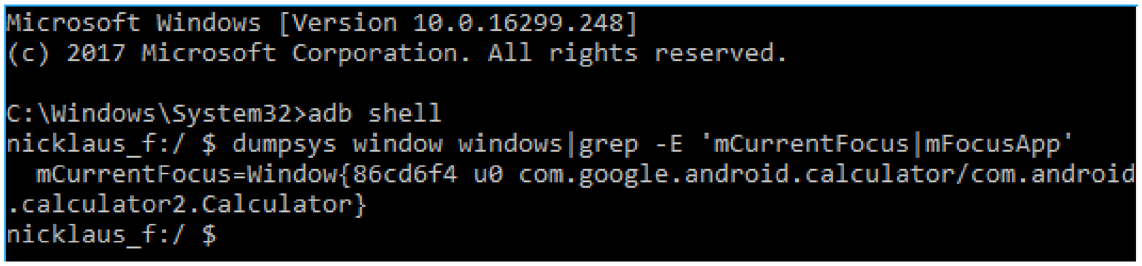
```
Microsoft Windows [Version 10.0.16299.192]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Windows\System32>adb devices
List of devices attached
ZH33L2HS2L      device


C:\Windows\System32>adb shell
nicklaus_f:/ $
```

11. In the shell, enter "`dumpsys window windows|grep -E 'mCurrentFocus|mFocusApp'`" and press Enter.



```
Microsoft Windows [Version 10.0.16299.248]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Windows\System32>adb shell
nicklaus_f:/ $ dumpsys window windows|grep -E 'mCurrentFocus|mFocusApp'
```

12. It will show the appPackage and appActivity.



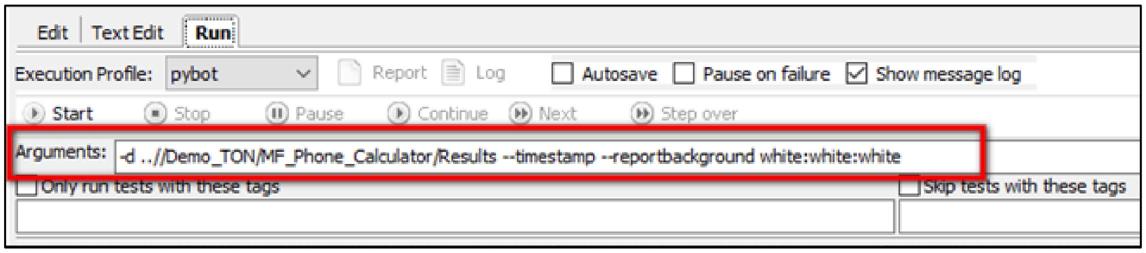
```
Microsoft Windows [Version 10.0.16299.248]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Windows\System32>adb shell
nicklaus_f:/ $ dumpsys window windows|grep -E 'mCurrentFocus|mFocusApp'
  mCurrentFocus=Window{86cd6f4 u0 com.google.android.calculator/com.android
.calculator2.Calculator}
nicklaus_f:/ $
```

13. The values for **appPackage** and **appActivity** will be different based on the app and the android device being used. The user should plug in the specific device that will be used for execution to fetch the above values.
14. For example, if the user opens Calculator app and enters the above command then it will display "`mCurrentFocus=Window{8e7b813 u0 com.google.android.calculator/com.android.calculator2.Calculator}`"
15. In the above message displayed, "`com.google.android.calculator`" is the **appPackage** and "`com.android.calculator2.Calculator`" is the **appActivity**.
16. The appPackage and appActivity will be used in the writing of Appium code.
17. After writing the code, save the file based on the above-mentioned naming conventions with a file extension of .py.

