



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

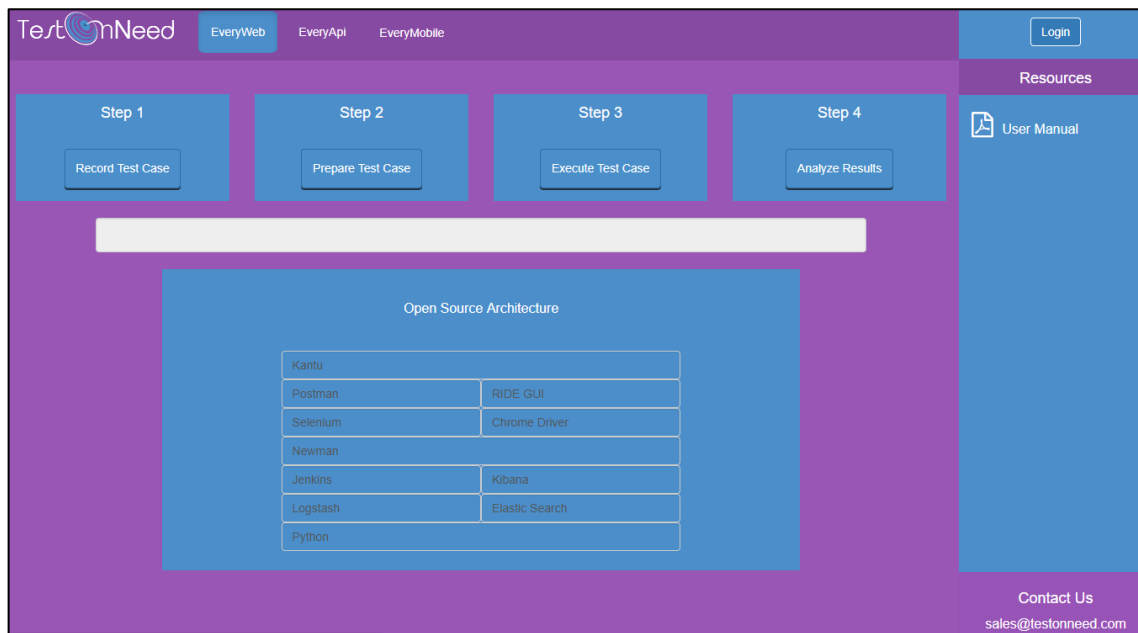
Reviewer	Version Reviewed	Signature	Date
Angelo Virgilio	0.3		June 9, 2017

Table of Contents

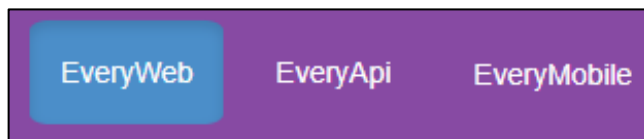
1. Description Of The Web Page	3
2. EveryWeb Page	5
2.1 Record Test Case.....	6
2.1.1 Record - Using Browser	7
2.1.2 Record - Using API.....	11
2.2 Prepare Test Case	17
2.2.1 Prepare – For Browser:.....	18
2.2.2 Prepare – For API	20
2.3 Execute Test Case	23
2.3.1 Execute – Via GUI.....	24
2.3.2 Execute - Via TestOps	49
2.4 Analyze Results.....	52
3. References.....	54
3.1 Creation of New View	54
3.2 Creation of New Job	56
3.3 Setting up of Proxy in Chrome.....	59
3.4 Changing Folder Location of Postman	74

1. DESCRIPTION OF THE WEB PAGE

- The outline of the Website looks like below:



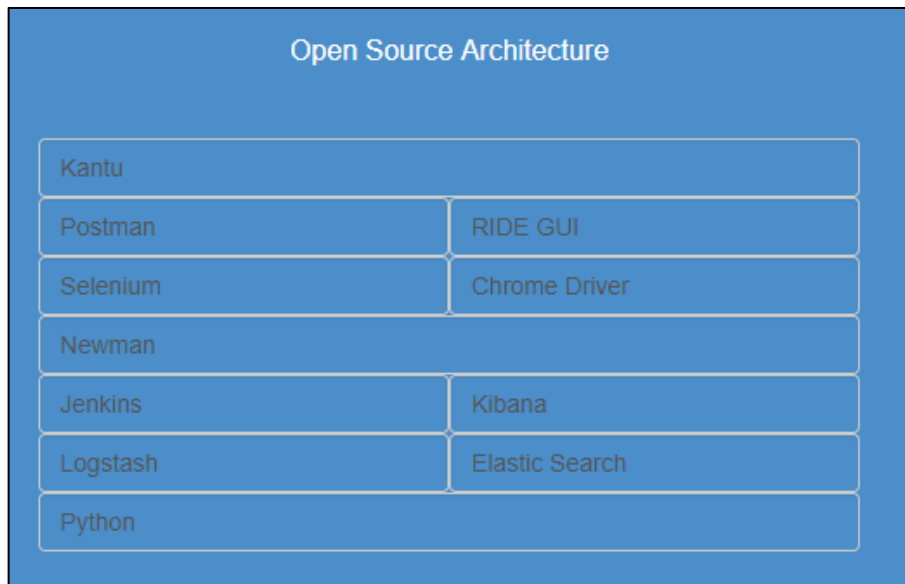
- As the web page opens up, there will be three tabs namely – EveryWeb, EveryAPI and EveryMobile.



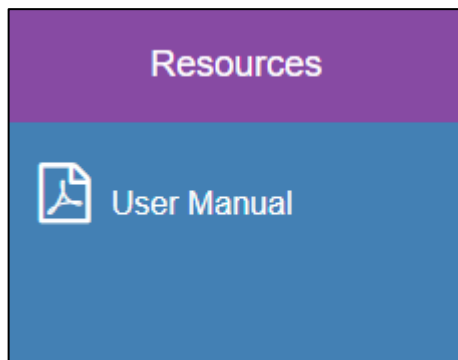
- Things that are common to all the tabs are the stack which contains all the open sources, a status bar and the user manual.
- In the status bar, the user will get know what is being done next by clicking any button.



- The stack of open sources will contain all the 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.

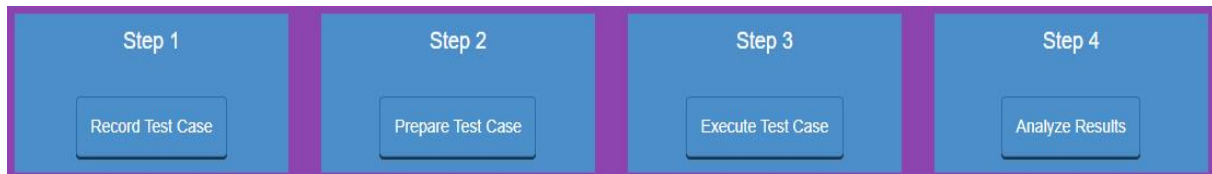


- The user manual guides the user through the web application. It will also change from tab to tab since working of each tab is different. It is a hyperlink, on clicking it, will open the respective document in pdf format in a new tab in the browser.



2. EVERYWEB PAGE

- The application opens with EveryWeb page. The default tab is the web tab.
- This tab has 4 buttons on the top namely –
 1. Record Test Case
 2. Prepare Test Case
 3. Execute Test Case
 4. Analyze Results.



- Each step is explained below

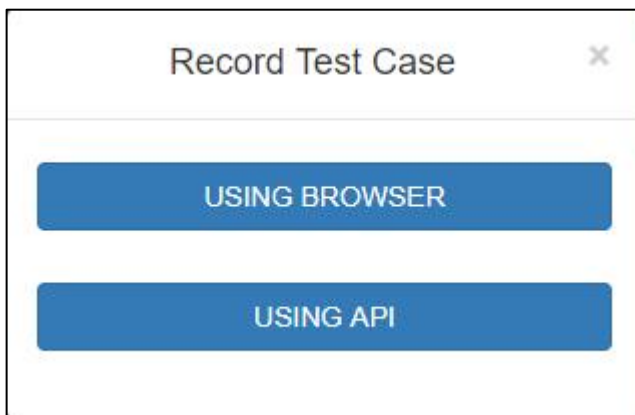
2.1 RECORD TEST CASE

The Record Test Case is used to record the scenario which is used for the further testing process.

1. Click the **Record Test Case**.



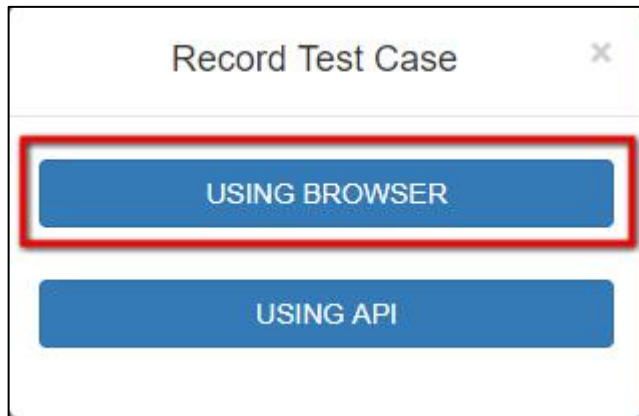
2. On clicking the **Record Test Case** button, a pop-up appears with 2 buttons.
3. The **Using Browser** button is used to record the process from the browser or frontend.
4. The **Using API** button is used to record the process or the services that are being called on the backend.



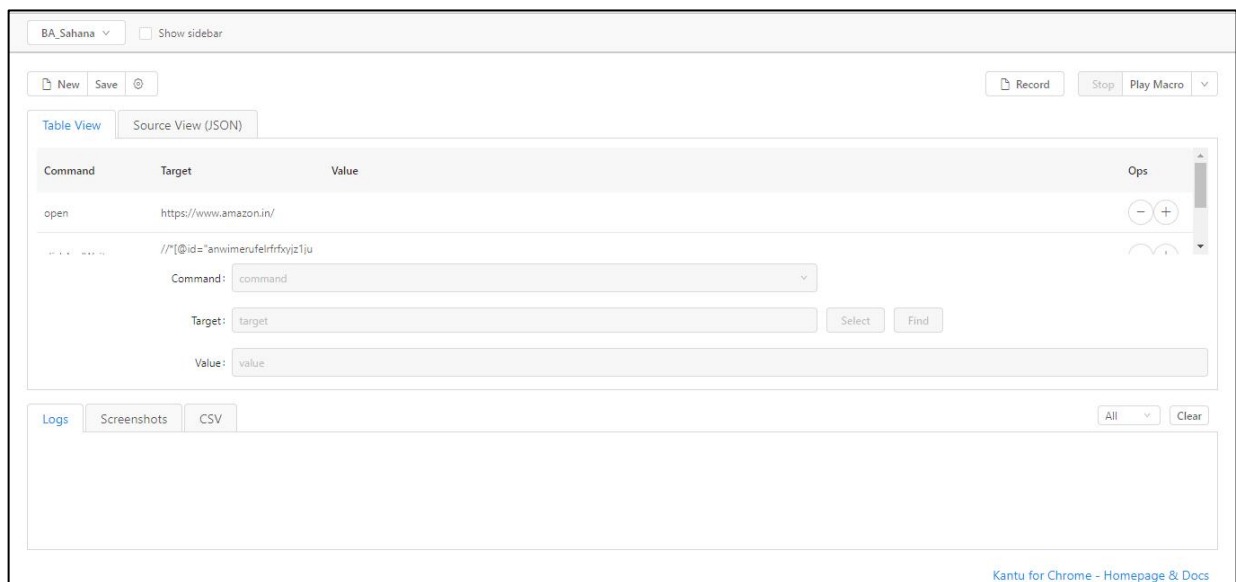
5. From the pop-up, user can choose either of **Using Browser** and **Using API** based on the requirement.

2.1.1 RECORD - USING BROWSER

1. Click on **Using Browser**.



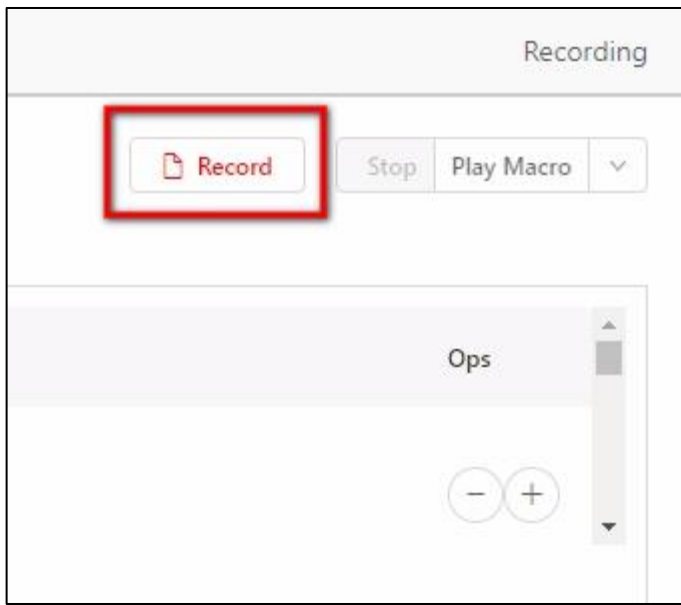
2. On clicking the button will open the Kantu extension on a new window of the Chrome browser.



3. Click on the **Record** button.



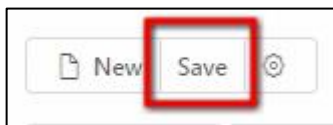
4. On clicking the Record button, the button will turn red.



5. Now open the web application that has to be recorded.
6. Perform the steps.
7. To stop click the Record button again. It will change to blue from red.



8. Now save the recording by clicking the save button.

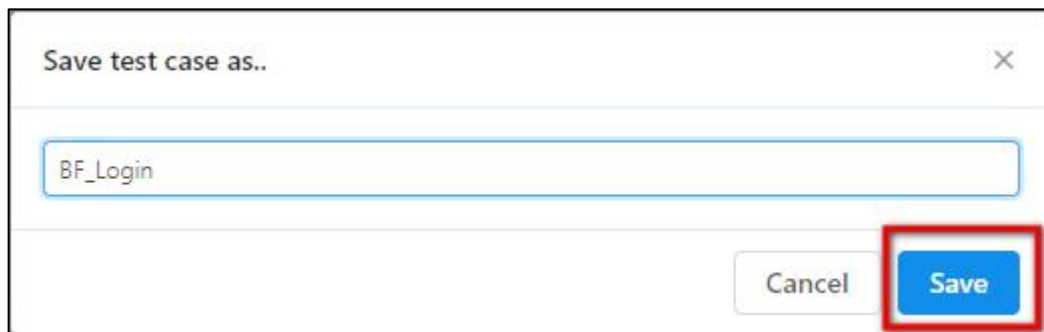


9. Clicking on the save button a pop up comes up where name of the test case has to be entered based on the naming conventions as shown below



10. Now save the recording based on the type of testing functionality that has been recorded. There are 3 types of testing functionality– **Function, Automation and Load**.

11. Functional Testing is a testing technique that is used to test the features/functionality of the system or Software, should cover all the scenarios including failure paths and boundary cases. In that case, the test case should be saved as **BF_<TestCase_Name>**. For example, the recorded scenario is log in, then the test case should be saved as **BF_Login**.



12. Automation testing makes use of specialized tools to control the execution of tests and compares the actual results against the expected result. In that case, the test case should be saved as **BA_<TestCase_Name>**. For example, the recorded scenario is both signup and login, then the test case should be saved as **BF_LoginGateways**.



13. Load testing is performance testing technique using which the response of the system is measured under various load conditions. The load testing is performed for normal and peak load conditions. In that case, the test case should be saved as **BL_<TestCase_Name>**. For example, to test the performance of the login scenario that at a time how many users can log in, the test case should be saved as **BL_LoginLoad**.

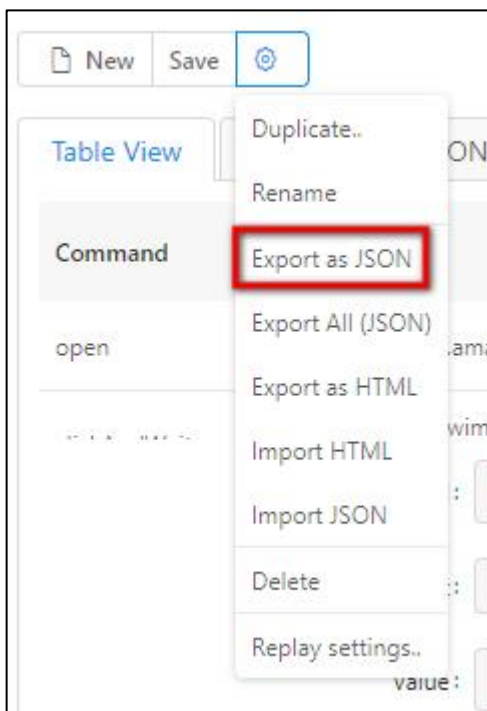


14. On giving the appropriate name, then click save to save the test case.

15. Click the setting button.



16. On clicking the button, a drop-down will appear, from that select **Export as JSON**.

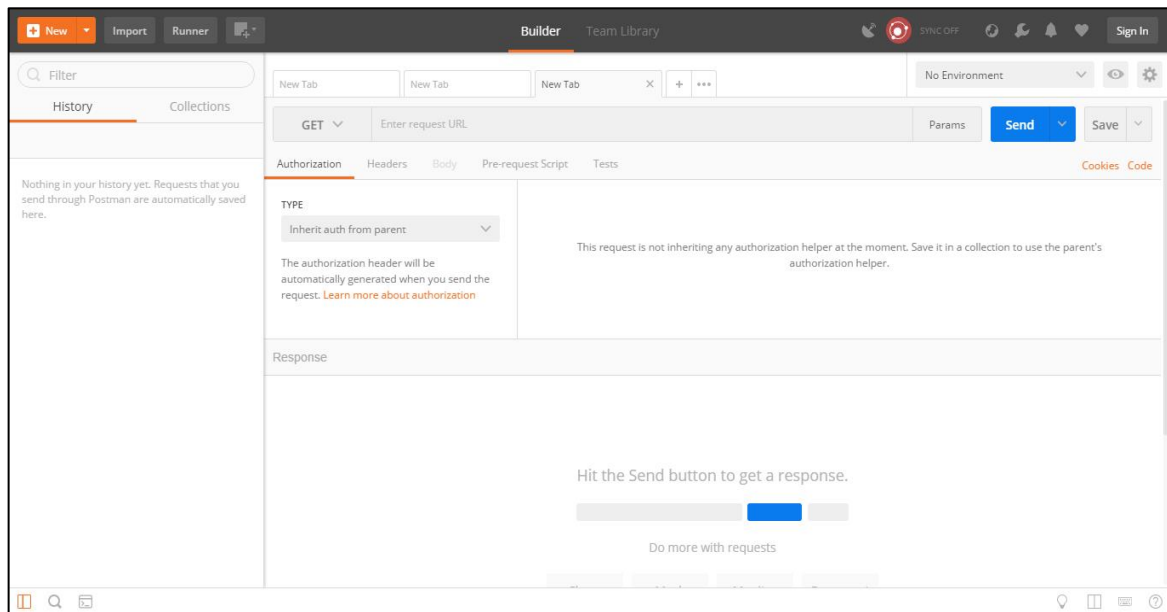


17. Now the JSON file gets downloaded in the default downloads folder.

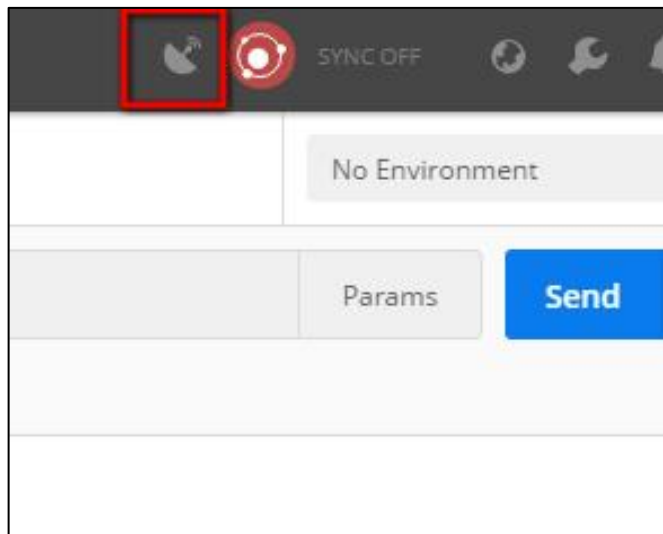
NOTE: The export should only be done in the JSON format from the Kantu browser.

2.1.2 RECORD - USING API

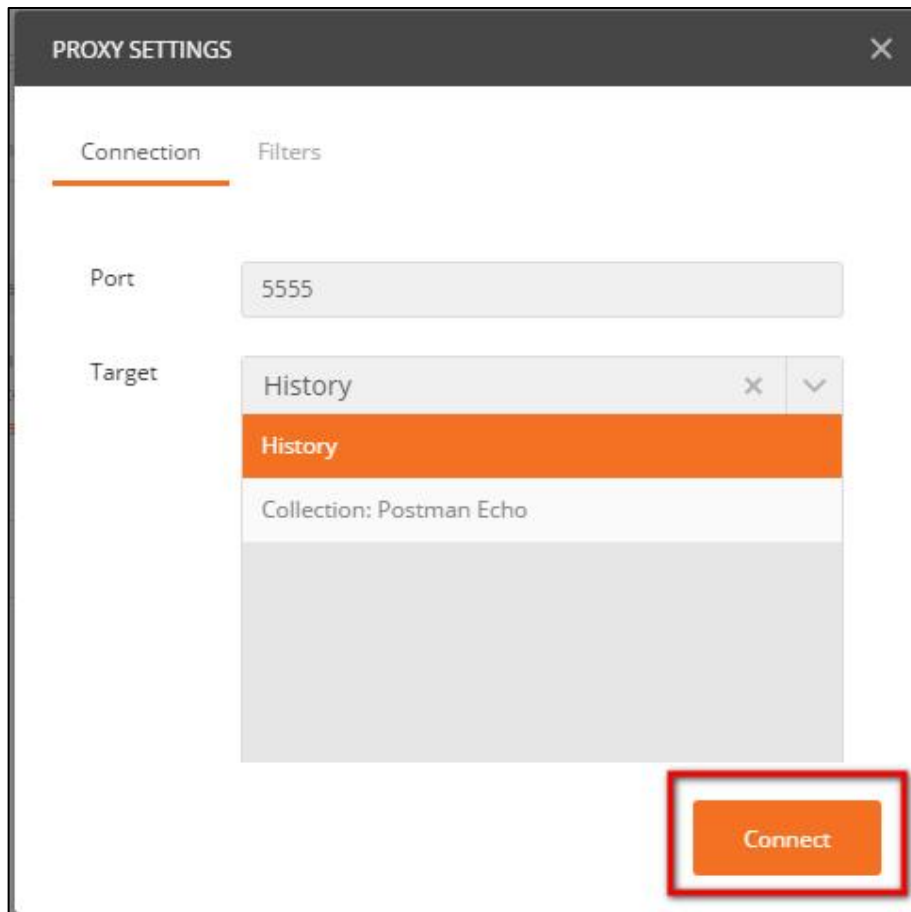
1. The user has to change the folder location of the postman. To change the folder location, click [Here](#).
2. For setting up of proxy in order to record, click [Here](#).
3. On clicking **Using API** button will open the Postman app.



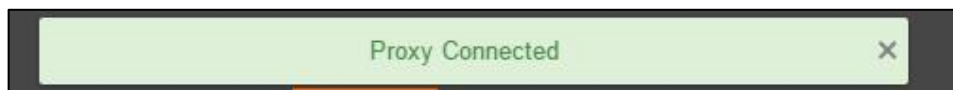
4. Start the proxy in postman application.



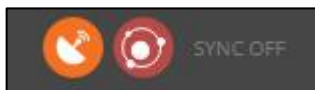
5. On clicking the proxy button, a pop window comes up and click **Connect** on it.



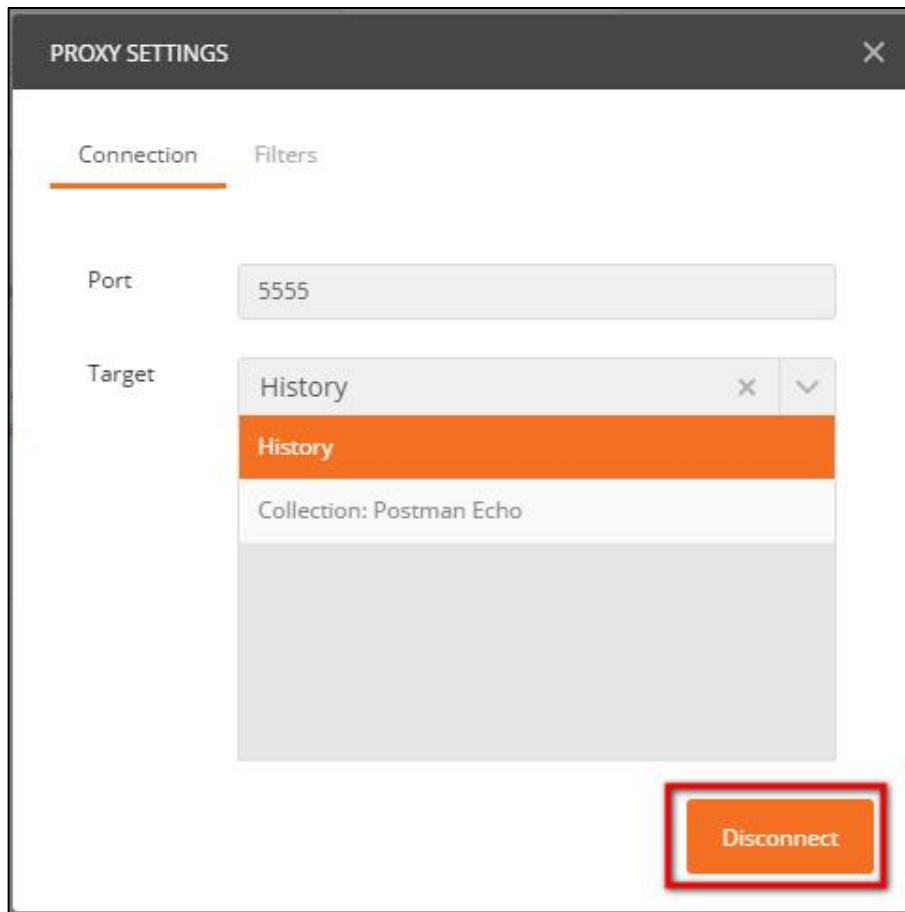
6. Once the connect button is clicked, at the top of the page a message will be displayed as **Proxy Connected** if connected.



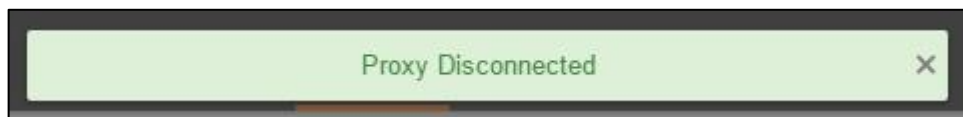
7. If it is connected, the proxy button changes to orange.



8. Now open the site that has to be record.
9. Now perform the steps on the site.
10. To stop the recording, stop the proxy in the postman application by clicking the proxy button again. The pop-up window comes and click Disconnect.



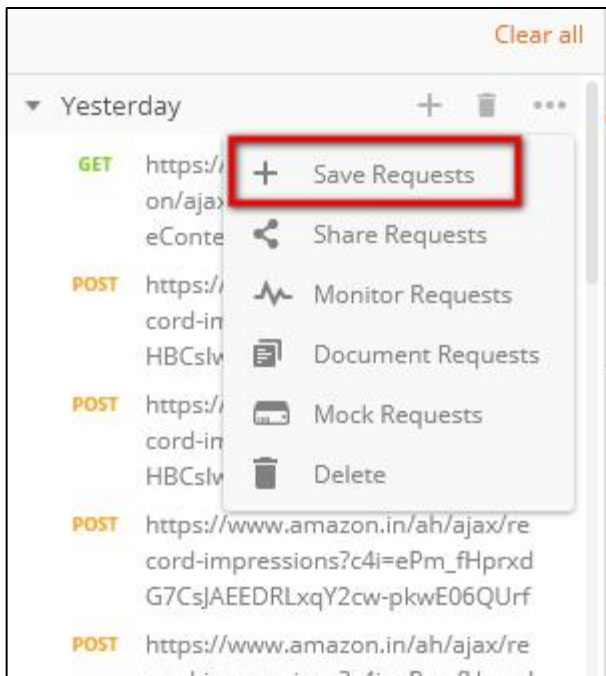
11. Once the disconnect button is clicked, at the top of the page a message will be displayed as **Proxy Disconnected**.



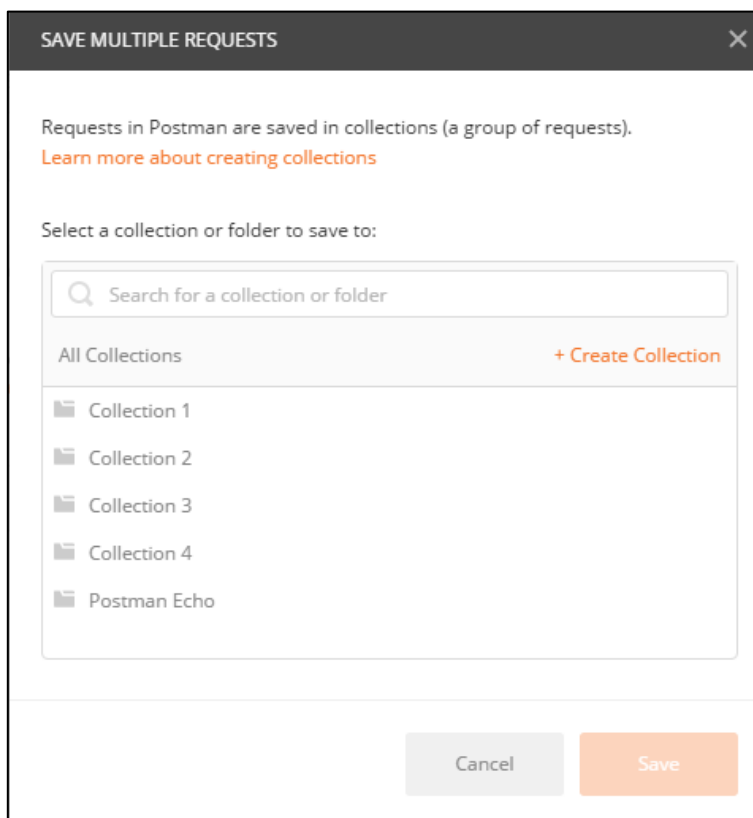
12. After disconnecting the proxy, the proxy button changes back to grey.



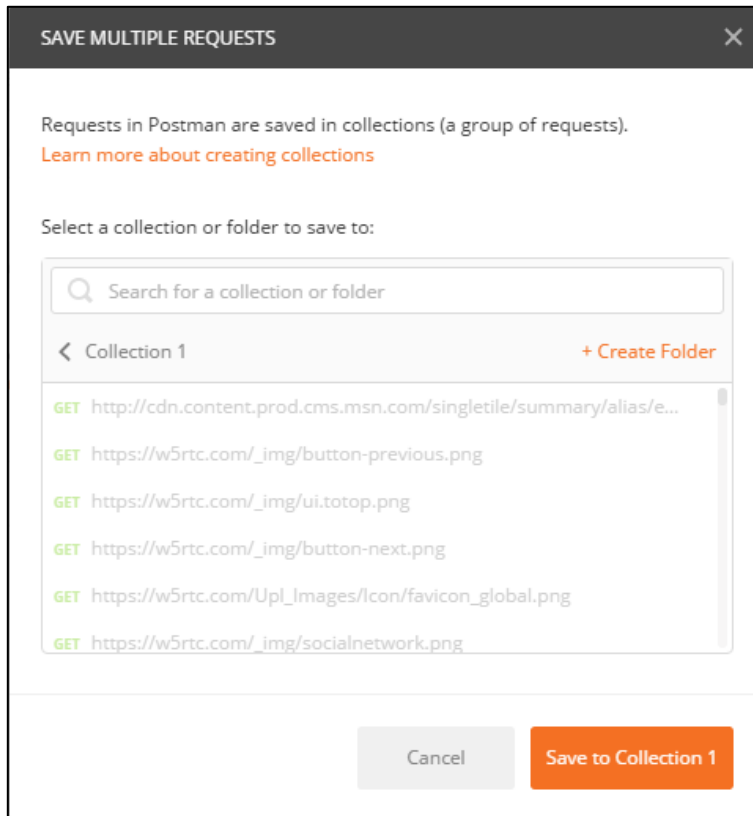
13. In the postman application, click Save Request and save in the appropriate collection.



14. On clicking of the Save Requests button, a dialog box appears.



15. At the bottom of the dialog box, select the collection name in which the test cases have to be saved. If the user wants to save under a new collection name click Create Collection.
16. On click of the collection name, the Save button becomes active.



17. Now save the recording based on the type of testing functionality the user has recorded. There are 3 types of testing functionality – **Function, Automation and Load**.
18. Functional Testing is a testing technique that is used to test the features/functionality of the system or Software, should cover all the scenarios including failure paths and boundary cases. In that case, the testcase should be saved as **AF_<TestCase_Name>**. For example, if the user has recorded the scenario of login, then the user has to save the test case as **AF_Login**.
19. Automation testing makes use of specialized tools to control the execution of tests and compares the actual results against the expected result. In that case, the test case should be saved as **AA_<TestCase_Name>**. For example, if the user has recorded the scenario of both signup and login, then the user has to save the test case as **AA_LoginGateways**.

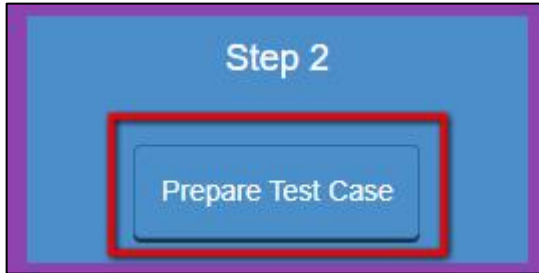
20. When recording for **Automation** testing, the user has to always **record multiple testcases**.
21. Load testing is performance testing technique using which the response of the system is measured under various load conditions. The load testing is performed for normal and peak load conditions. In that case, the test case should be saved as **AL_<TestCase_Name>**. For example, if the user wants to test the performance of the login scenario that at a time how many users can log in, in such case the user has to save the test case as **AL_LoginLoad**.
22. Right-click on the saved link, click Export (it will automatically be exported as JSON format).

NOTE: ONLY IN THE CASE OF LOAD TESTING, JSON FILE SHOULD BE DOWNLOADED WHILE DOWNLOADING USING POSTMAN.

2.2 PREPARE TEST CASE

The Prepare Test Case button is used for conversion of the downloaded JSON file to its corresponding test case files.

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



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



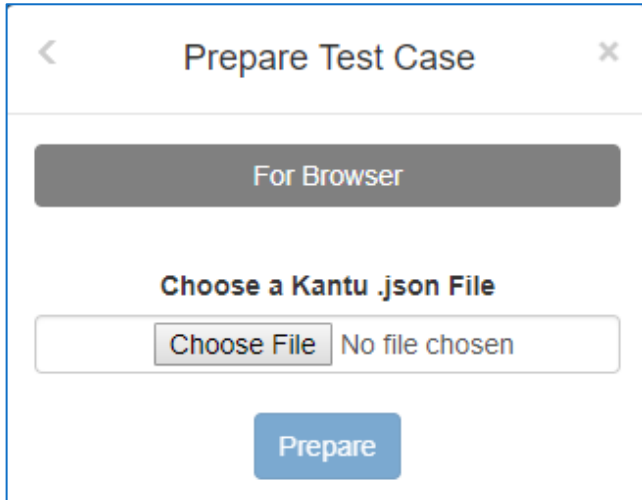
3. To upload the JSON file downloaded using the Kantu browser in the previous step then click **FOR BROWSER**.
4. To upload the JSON file downloaded using the Postman in the previous step then click **FOR API**.
5. From the pop up select either – **FOR BROWSER** or **FOR API** button.

2.2.1 PREPARE – FOR BROWSER:

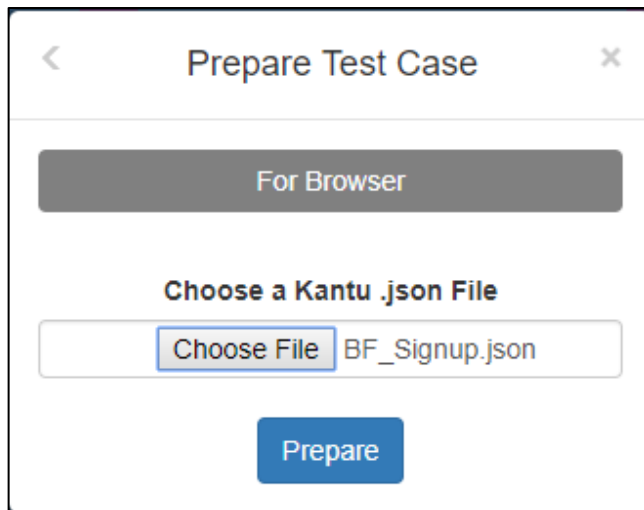
1. Click on the **For Browser** button.



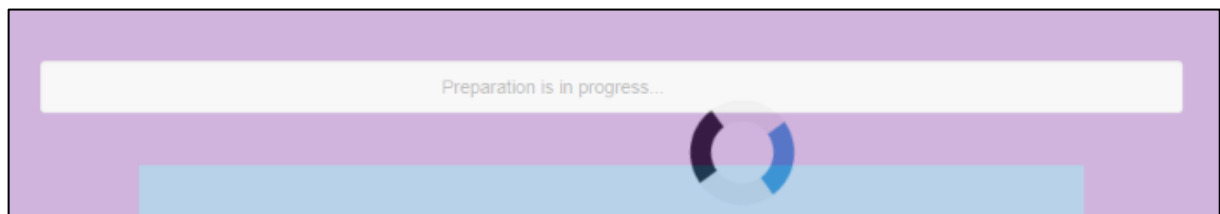
2. On clicking **For Browser** button, **Choose File** button appears.



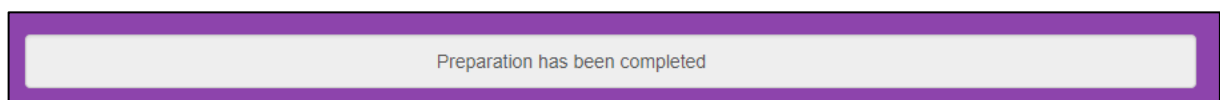
3. Click on the **Choose File** button.
4. Surfing the folder location where saved the JSON file that was downloaded in the previous step.
5. Now upload the JSON file. The user can upload **only one file** at a time.
6. On uploading the JSON file, the **Prepare** button becomes active.