6.3.1.2 EXECUTION ON PHONE

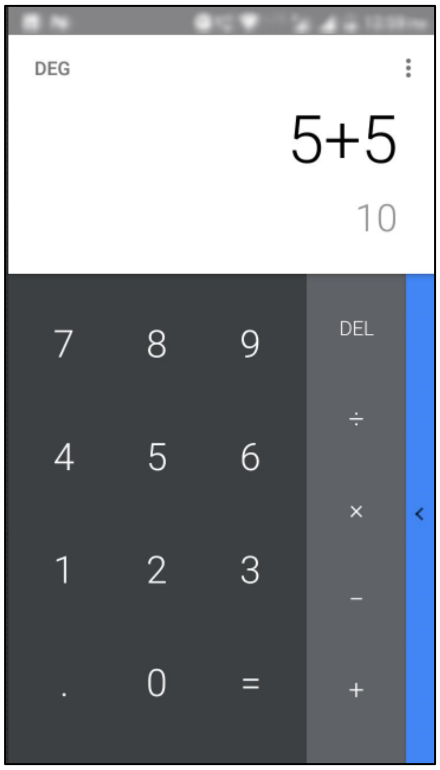
1. During execution, the same android phone that was used in the writing of code must be connected.
2. After opening RIDE, the user has to change the **Arguments**.
3. At the top of the Run tab, configure the **Arguments** textbox by entering:

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

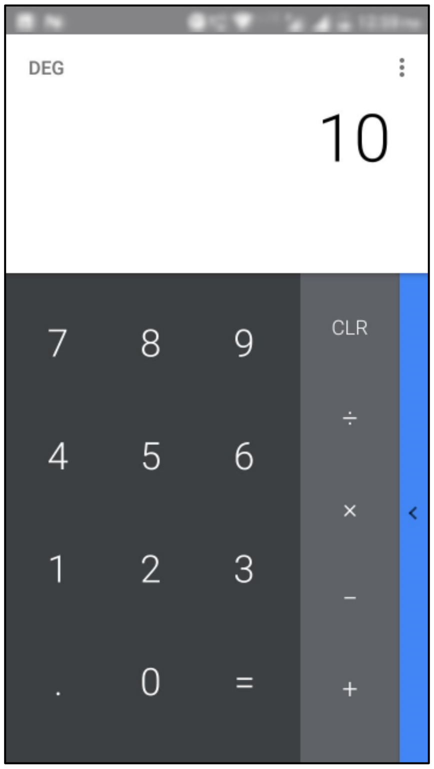


Execution Profile: pybot
Autosave [ ] Pause on failure [ ] Show message log [✓]
Arguments: -d ..//Demo_TON/MF_Phone_Calculator/Results --timestamp --reportbackground white:white:white
Only run tests with these tags [ ]
Skip tests with these tags [ ]

4. Then run the code as mentioned in the Execute Test Part.
5. The user can see the code running in their phone.
6. The example shows for addition of 2 numbers in the Calculator app of the android phone.
7. Firstly, it will open the Calculator app.
8. Then it will enter then numbers mentioned in the code one by one with a plus sign in between.
9. It will add both the numbers.



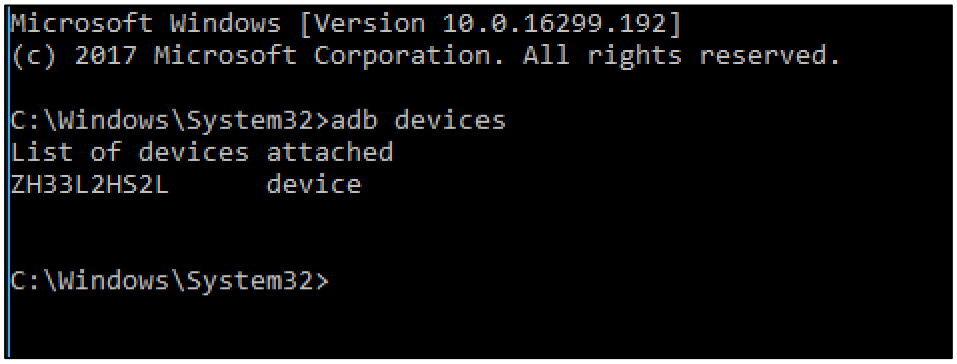
10. Then it will display the result.