7. 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 > WebTesting > Browser > GUI > Demo_TON ><TestCase_Name_Folder>**.
 - In case if the test case name was saved as **BF_Login** in the previous step then it will create a folder with same name **BF_Login**.
 - In this folder, the JSON file that was uploaded and its corresponding Python file will be found.
8. 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.



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

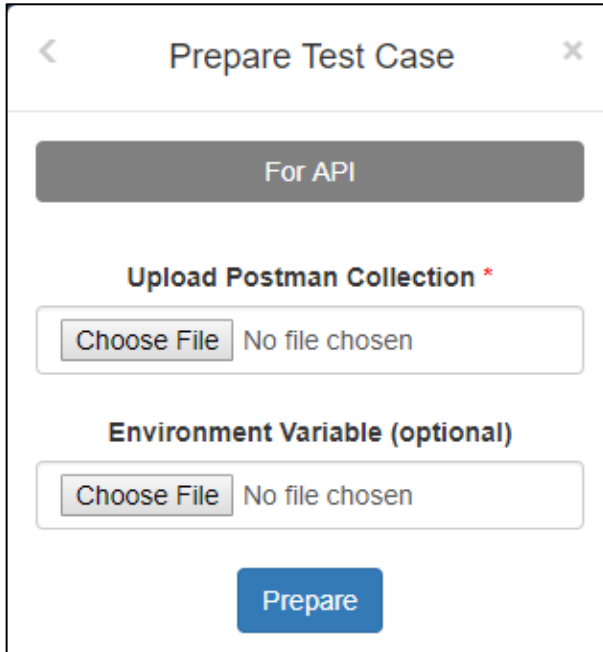


2.2.2 PREPARE – FOR API

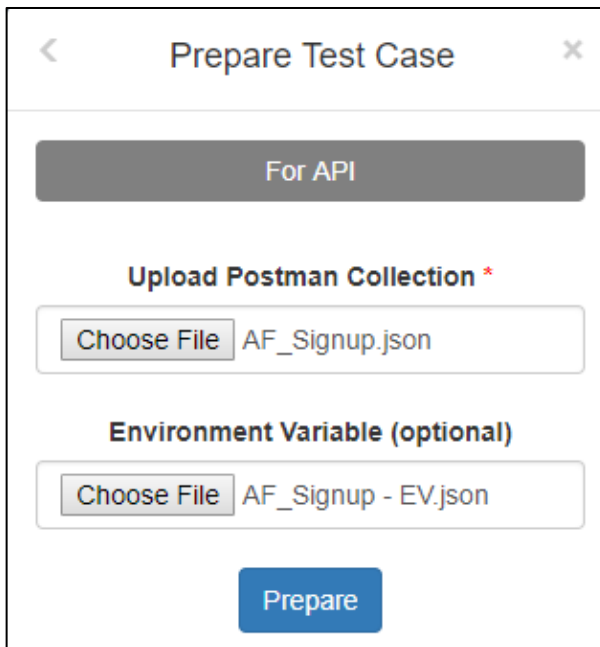
1. Click on the **For API** button.



2. On clicking of **FOR API** button, **Choose File** button appears.

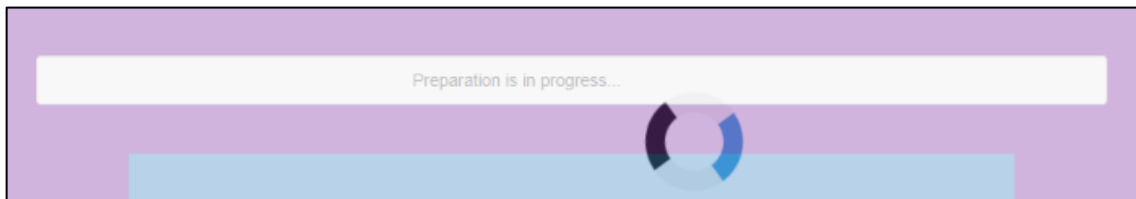


3. Click on the **Choose File** button.
4. Surfing the folder location where saved the JSON file that was downloaded in the previous step.
5. Now upload the JSON file.
6. The user can upload **only one file** at a time.
7. On uploading the JSON file, the **Prepare** button becomes active.



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 > WebTesting >API > GUI > Demo_TON > < TestCase_Name_Folder >
 - In case if the test case name was saved as **BF_LoginGateway** in the previous step then it will create a folder with the same name **BF_LoginGateway**.
 - In this folder, the JSON file that was uploaded and its corresponding Python and JS file 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.



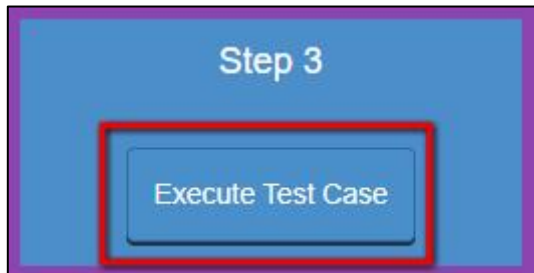
10. Once preparation is finished, a message will be shown as “**Preparation has been completed**” in the status bar.

Preparation has been completed

2.3 EXECUTE TEST CASE

The Execute Test Case button is used for the execution process using RIDE or Postman.

1. Click on the Execute Test Case button.

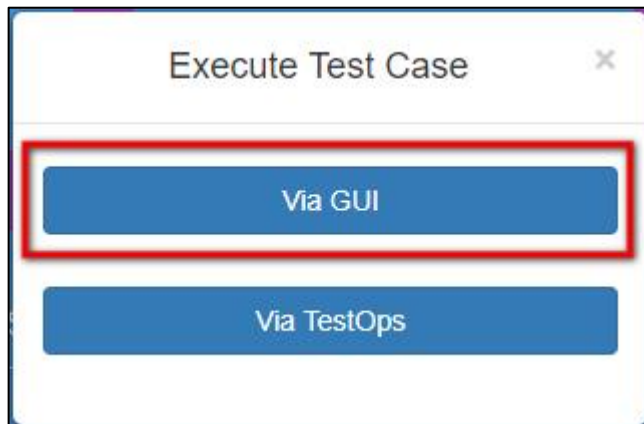


2. On clicking on the button, a pop up appears.
3. From the pop-up, select the button either as **Via GUI**, **Via TESTOPS**.

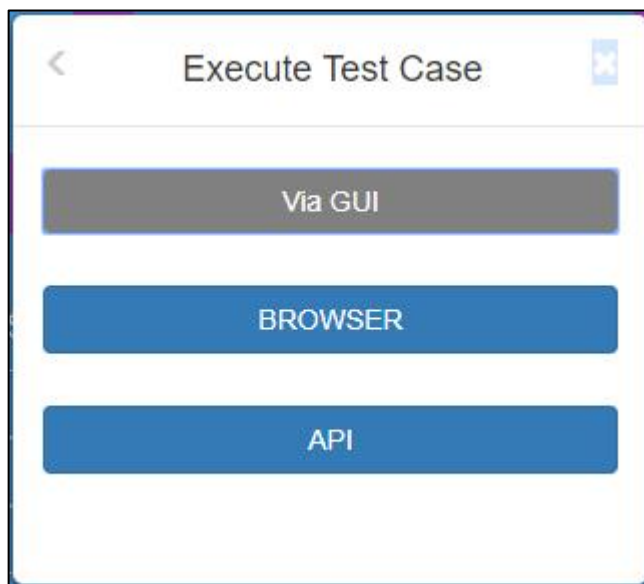


2.3.1 EXECUTE – VIA GUI

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



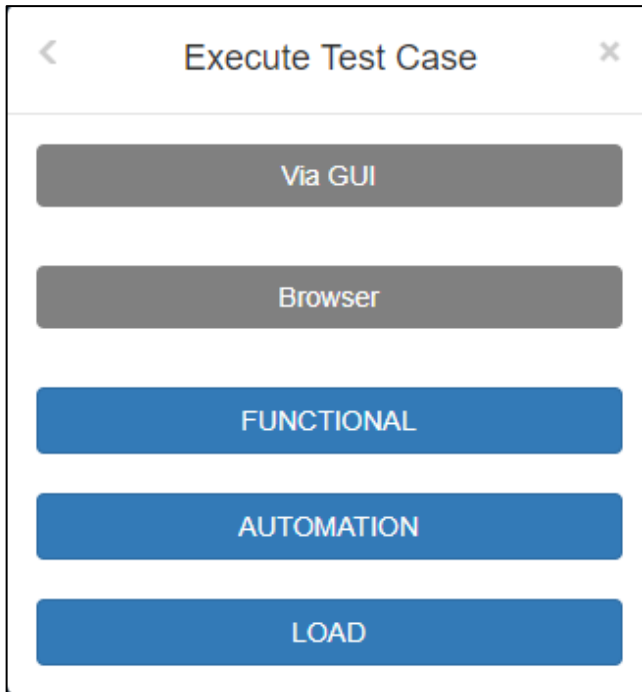
2. On clicking the button, execution through GUI can be done using 2 buttons– **Browser** and **API**.



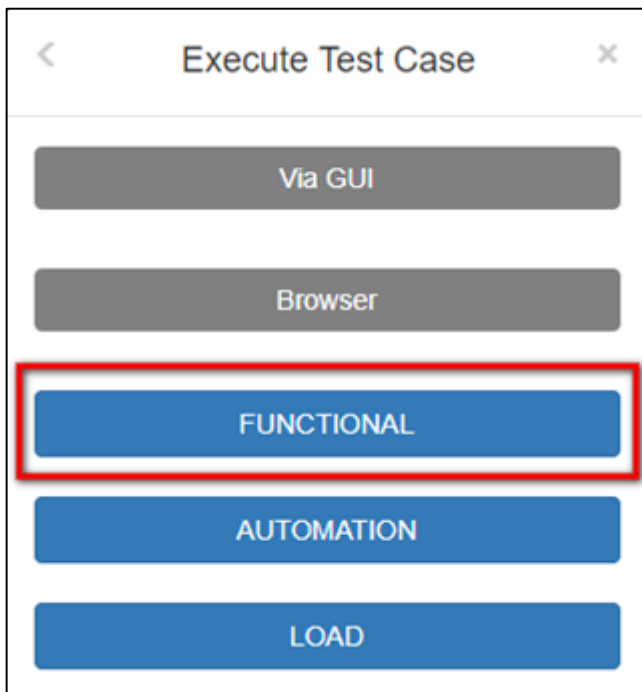
3. Under each button, 3 buttons – **Function**, **Automation** and **Load** will be found.

2.3.1.1 EXECUTE – VIA GUI – BROWSER

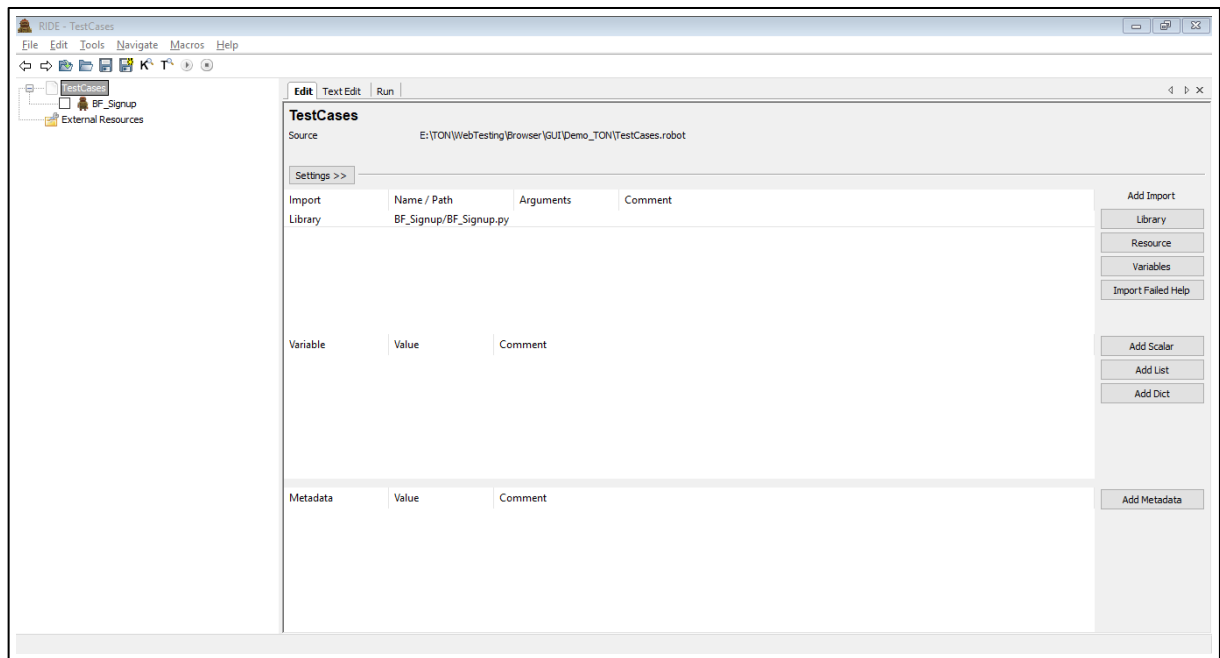
1. On click of the Via GUI button, 3 buttons appear – **Functional**, **Automation** and **Load**.



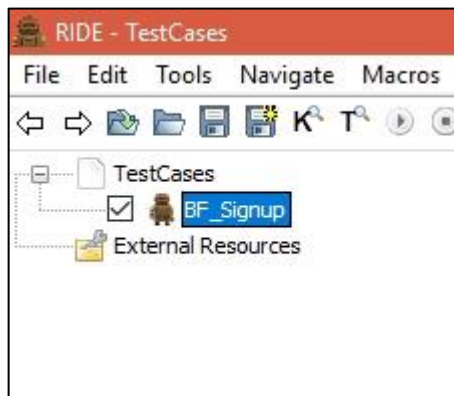
2. To perform functional testing, press **Functional** button.



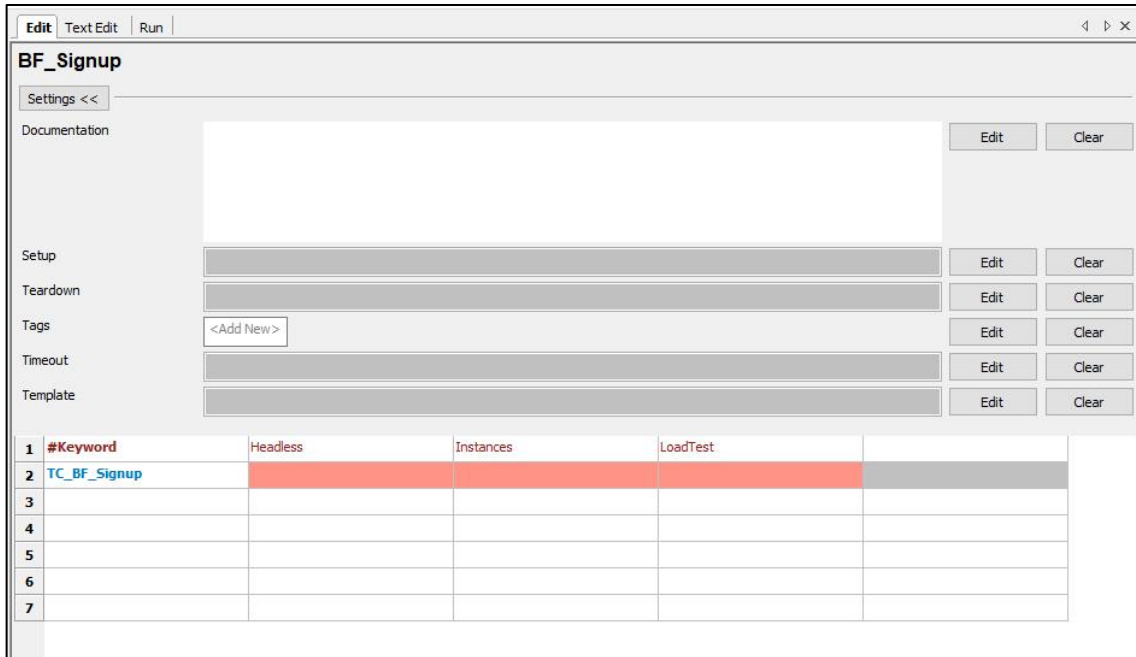
3. On click of the button, **RIDE** will open.



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



5. To configure the parameters, click on the test case name.
6. On click of the test case name, Edit tab changes so that parameters can be configured.



#	Keyword	Headless	Instances	LoadTest	
2	TC_BF_Signup				
3					
4					
5					
6					
7					

7. The value of the parameters should be:

Headless = No / Yes | Instances = 0 | LoadTest = No

For Headless = Yes, browser simulation takes place without visible browser.

For Headless = No, browser simulation takes place with the visible browser.

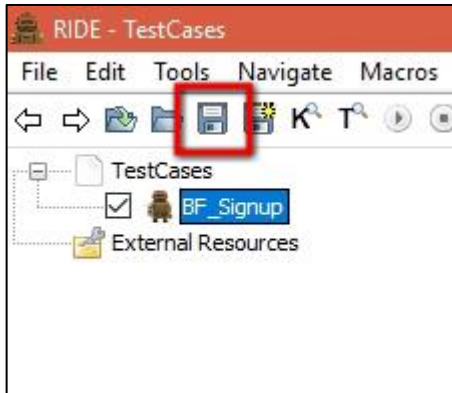
Instances = 0 (Only valid for Load Test)

LoadTest = No (Only valid for Load Test)

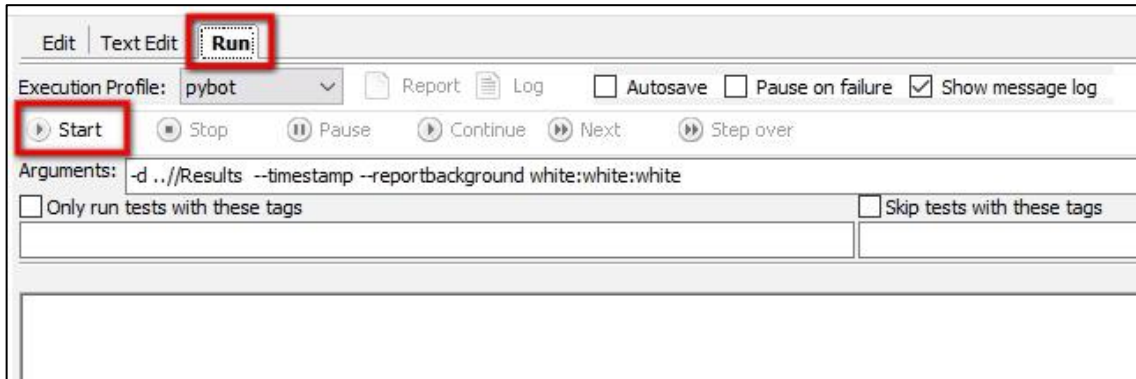
8. Configure the parameters – **Headless, Instances and LoadTest.**

1	#Keyword	Headless	Instances	LoadTest	
2	TC_BF_Signup	NO	0	NO	
3					
4					
5					
6					
7					

9. Click **Save** button to save the test case.



10. Configure the **Arguments** textbox by entering:
`-d ../Demo_TON<test_case_name>\Results --timestamp --reportbackground white:white:white`
11. Under Run tab, click **Start** button to start the execution.



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



13. 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

LOG

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	<div style="width: 100%;"></div>
All Tests	1	1	0	00:00:55	<div style="width: 100%;"></div>

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	<div style="width: 100%;"></div>

Test Details

Type:
☐ Critical Tests
☐ All Tests

14. 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

REPORT

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:00:55	<div style="width: 100%;"></div>
All Tests	1	1	0	00:00:55	<div style="width: 100%;"></div>

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	<div style="width: 100%;"></div>

Test Execution Log

SUITE

TestCases

00:00:55.548

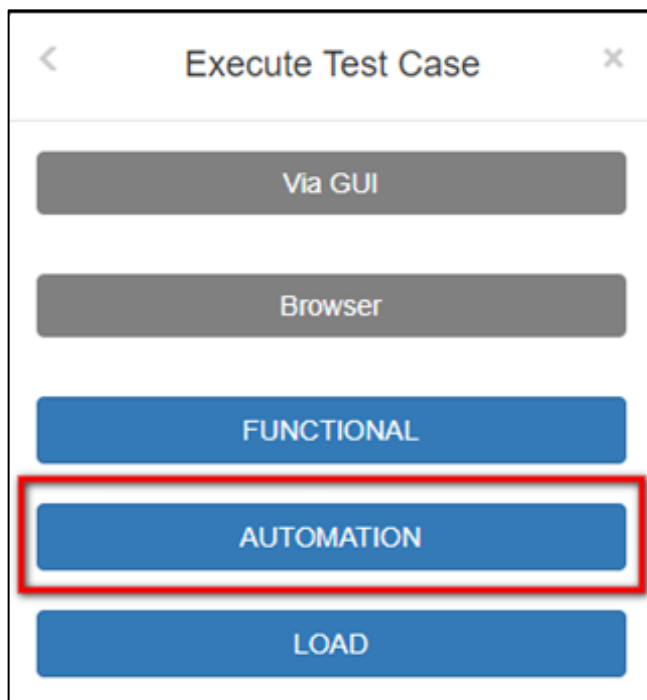
Full Name: TestCases
Source: E:\TON\WebTesting\Browser\GUI\Demo_TON\TestCases.robot
Start / End / Elapsed: 20180112 15:19:20.224 / 20180112 15:20:15.772 / 00:00:55.548
Status: 1 critical test, 1 passed, 0 failed
 1 test total, 1 passed, 0 failed

TEST

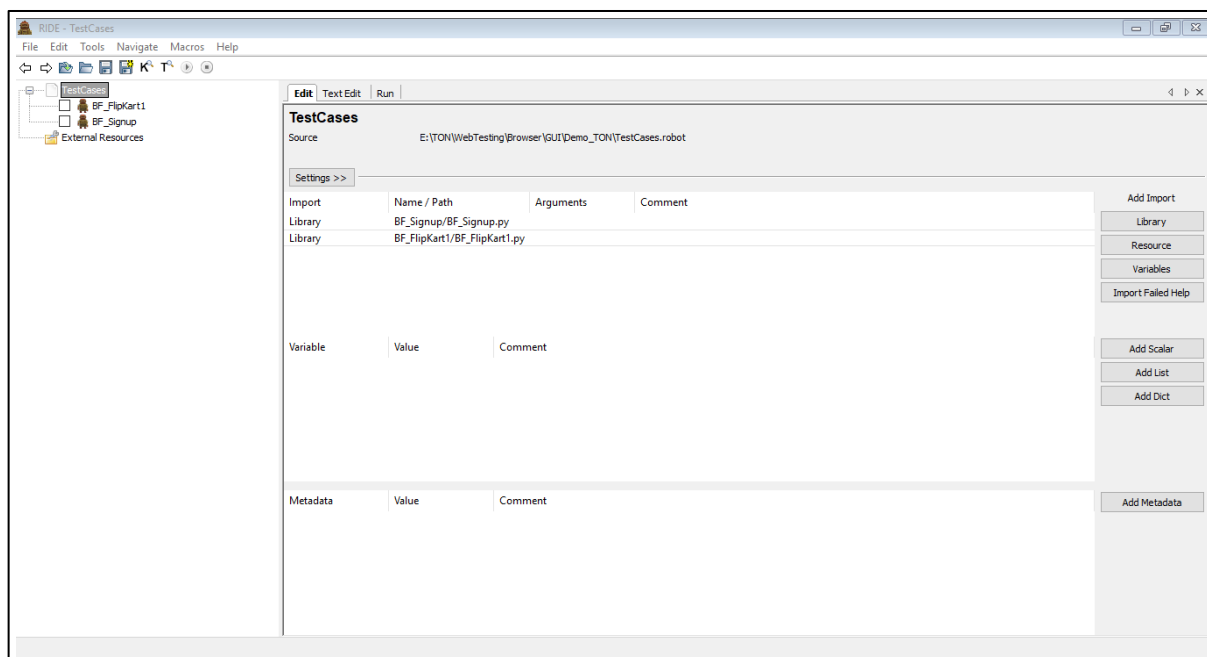
BF_Signup

00:00:55.131

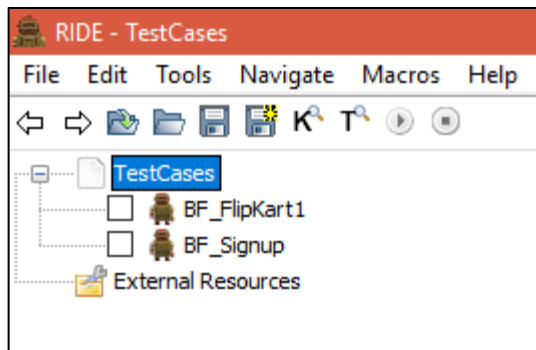
15. To perform automation testing of the JSON file downloaded using Kantu browser, click **Automation** button.



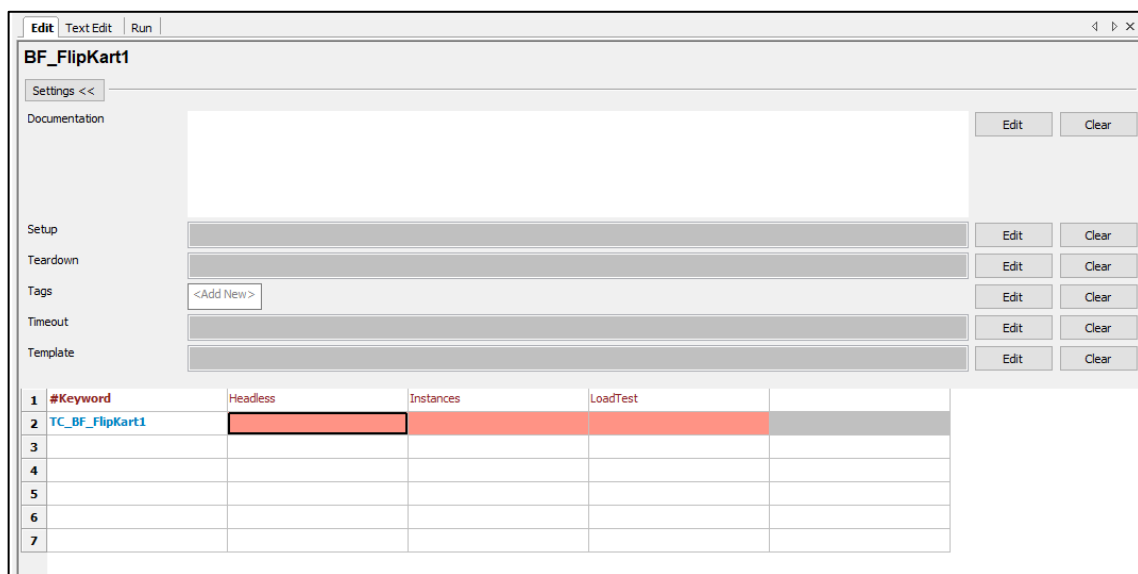
16. On click of the button, **RIDE** will open.



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



18. To configure the parameters, click on the test case name.
19. On click of the test case name, Edit tab changes so that parameters can be configured.



20. The value of the parameters should be:

Headless = No / Yes, Instances = 0, LoadTest = No

For Headless = Yes, browser simulation takes place without GUI.

For Headless = No, browser simulation takes place with the visible browser.

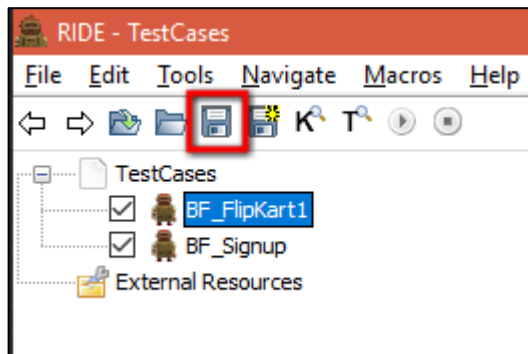
Instances = 0 (Only valid for Load Test)

LoadTest = No (Only valid for Load Test)

21. Configure the parameters – **Headless, Instances and LoadTest.**

1	#Keyword	Headless	Instances	LoadTest	
2	TC_BF_FlipKart1	Yes	0	No	
3					
4					
5					
6					
7					

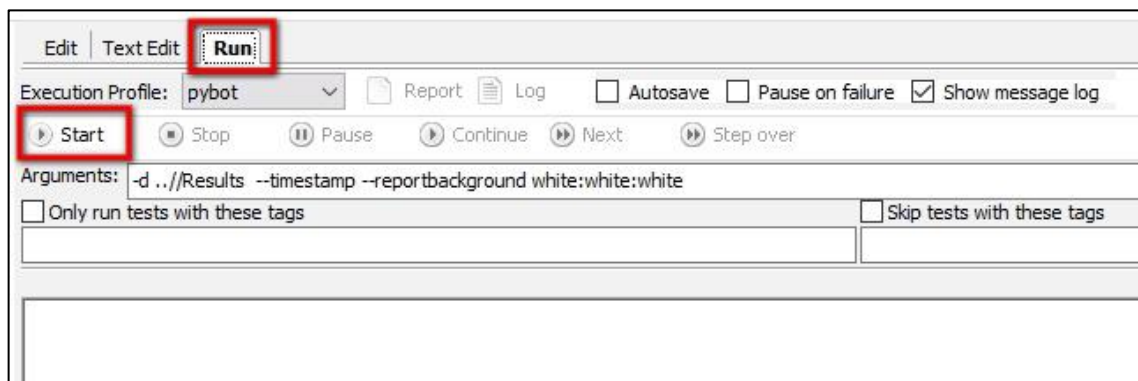
22. Click **Save** button to save the test case.



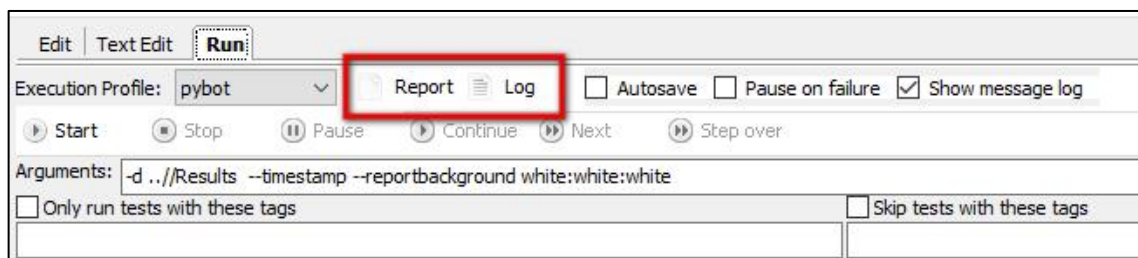
23. Configure the **Arguments** textbox by entering:

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

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



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



26. 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

LOG

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	<div style="width: 100%;"></div>
All Tests	1	1	0	00:00:55	<div style="width: 100%;"></div>

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	<div style="width: 100%;"></div>

Test Details

Type:
☐ Critical Tests
☐ All Tests

27. 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

REPORT

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:00:55	<div style="width: 100%;"></div>
All Tests	1	1	0	00:00:55	<div style="width: 100%;"></div>

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	<div style="width: 100%;"></div>

Test Execution Log

SUITE

TestCases

00:00:55.548

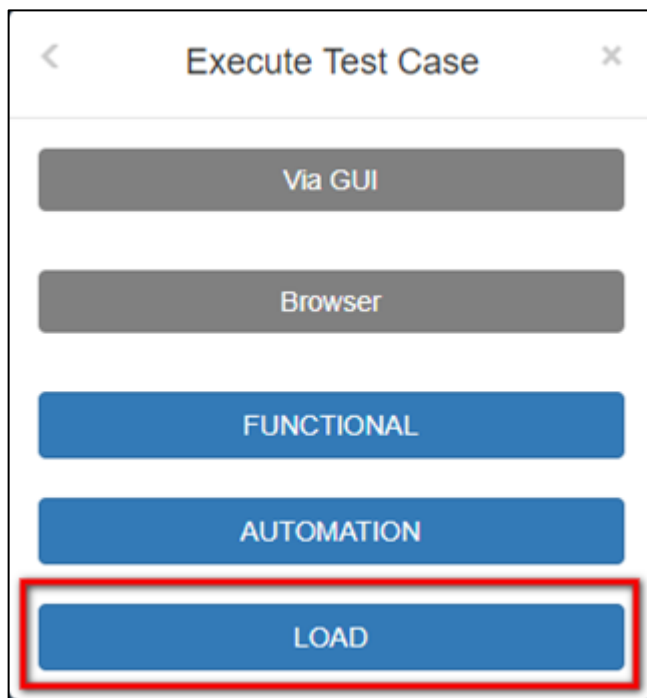
Full Name: TestCases
Source: E:\TON\WebTesting\Browser\GUI\Demo_TON\TestCases.robot
Start / End / Elapsed: 20180112 15:19:20.224 / 20180112 15:20:15.772 / 00:00:55.548
Status: 1 critical test, 1 passed, 0 failed
 1 test total, 1 passed, 0 failed

TEST

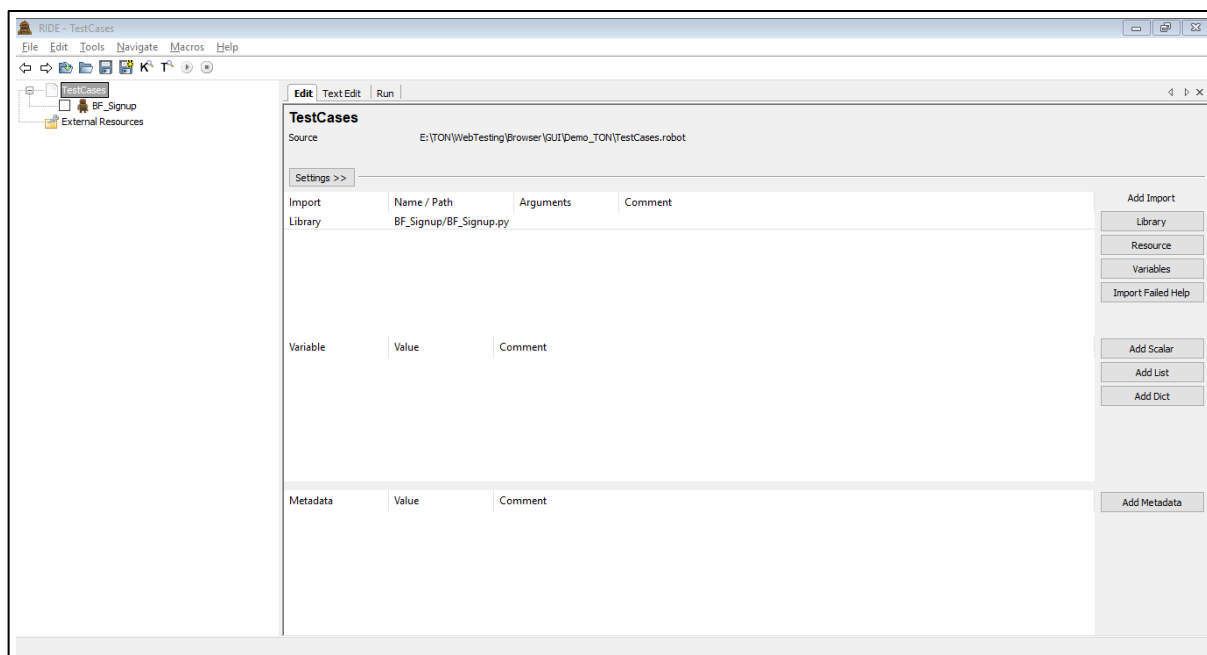
BF_Signup

00:00:55.131

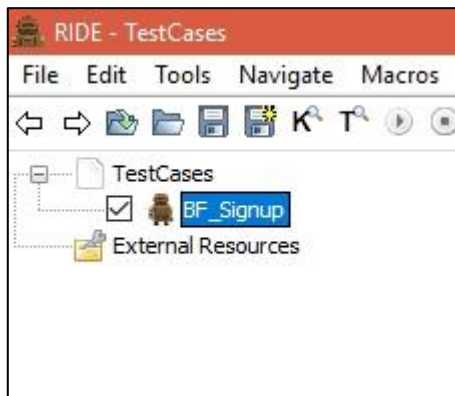
29. To perform load testing of the JSON file downloaded using Kantu browser, click **Load** button.



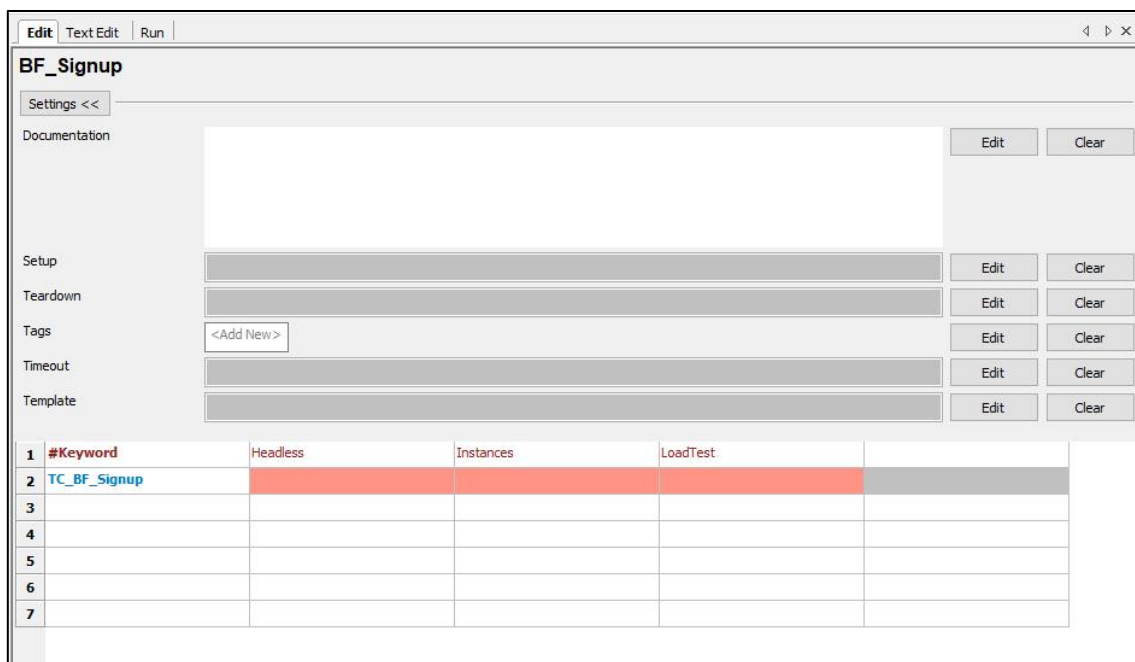
30. On click of the button, **RIDE** will open.