DEG
10
7
8
9
CLR
4
5
6
÷
1
2
3
×
.
0
=
−
+

6.3.2 THROUGH GENYMOTION

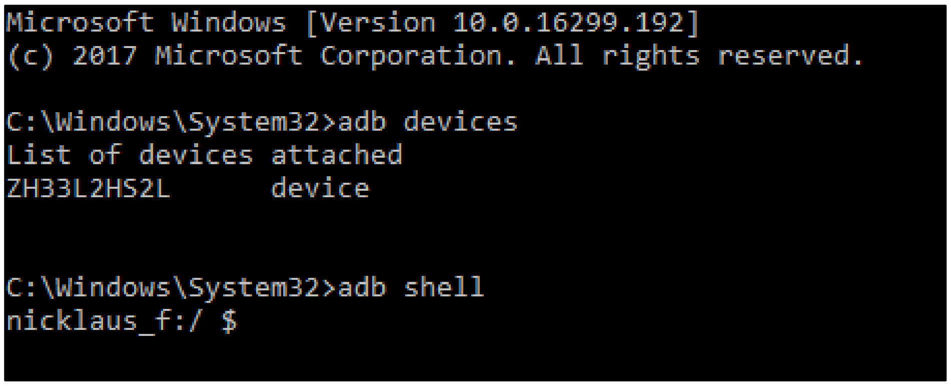
6.2.2.1 CONNECTING TO GENYMOTION

1. Now open a command prompt.
2. Enter the command "`adb devices`" and press Enter. It will show the devices that are connected to the system.



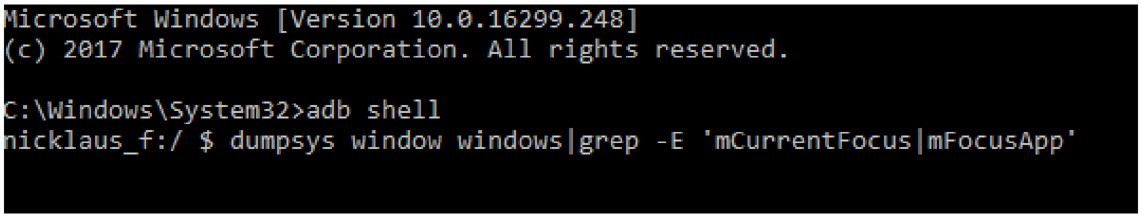
```
Microsoft Windows [Version 10.0.16299.192]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Windows\System32>adb devices
List of devices attached
ZH33L2HS2L      device


C:\Windows\System32>
```

3. Now enter a command "`adb shell`" and press Enter. It will open a shell.



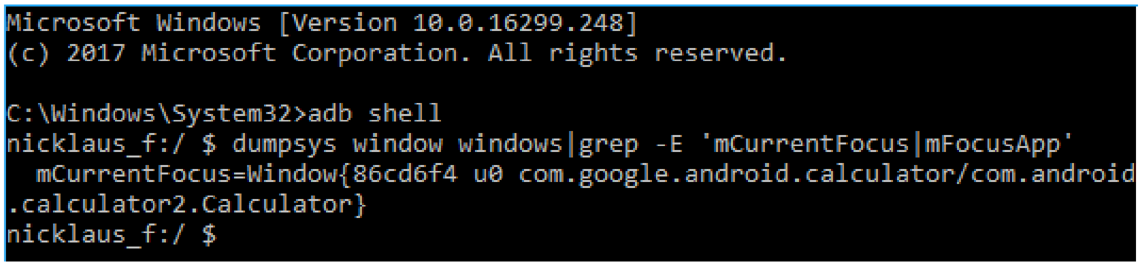
```
Microsoft Windows [Version 10.0.16299.192]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Windows\System32>adb devices
List of devices attached
ZH33L2HS2L      device


C:\Windows\System32>adb shell
nicklaus_f:/ $
```