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



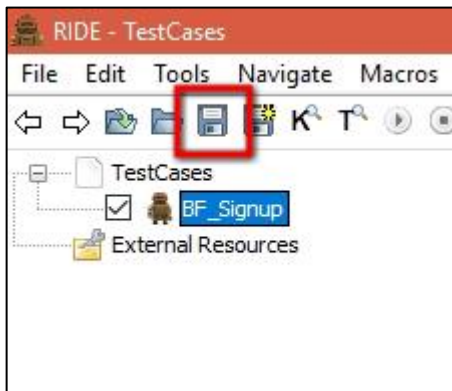
32. To configure the parameters, click on the test case name.
33. On click of the test case name, Edit tab changes so that parameters can be configured.



34. The value of the parameters should be:
- Headless = No / Yes, Instances = <user defined>, LoadTest = Yes.
- For Headless = Yes, browser simulation takes place without GUI.
- For Headless = No, browser simulation takes place with visible browser.
- Instances = 0 (Only valid for Load Test)
- LoadTest = Yes (Only valid for Load Test)
35. Configure the parameters – **Headless, Instances and LoadTest.**

1	#Keyword	Headless	Instances	LoadTest	
2	TC_BF_Signup	Yes	5	Yes	
3					
4					
5					
6					
7					

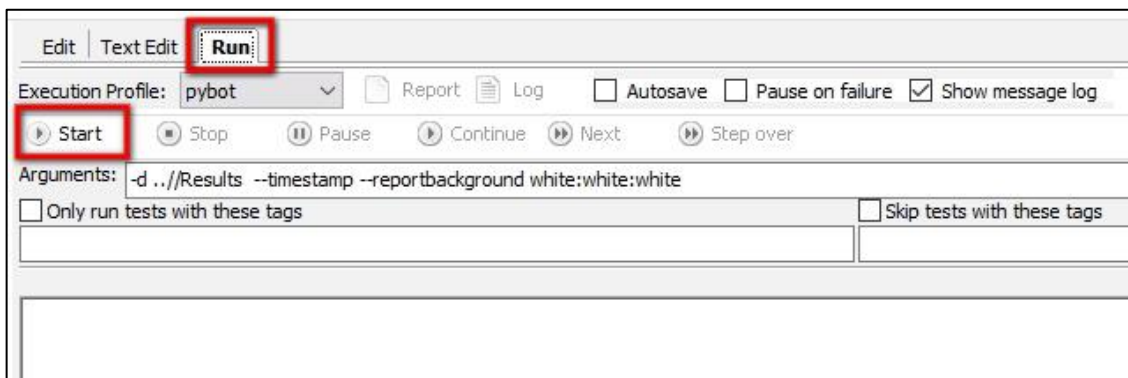
36. Click **Save** button to save the test case.



37. Configure the **Arguments** textbox by entering:

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

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



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



40. 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

LOG

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	<div style="width: 100%; height: 10px; background-color: green;"></div>
All Tests	1	1	0	00:00:55	<div style="width: 100%; height: 10px; background-color: green;"></div>

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	<div style="width: 100%; height: 10px; background-color: green;"></div>

Test Details

Totals

Tags

Suites

Search

Type:

☒ Critical Tests
 ☐ All Tests

41. 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

REPORT

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:00:55	<div style="width: 100%; height: 10px; background-color: green;"></div>
All Tests	1	1	0	00:00:55	<div style="width: 100%; height: 10px; background-color: green;"></div>

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	<div style="width: 100%; height: 10px; background-color: green;"></div>

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

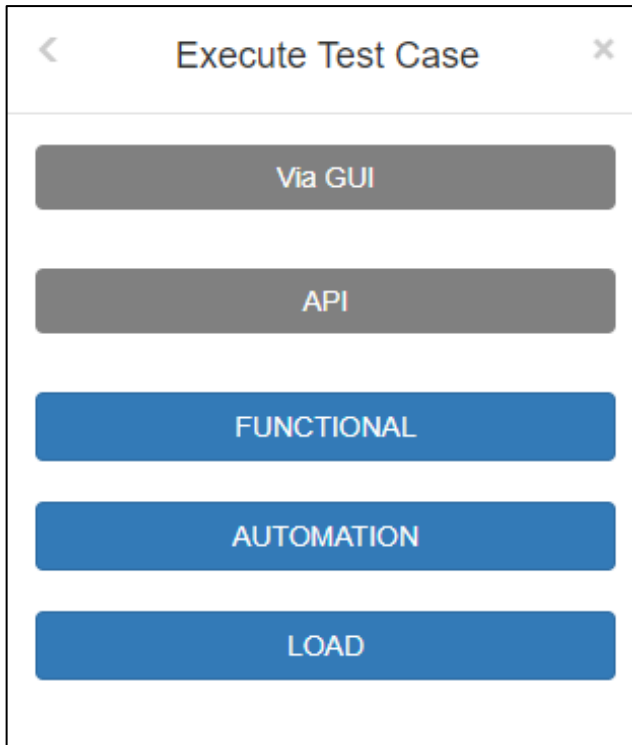
TEST

BF_Signup

00:00:55.131

2.3.1.2 EXECUTE – VIA GUI – API

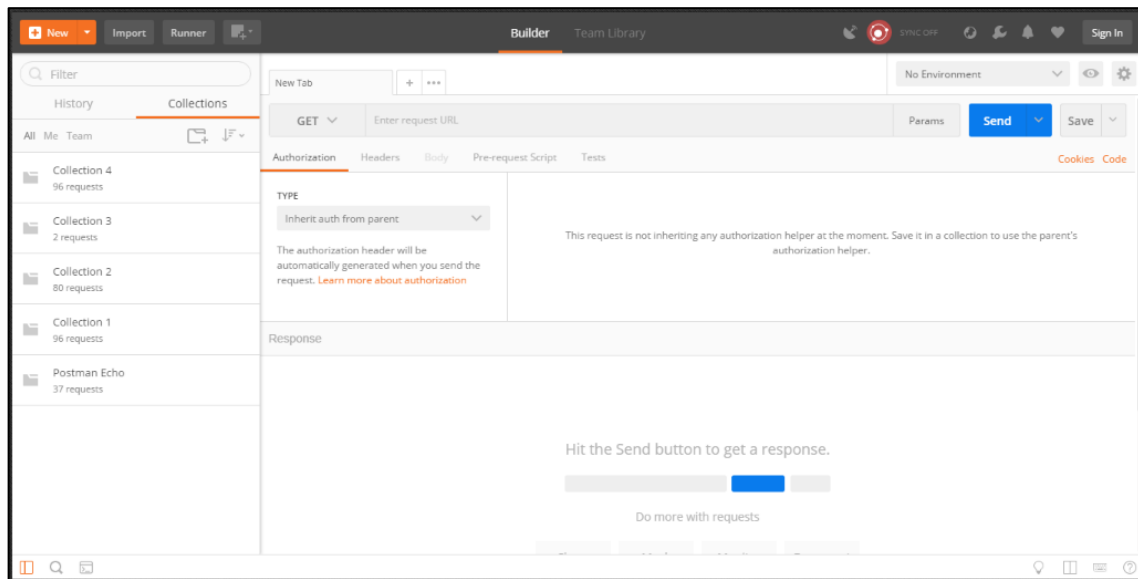
1. On click of the API button, 3 buttons will appear – **Functional**, **Automation** and **Load**.



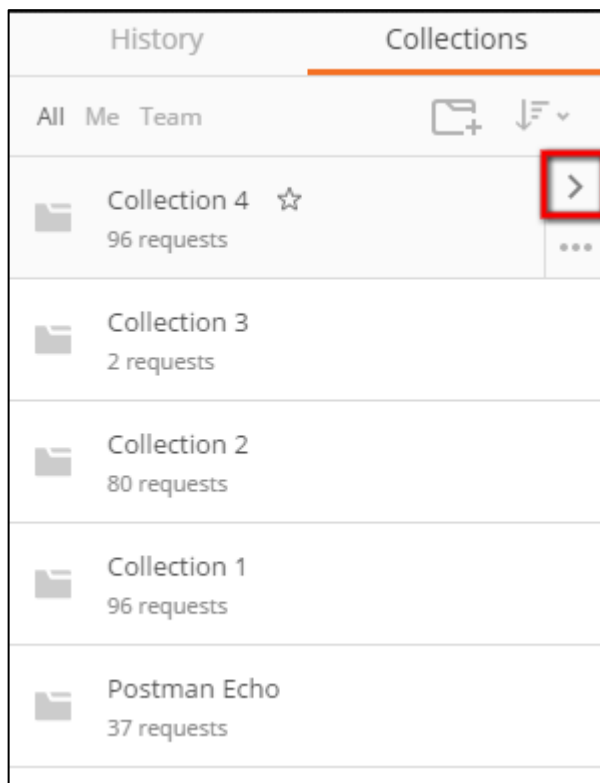
2. To perform functional testing of the recorded scenario ~~using the Postman application~~, press **Functional** button.



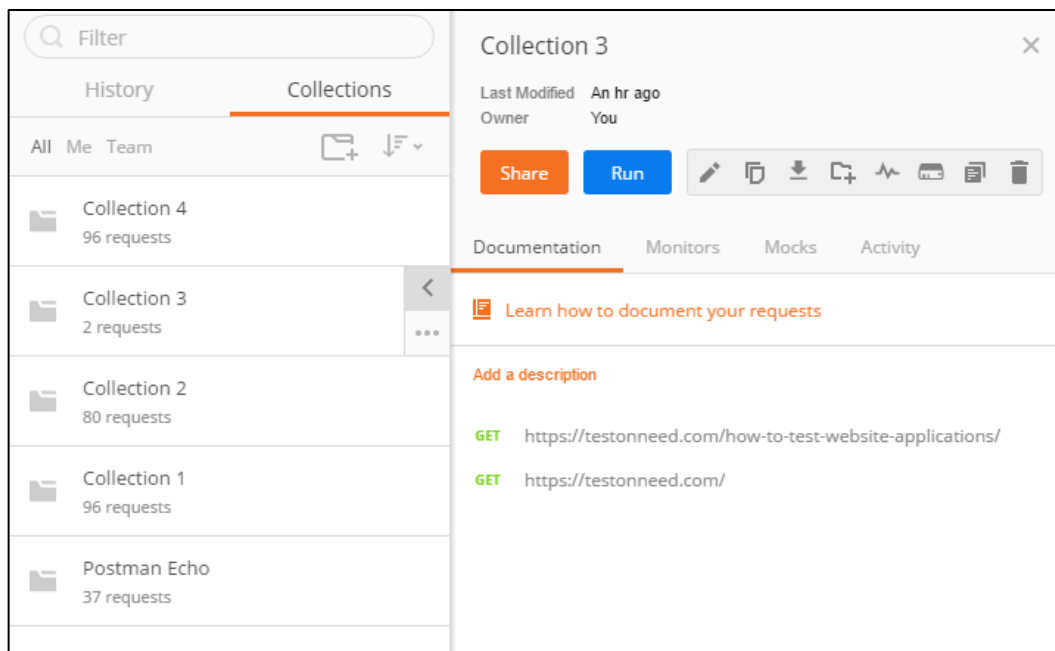
3. On click of the button, Postman will open.



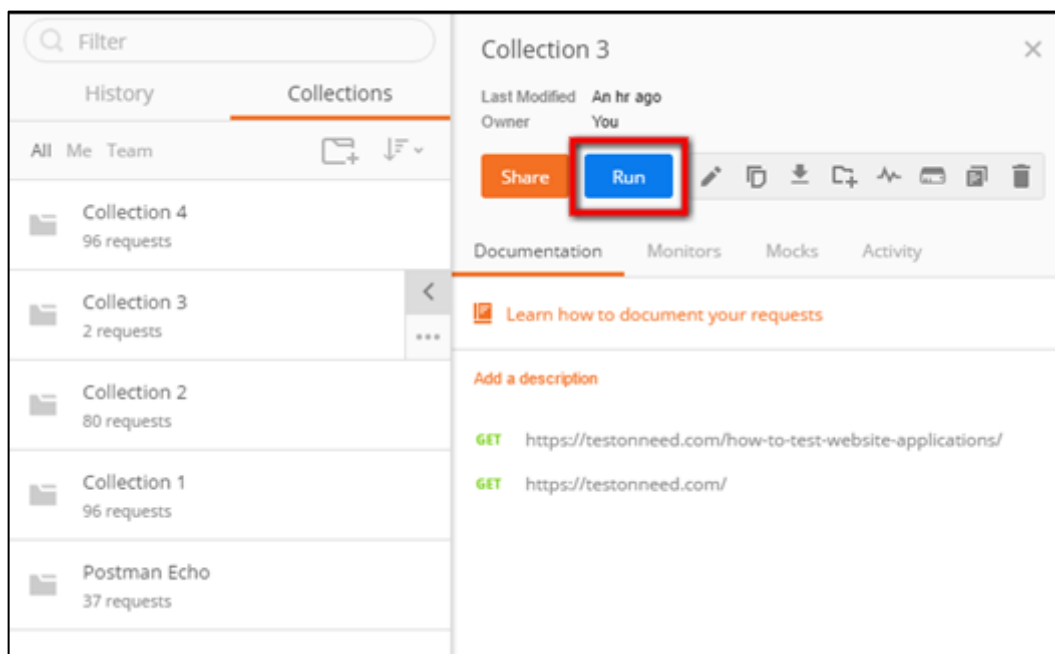
4. On the left, there will be two tabs History and Collections. In the collections tab look for the **Collection** in which the JSON was saved during the **Record Test Case** button.
5. On mouse hover of the **Collection name**, click on the right arrow.



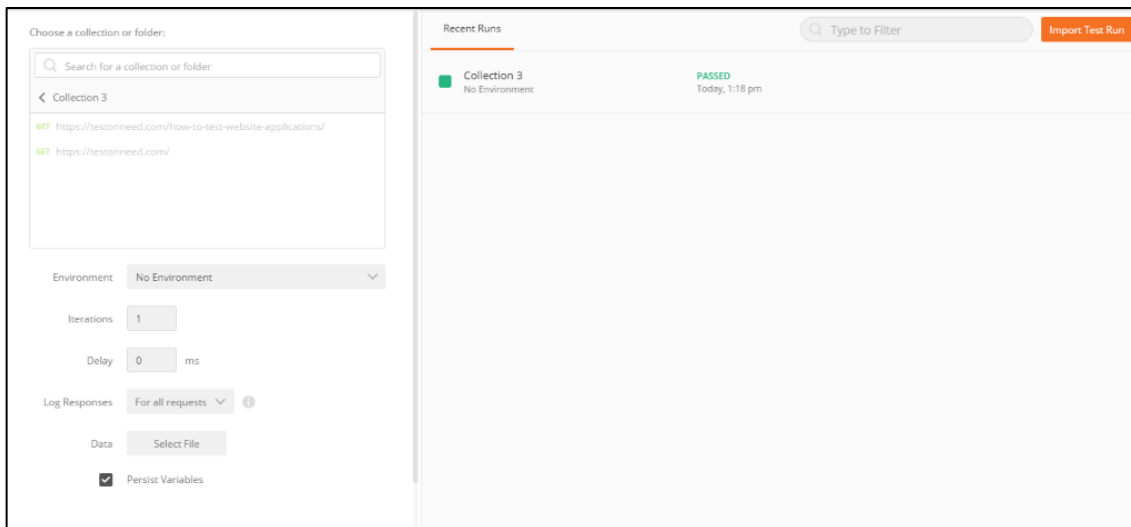
6. On clicking of the arrow button, a small window comes on the right next to the collection selected.



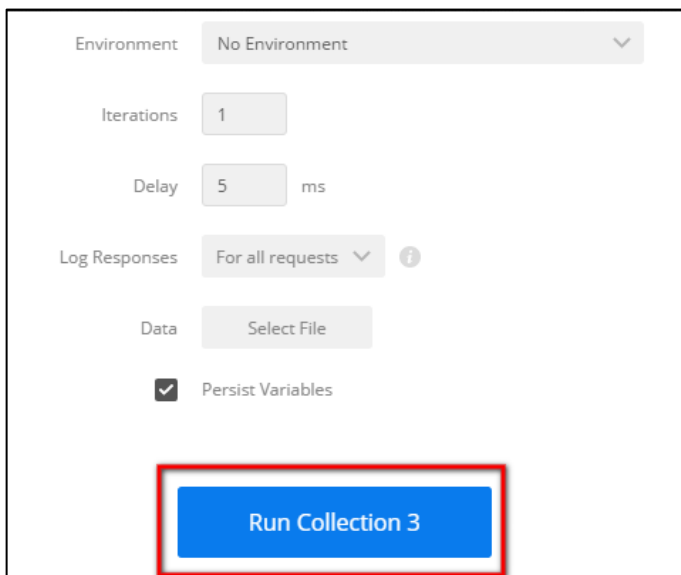
7. From the pop-up, click **Run** button.



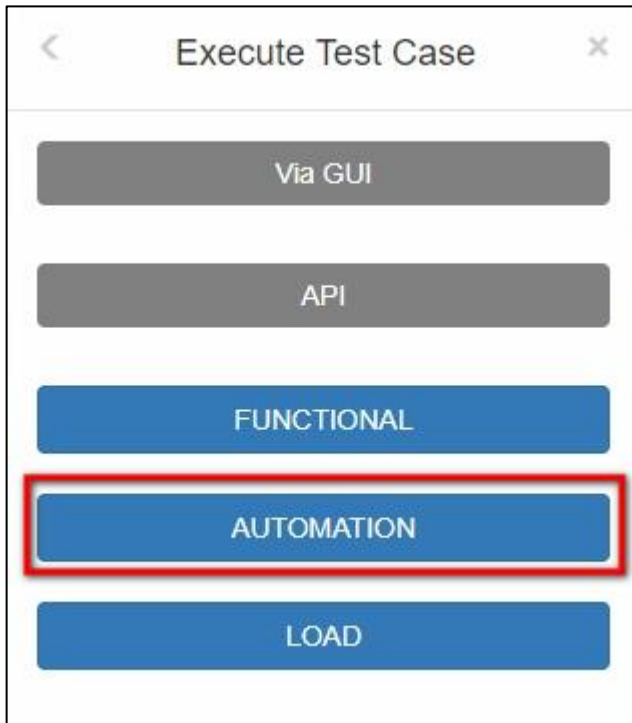
8. On click of **Run** button, a new window will open called **Collection Runner**.



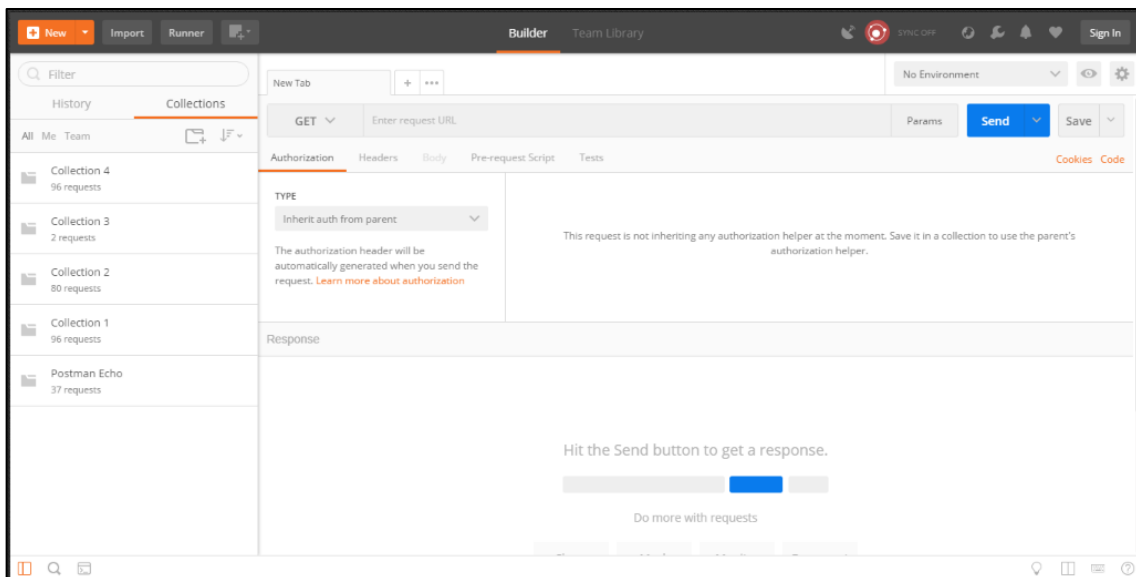
9. In the window, on the left panel, set the parameters and click **Run <Collection_Name>**.



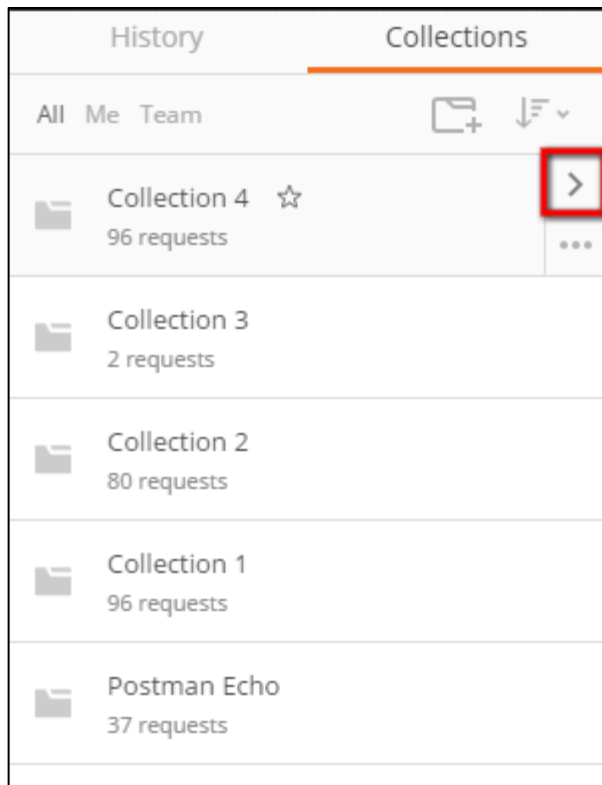
10. To perform automation testing of the JSON downloaded using the postman, press **Automation** button.



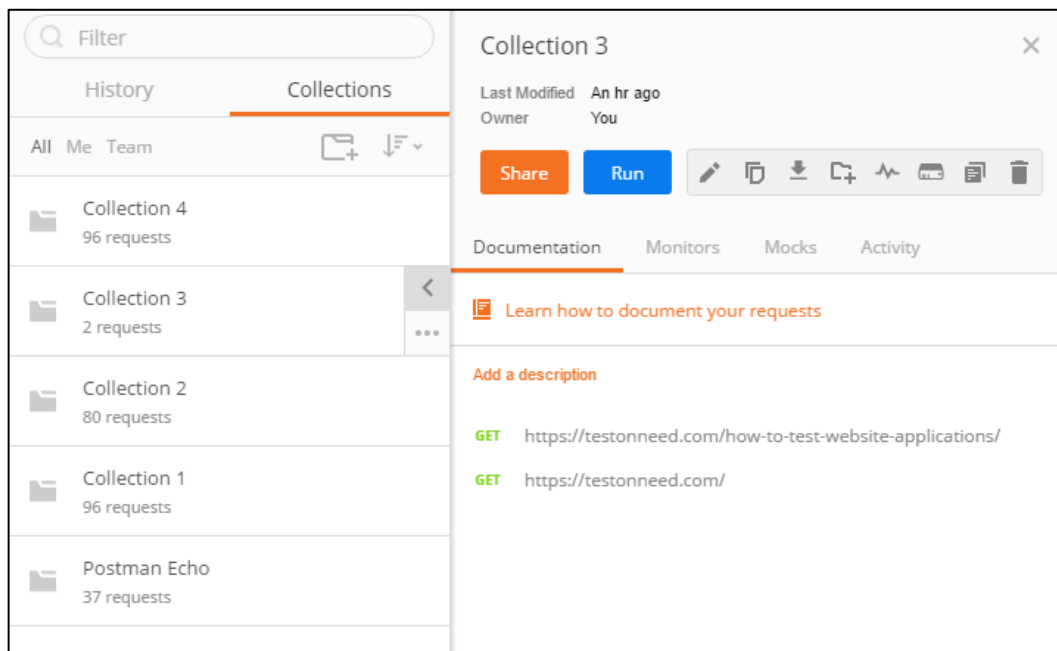
11. On click of the button, Postman will open.



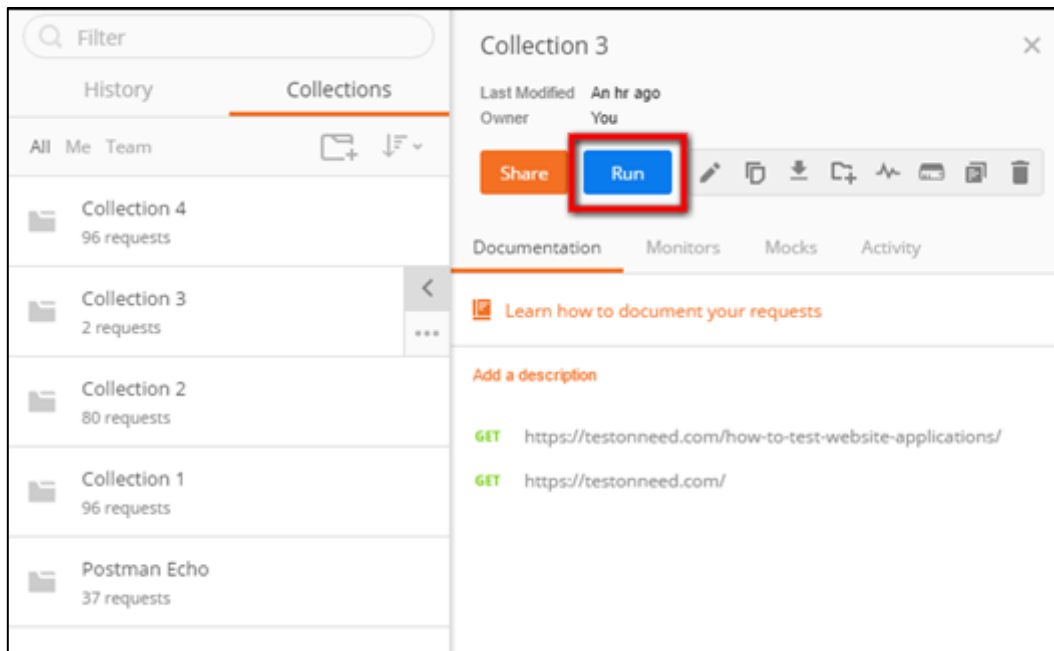
12. On the left, there will be two tabs History and Collections. In the collections tab look for the **Collection** in which the JSON was saved during the **Record Test Case** button.
13. On mouse hover of the **Collection name**, click on the right arrow.



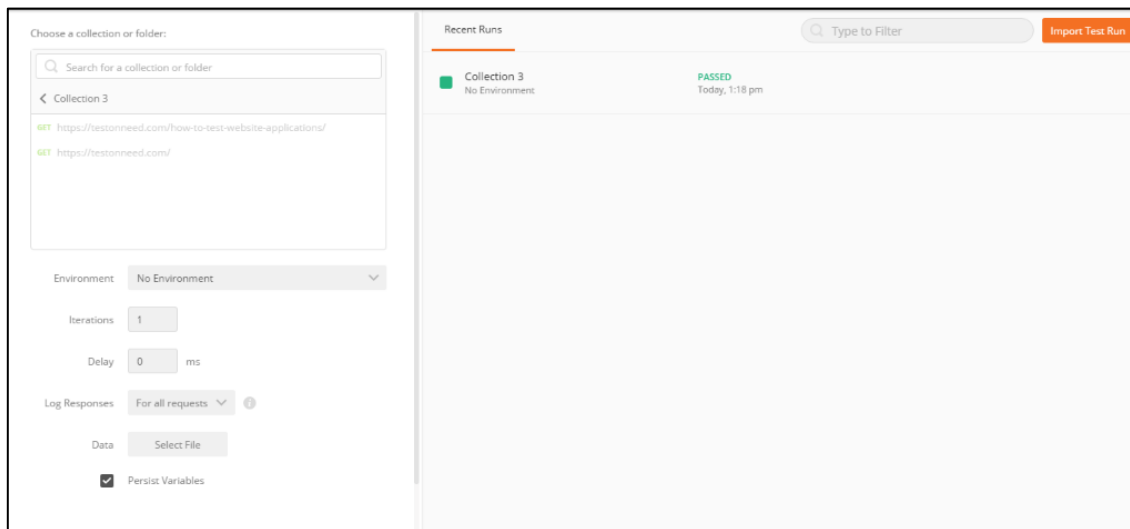
14. On clicking of the arrow button, a small window comes on the right next to the collection selected.



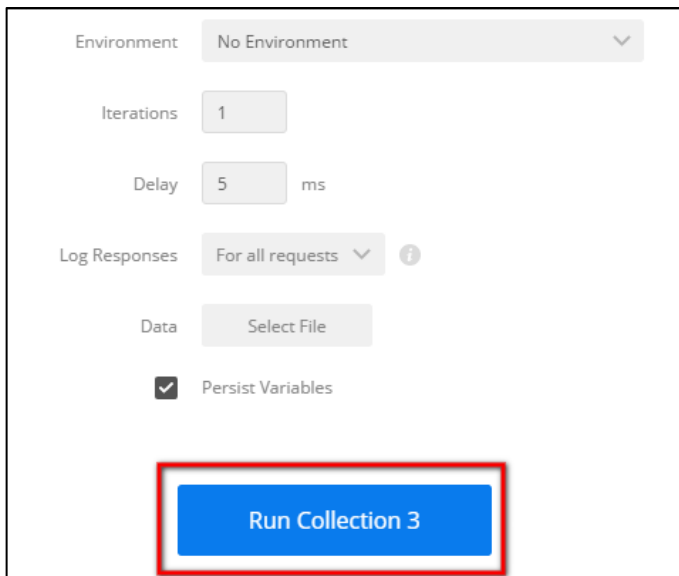
15. From the pop-up, click **Run** button.



16. On click of **Run** button, a new window will open called **Collection Runner**.



17. In the window, on the left panel, set the parameters and click **Run <Collection_Name>**.



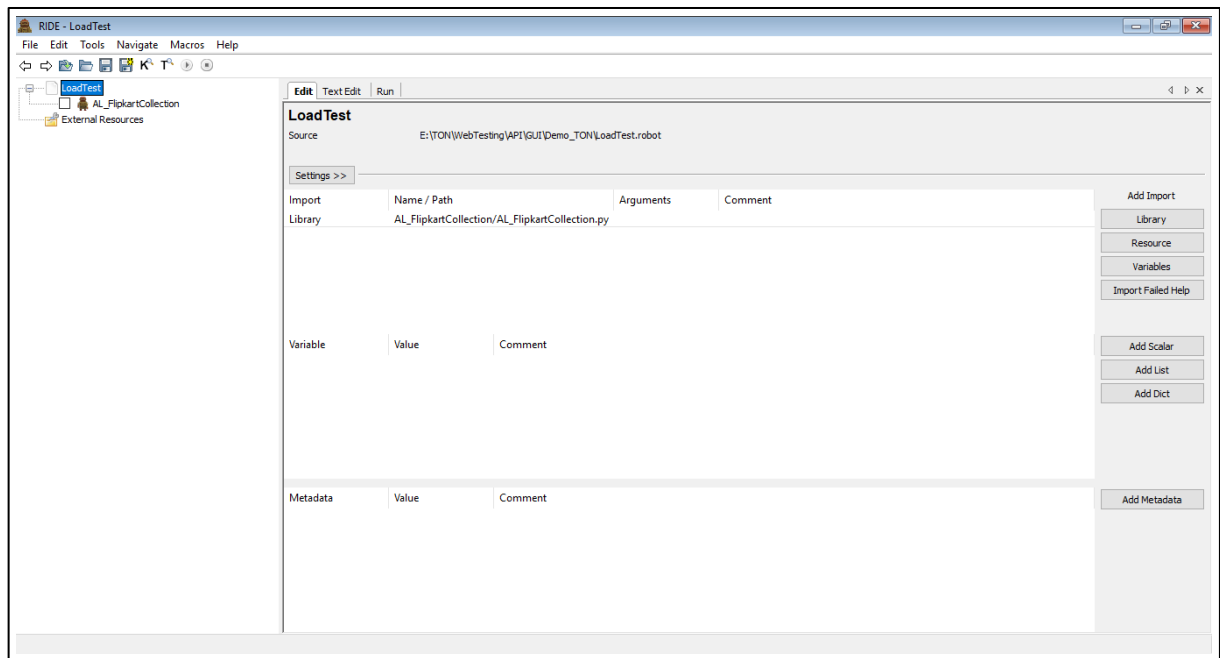
The screenshot shows a configuration panel for running a collection. It includes fields for 'Environment' (set to 'No Environment'), 'Iterations' (set to '1'), 'Delay' (set to '5 ms'), 'Log Responses' (set to 'For all requests'), and a 'Data' section with a 'Select File' button. A checkbox for 'Persist Variables' is checked. The 'Run Collection 3' button is highlighted with a red rectangle.

18. To perform load testing of the JSON downloaded using the postman, press **Load** button.

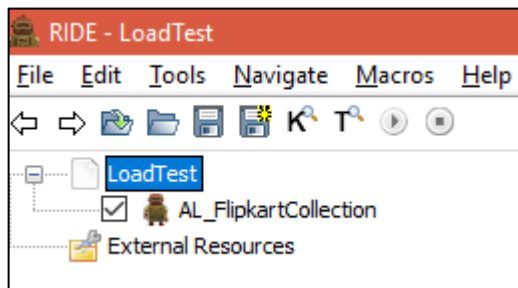


The screenshot shows a dialog box titled 'Execute Test Case'. It contains five buttons: 'Via GUI', 'API', 'FUNCTIONAL', 'AUTOMATION', and 'LOAD'. The 'LOAD' button is highlighted with a red rectangle.

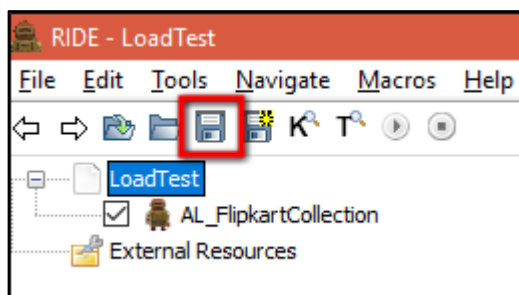
19. On click of the button, RIDE will open.



20. On the left panel, select one test case to executed.



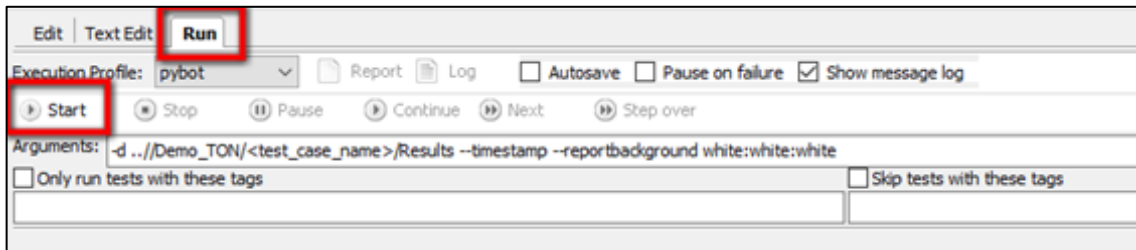
21. To configure the parameter, click on the test case name.
22. On click of the test case name, edit tab changes so that parameters can be configured.
23. The value of the parameters is:
Iterations, VirtualUsers and RampUP Period.
24. Configure the parameters – **Iterations**, **VirtualUsers** and **RampUP Period**.
25. Click **Save** button to save the test case.



26. Configure the **Arguments** textbox by entering:

```
-d ../Demo_TON/<test_case_name>/Results --timestamp --reportbackground
white:white:white
```

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



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



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

TestCases Test Report

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](#)

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

LOG

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:00:55	<div style="width: 100%; height: 10px; background-color: green;"></div>
All Tests	1	1	0	00:00:55	<div style="width: 100%; height: 10px; background-color: green;"></div>

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	<div style="width: 100%; height: 10px; background-color: green;"></div>

Test Details

Totals Tags Suites Search

Type: ☐ Critical Tests ☐ All Tests

30. 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

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:00:55	<div style="width: 100%; height: 5px; background-color: green;"></div>
All Tests	1	1	0	00:00:55	<div style="width: 100%; height: 5px; background-color: green;"></div>

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	<div style="width: 100%; height: 5px; background-color: green;"></div>

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

REPORT

Test Execution Log

SUITE TestCases
00:00:55.548

Full Name: TestCases

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

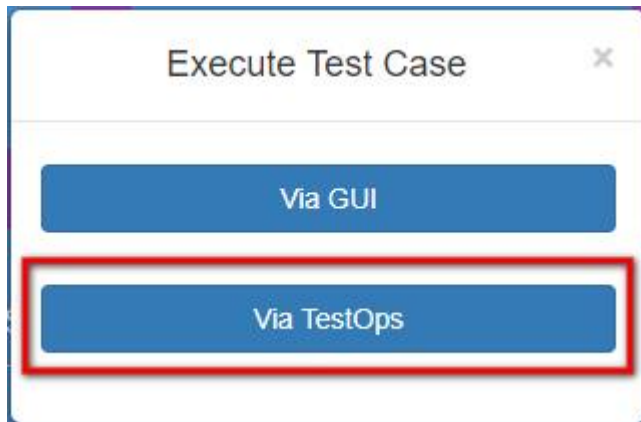
Start / End / Elapsed: 20180112 15:19:20.224 / 20180112 15:20:15.772 / 00:00:55.548

Status: 1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed

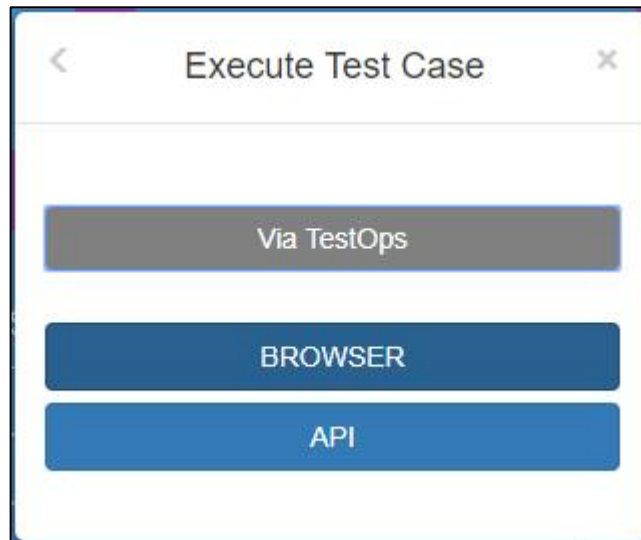
TEST BF_Signup
00:00:55.131

2.3.2 EXECUTE - VIA TESTOPS

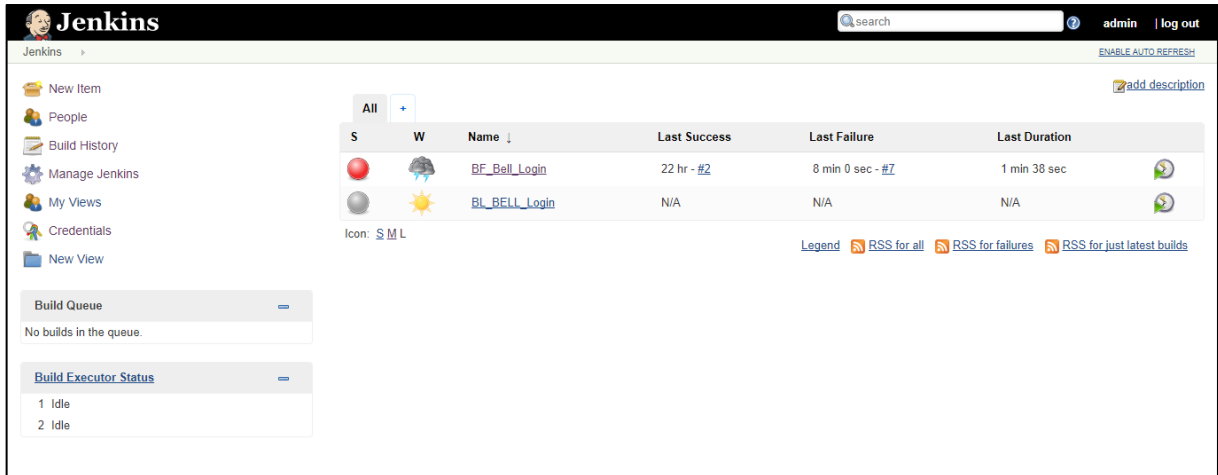
1. Click the Via TestOps button.