4. In the shell, enter "`dumpsys window windows|grep -E 'mCurrentFocus|mFocusApp'`" and press Enter.



```
Microsoft Windows [Version 10.0.16299.248]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Windows\System32>adb shell
nicklaus_f:/ $ dumpsys window windows|grep -E 'mCurrentFocus|mFocusApp'
```

5. It will show the appPackage and appActivity.

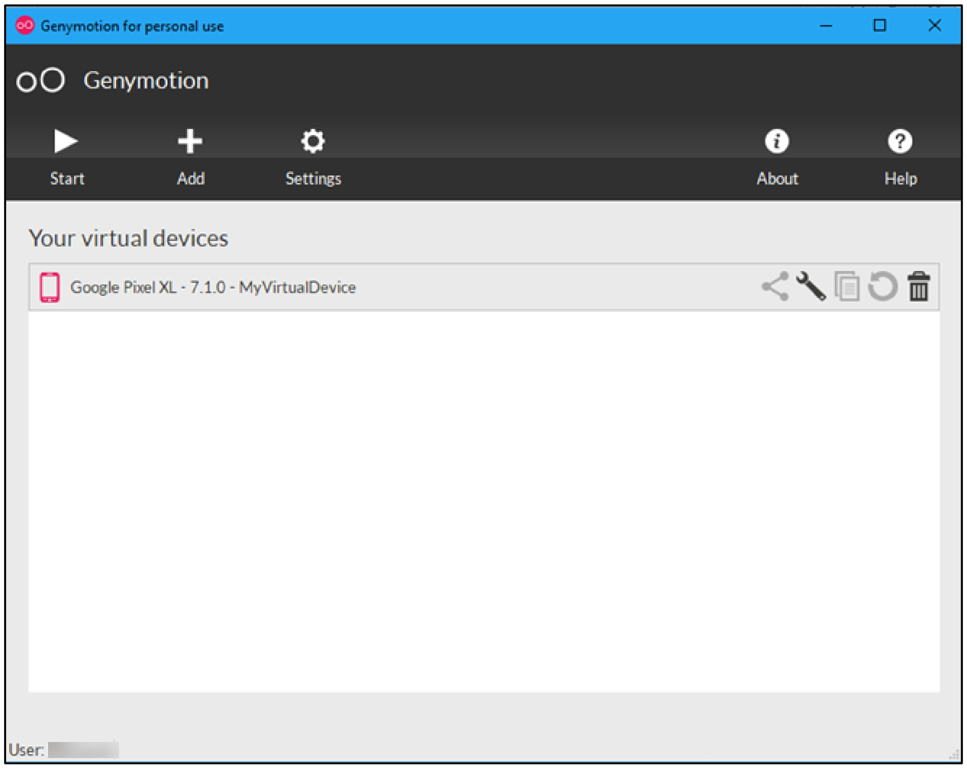


```
Microsoft Windows [Version 10.0.16299.248]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Windows\System32>adb shell
nicklaus_f:/ $ dumpsys window windows|grep -E 'mCurrentFocus|mFocusApp'
  mCurrentFocus=Window{86cd6f4 u0 com.google.android.calculator/com.android
.calculator2.Calculator}
nicklaus_f:/ $
```

6. The values for **appPackage** and **appActivity** will be different based on the app and the android device being used. The user should plug in the specific device that will be used for execution to fetch the above values.
7. For example, if the user opens Calculator app and enters the above command then it will display "`mCurrentFocus=Window{8e7b813 u0 com.google.android.calculator/com.android.calculator2.Calculator}`"
8. In the above message displayed, "`com.google.android.calculator`" is the **appPackage** and "`com.android.calculator2.Calculator`" is the **appActivity**.
9. The appPackage and appActivity will be used in the writing of Appium code.
10. After writing the code, save the file based on the above-mentioned naming conventions with a file extension of .py.

6.3.2.2 EXECUTION ON GENYMOTION

1. Start the Genymotion.

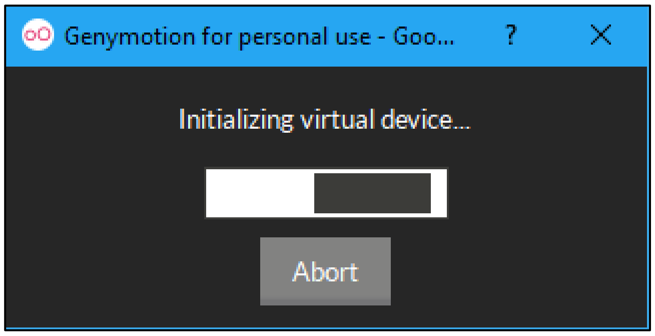


Your virtual devices

Google Pixel XL - 7.1.0 - MyVirtualDevice

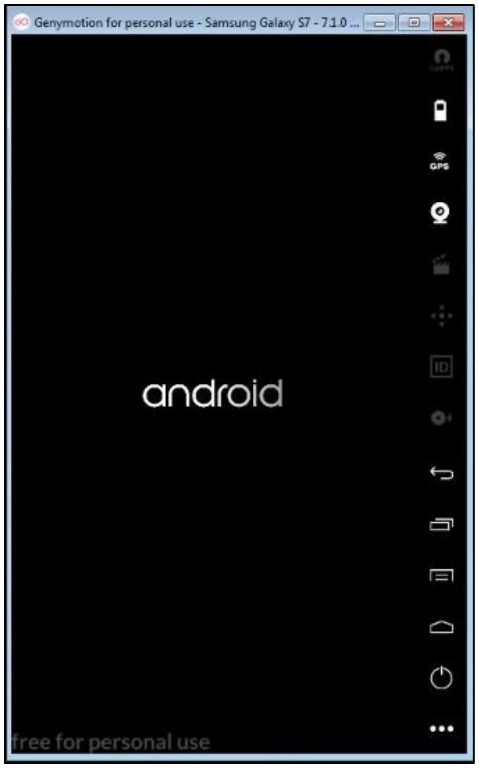
User:

2. Click on the virtual device to start the execution.

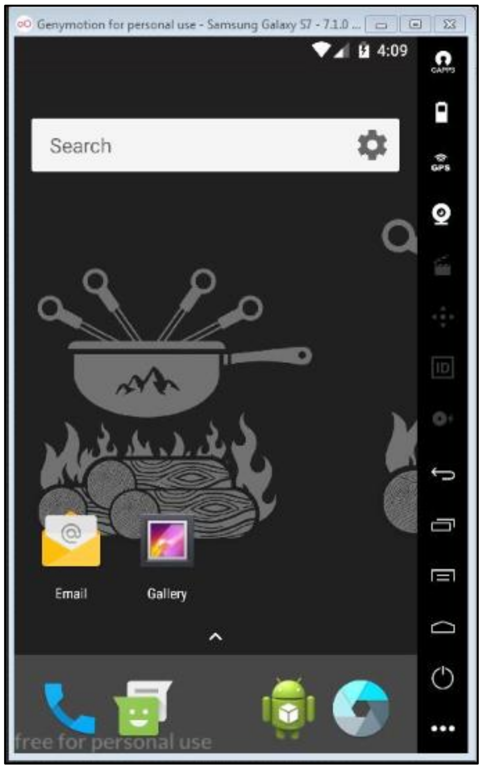


Initializing virtual device...