2. On clicking the button, 2 buttons will appear – **Browser** and **API**.



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



The screenshot shows the Jenkins dashboard. On the left is a sidebar with navigation links: New Item, People, Build History, Manage Jenkins, My Views, Credentials, and New View. Below these are sections for 'Build Queue' (showing 'No builds in the queue') and 'Build Executor Status' (showing '1 Idle' and '2 Idle'). The main area displays a table of jobs. The first job, 'BF_Bell_Login', is highlighted with a red circle. Below the table are links for 'Icon: S M L' and 'Legend' with RSS feeds for all, failures, and latest builds.

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

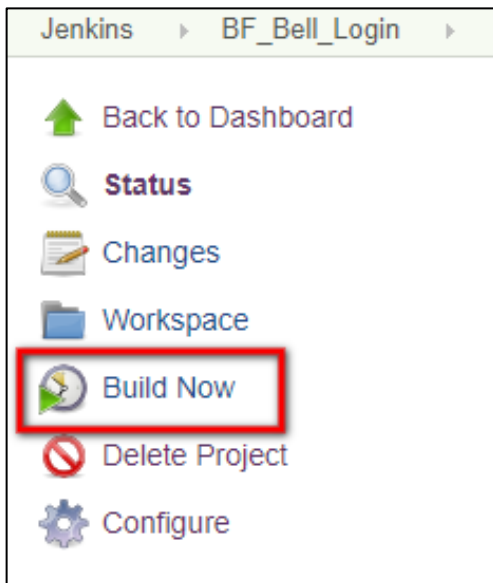
For creation of new [View](#) and [Job](#), refer the Reference Section below.

- Click on the job that has to be executed.



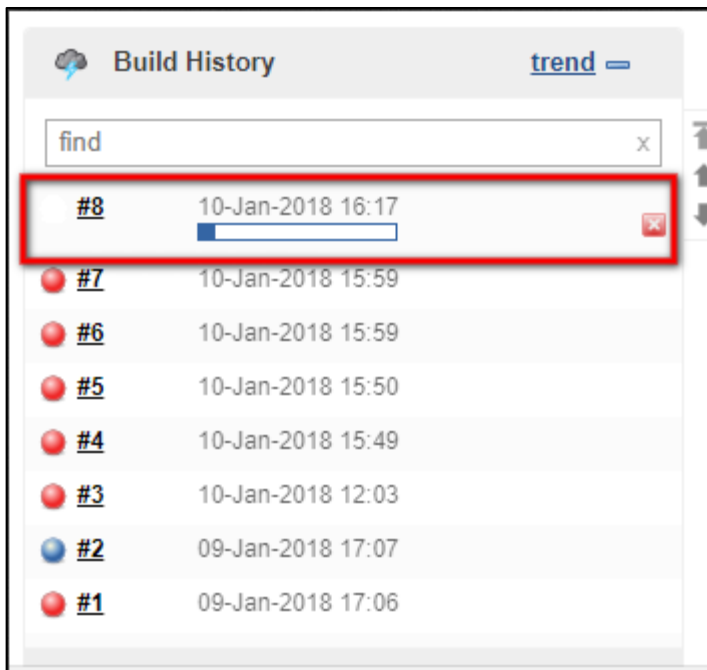
This screenshot is a zoomed-in view of the job list from the previous image. The first job, 'BF_Bell_Login', is highlighted with a red rectangle. The table structure and data are identical to the previous image.

- On the left side, select Build Now.

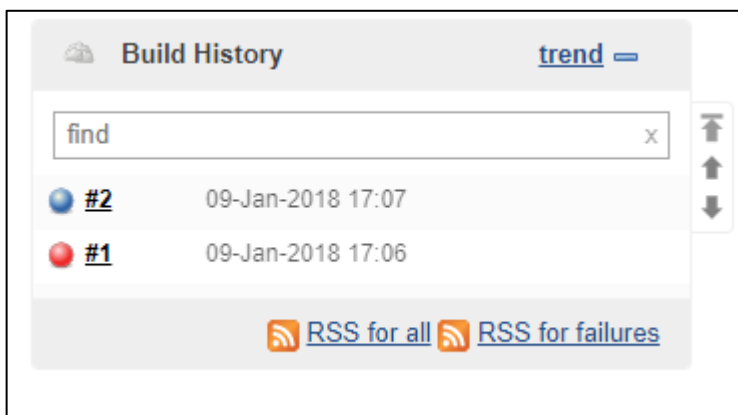


The screenshot shows the action menu for the 'BF_Bell_Login' job. The menu items are: Back to Dashboard, Status, Changes, Workspace, Build Now (highlighted with a red rectangle), Delete Project, and Configure.

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



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

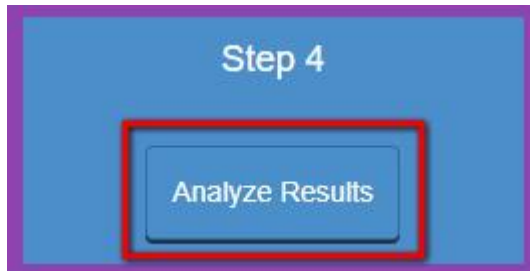


8. All the jobs of the Jenkins will be saved in:
- > TON > WebTesting > Browser > TestOps > Demo_TON > <Job_Name_Folder>**

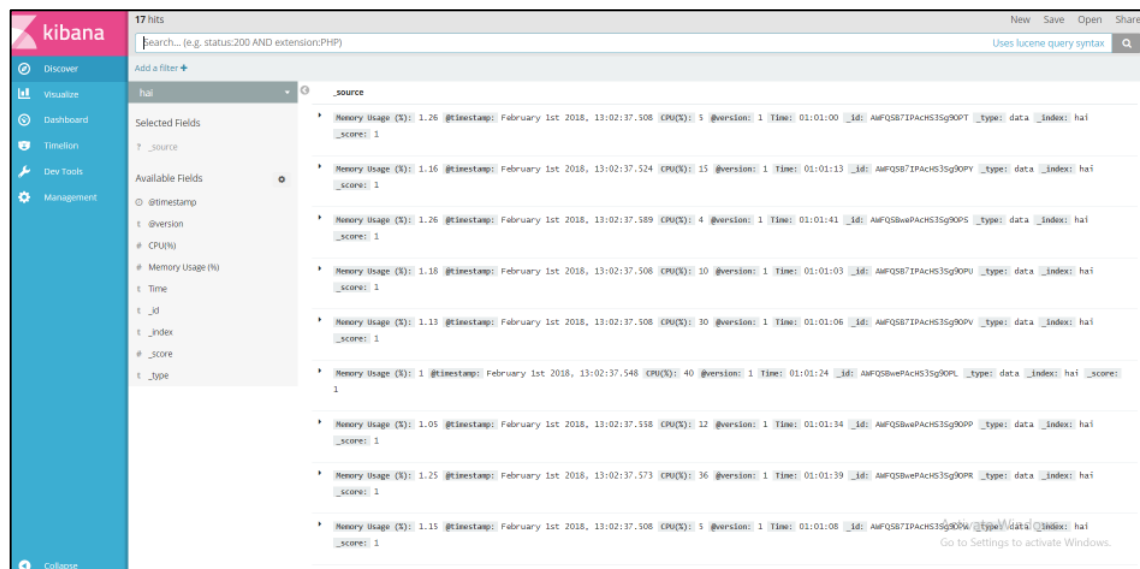
2.4 ANALYZE RESULTS

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

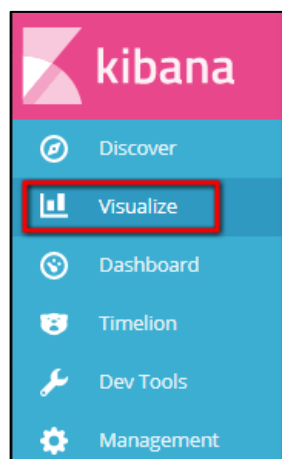
1. Click on the **Analyze Results**.



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



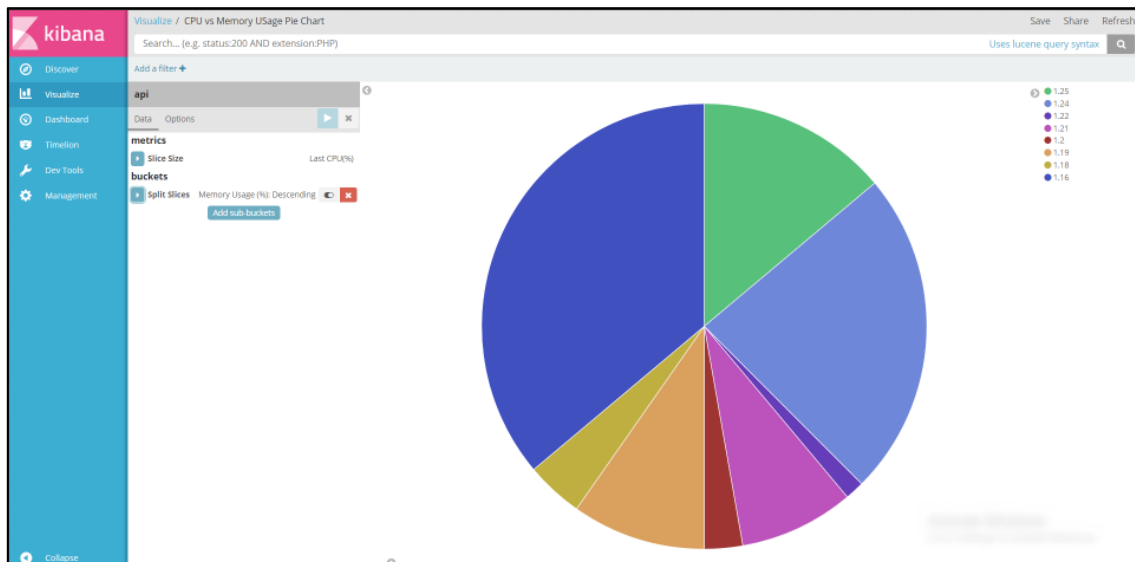
3. On the left panel, click Visualize.



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

<input type="text" value="Search..."/> + 1-5 of 5	
<input type="checkbox"/> Name ▲	Type
<input type="checkbox"/> CPU vs Memory USage Pie Chart	Pie
<input type="checkbox"/> CPU vs Memory USage vs Time Pie Chart	Pie
<input type="checkbox"/> CPU vs Memory Usage bar chart	Vertical Bar
<input type="checkbox"/> CPU vs Time Pie Chart	Pie
<input type="checkbox"/> CPU vs Time bar chart	Vertical Bar
1-5 of 5	

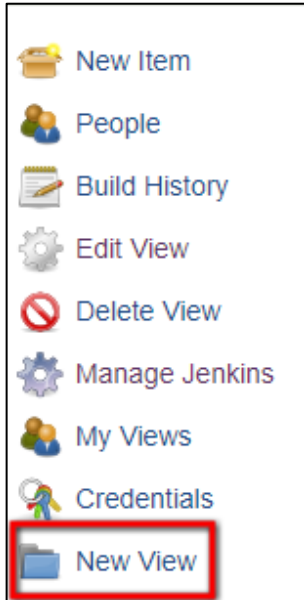
5. On click of the chart name, the user can see the graph based on the type of the chart.
(The example shows for Pie Chart)



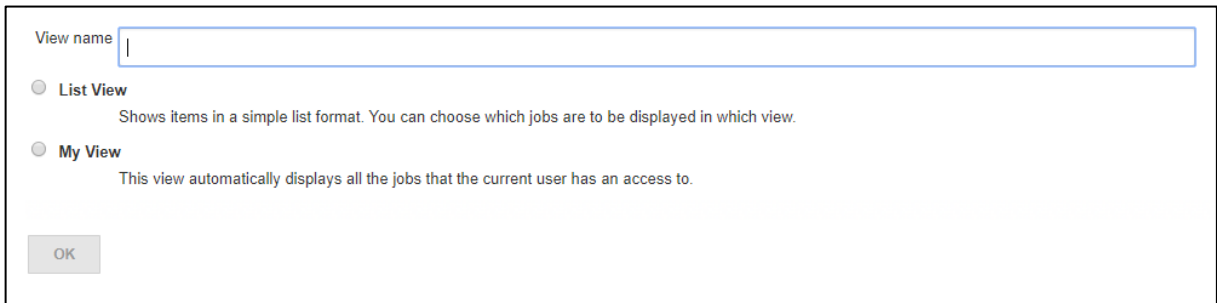
3. REFERENCES

3.1 CREATION OF NEW VIEW

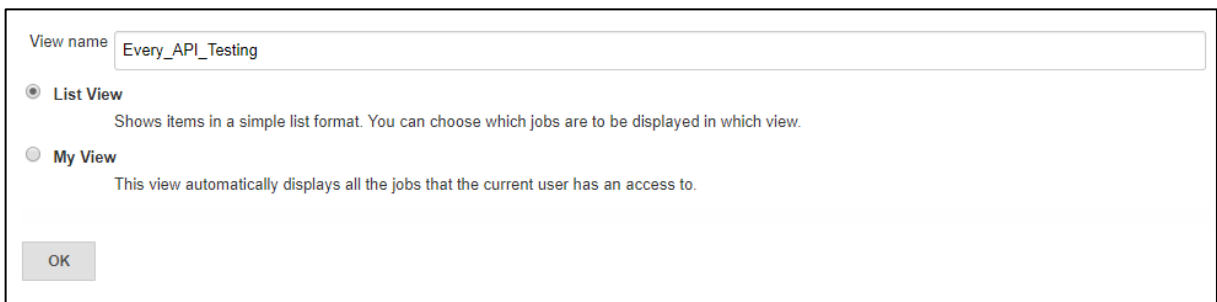
- To create a new view, on the left panel, click **New View**.



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

A screenshot of the 'New View' dialog box. It has a text input field for 'View name' which is currently empty. Below it are two radio button options: 'List View' and 'My View'. The 'List View' option is selected. Below the radio buttons are two lines of descriptive text: 'Shows items in a simple list format. You can choose which jobs are to be displayed in which view.' for List View, and 'This view automatically displays all the jobs that the current user has an access to.' for My View. At the bottom left is an 'OK' button.

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

A screenshot of the 'New View' dialog box. The 'View name' text input field now contains the text 'Every_API_Testing'. The 'List View' radio button remains selected. The 'OK' button at the bottom left is now active (no longer disabled).

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

Name

Description

[Plain text] [Preview](#)

Filter build queue ☐

Filter build executors ☐

Job Filters

Status Filter

Recurse in subfolders ☐

Jobs

- ☒ AL_FlipKartCollection
- ☐ BF_Bell_Login
- ☐ BL_BELL_Login

☐ Use a regular expression to include jobs into the view

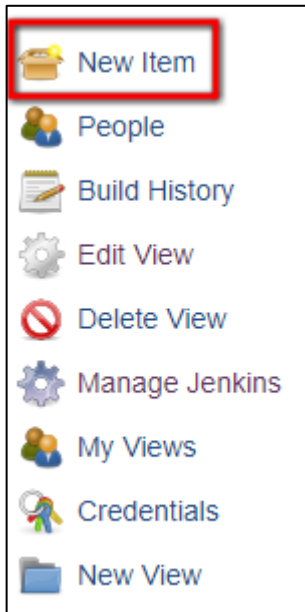
Add Job Filter ▼

Columns

- 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

BL_BELL_Login


» 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.

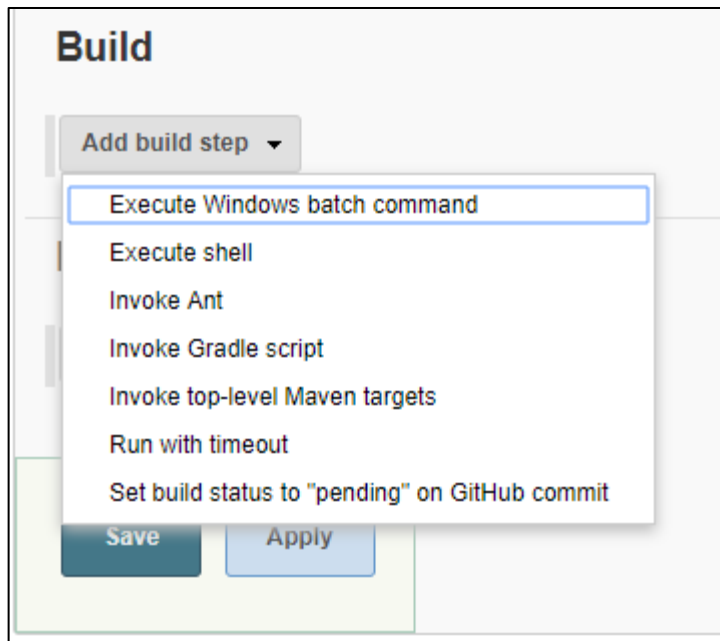
OK

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

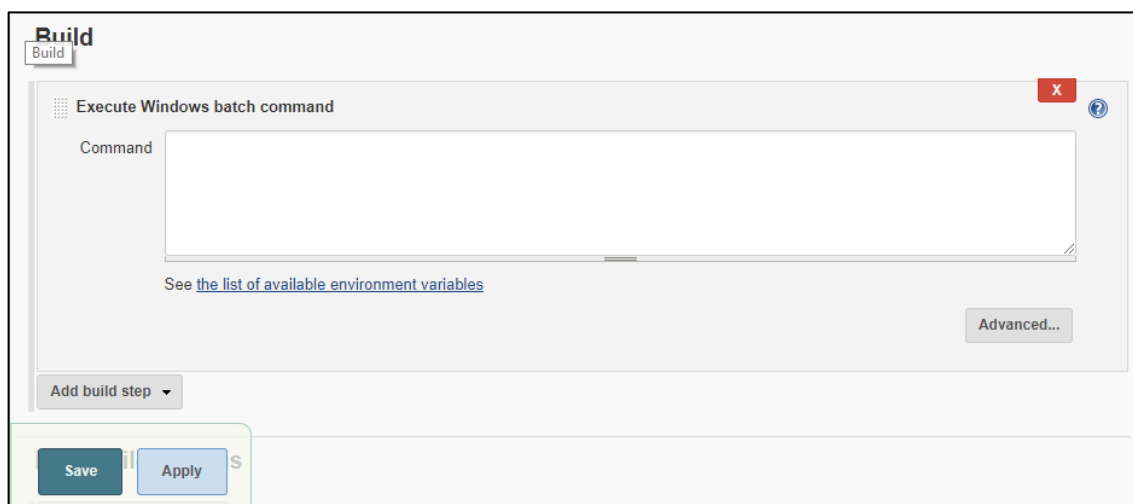
Build

Add build step ▼

- Under build, click Add Build Setup drop down.