3. It will start the virtual device. It may take some time.

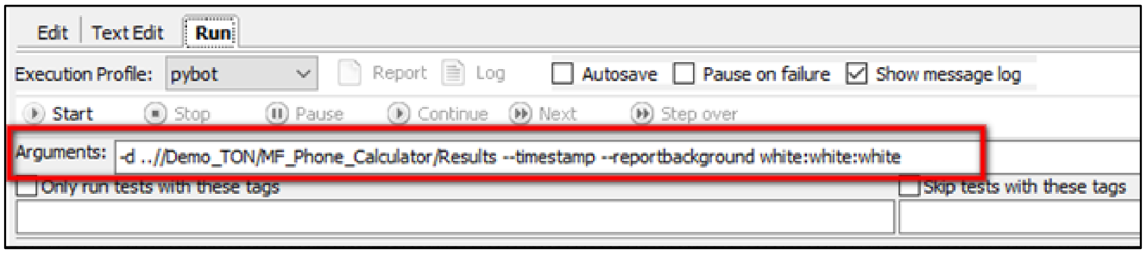


4. Once it starts, the user can execute the code.



11. After opening RIDE, the user has to change the **Arguments**.

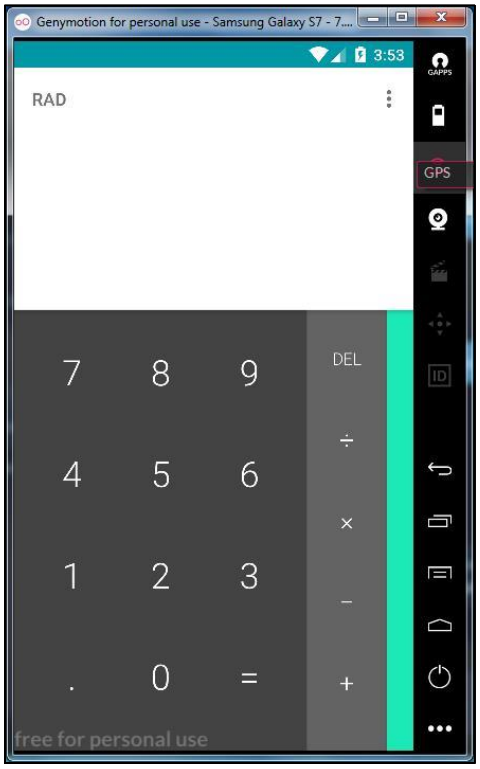
12. At the top of the Run tab, configure the **Arguments** textbox by entering:
-d ..\\Demo_TON\<test_case_name>\Results --timestamp --reportbackground white:white:white -t



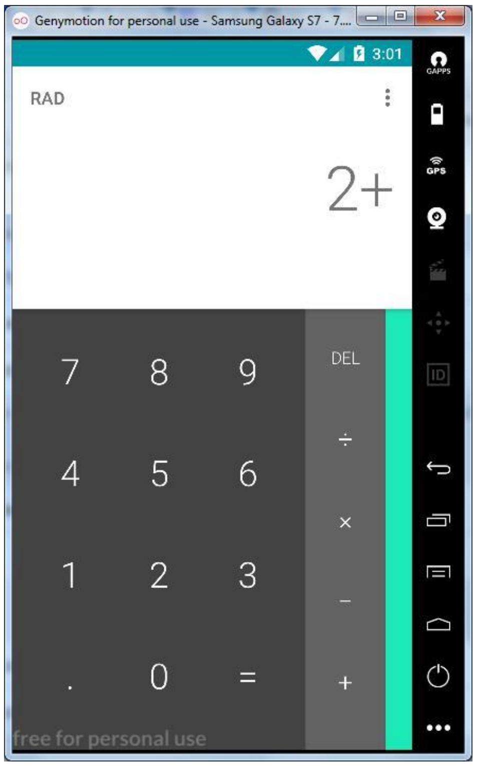
Execution Profile: pybot

Arguments: -d ..//Demo_TON/MF_Phone_Calculator/Results --timestamp --reportbackground white:white:white

5. Then run the code as mentioned in the Execute Test Part.
6. The example shows for addition of 2 numbers in the Calculator app of the virtual device.
7. Firstly, it will open the Calculator app.



8. Then it will enter then numbers mentioned in the code one by one with a plus sign in between.



9. It will add both the numbers.
10. Then it will display the result.