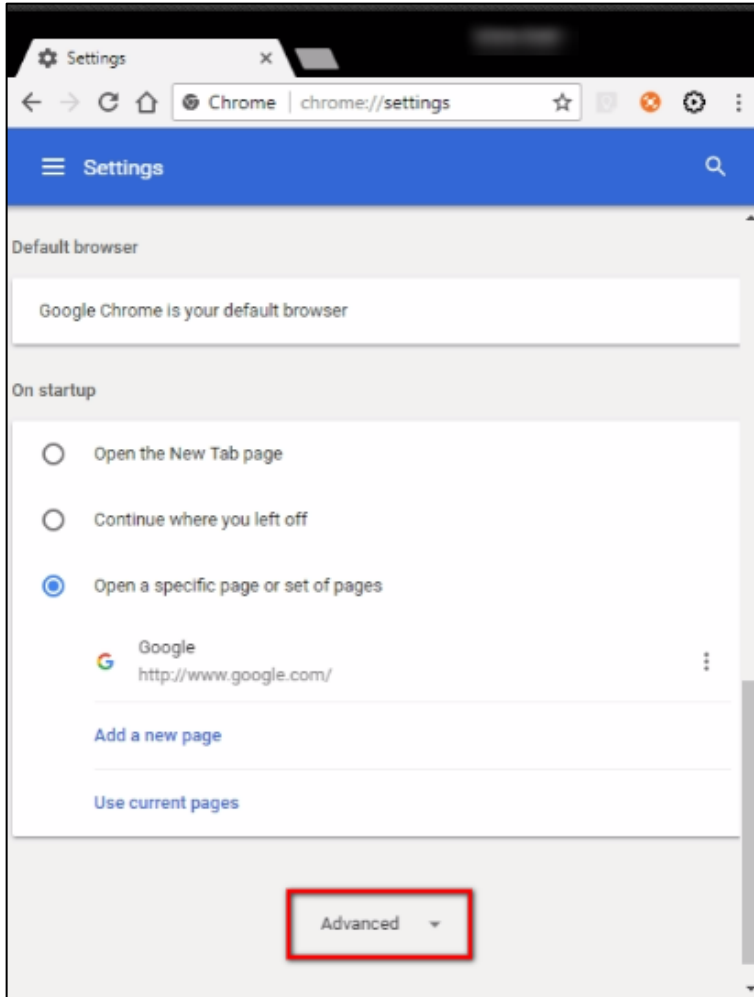


- From the drop down, select **Execute Windows Batch Command**.
- On selecting, a text area appears.

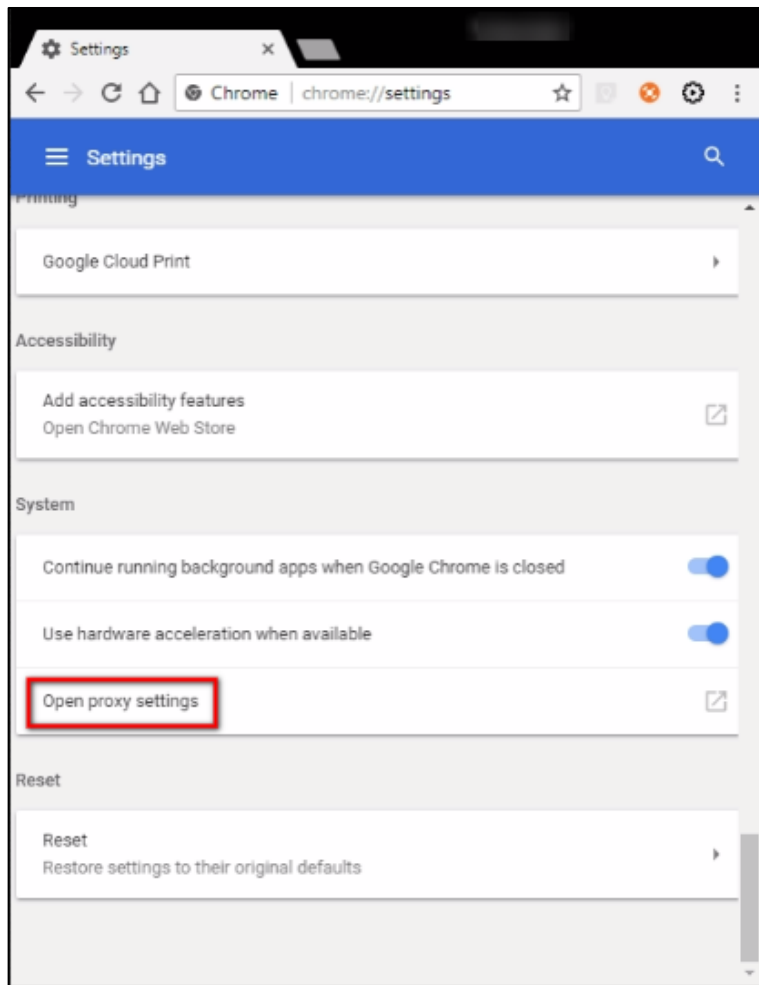


3.3 SETTING UP OF PROXY IN CHROME

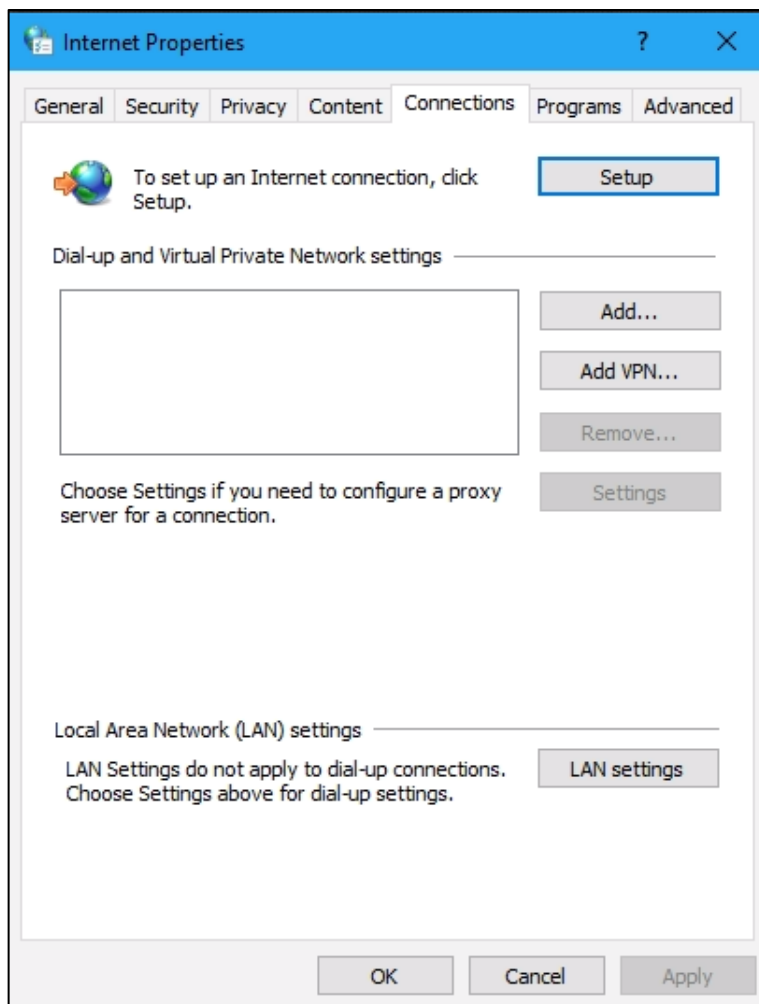
- In the chrome, open Chrome settings.
- Scroll down and click Advanced.



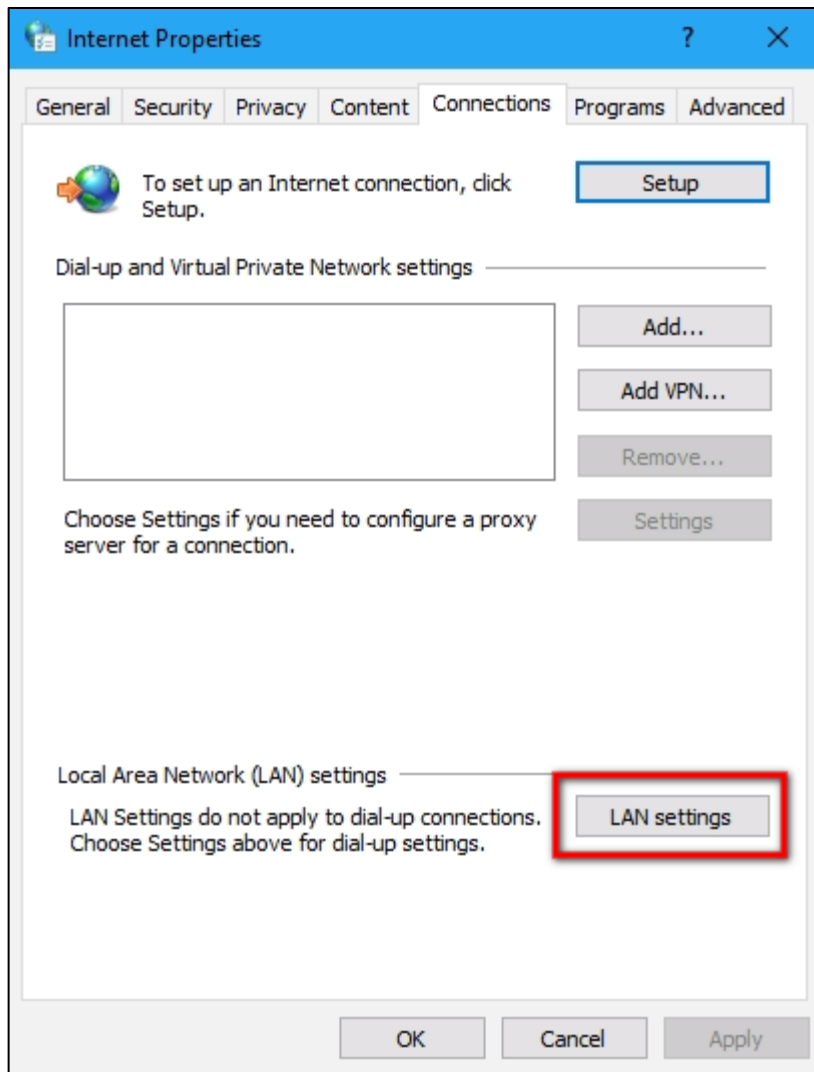
- Again, scroll down and click “Open Proxy Settings”.



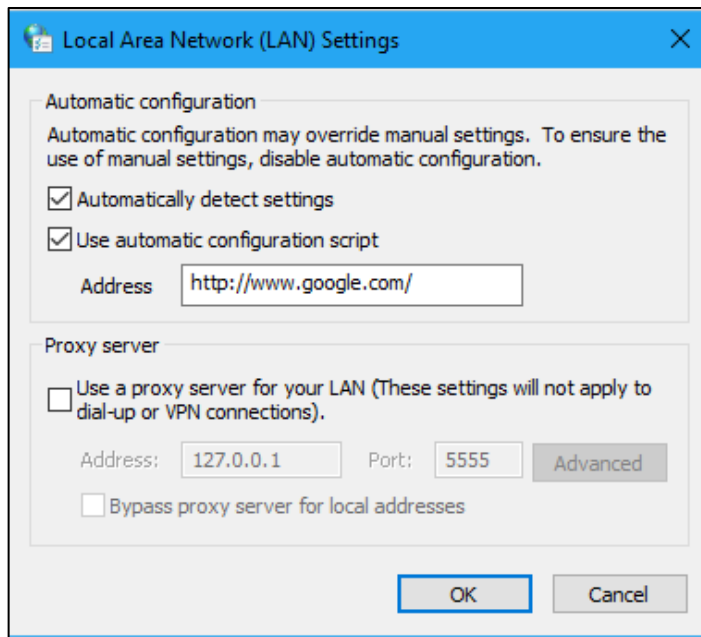
- A dialog box will appear.



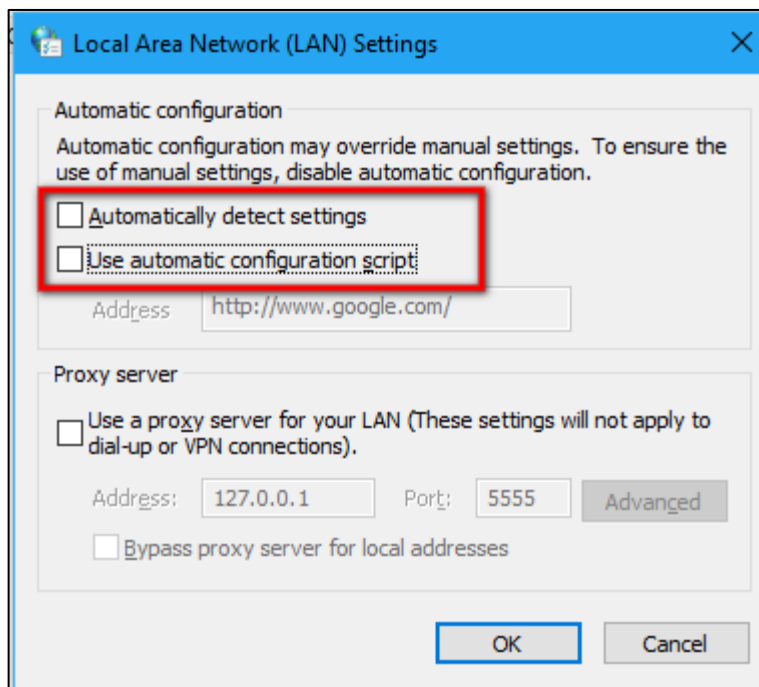
- In the dialog box, click LAN Settings.



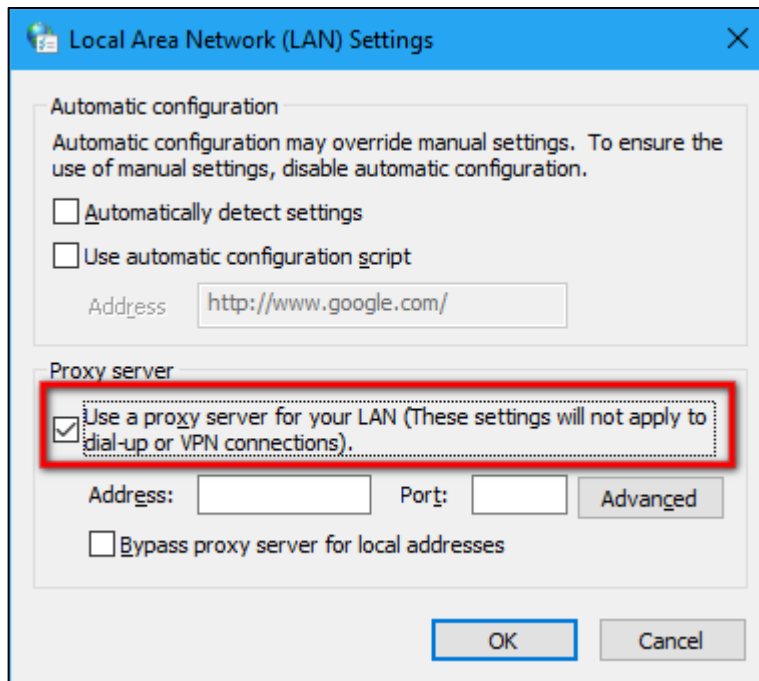
- It will display another dialog box.



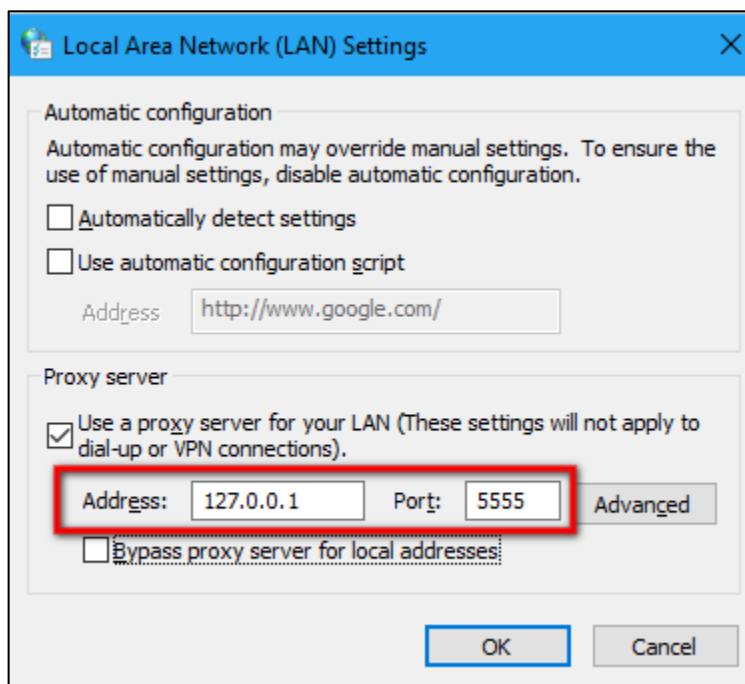
- In the dialog box, uncheck the Automatically detect settings and Use automatic configuration script under Automatic configuration.



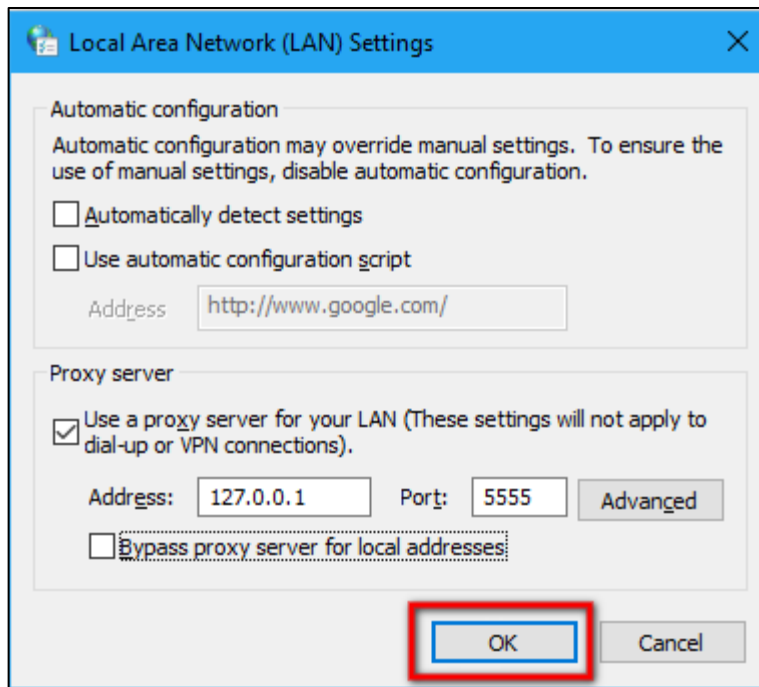
- In the same dialog box, check the box under Proxy server.



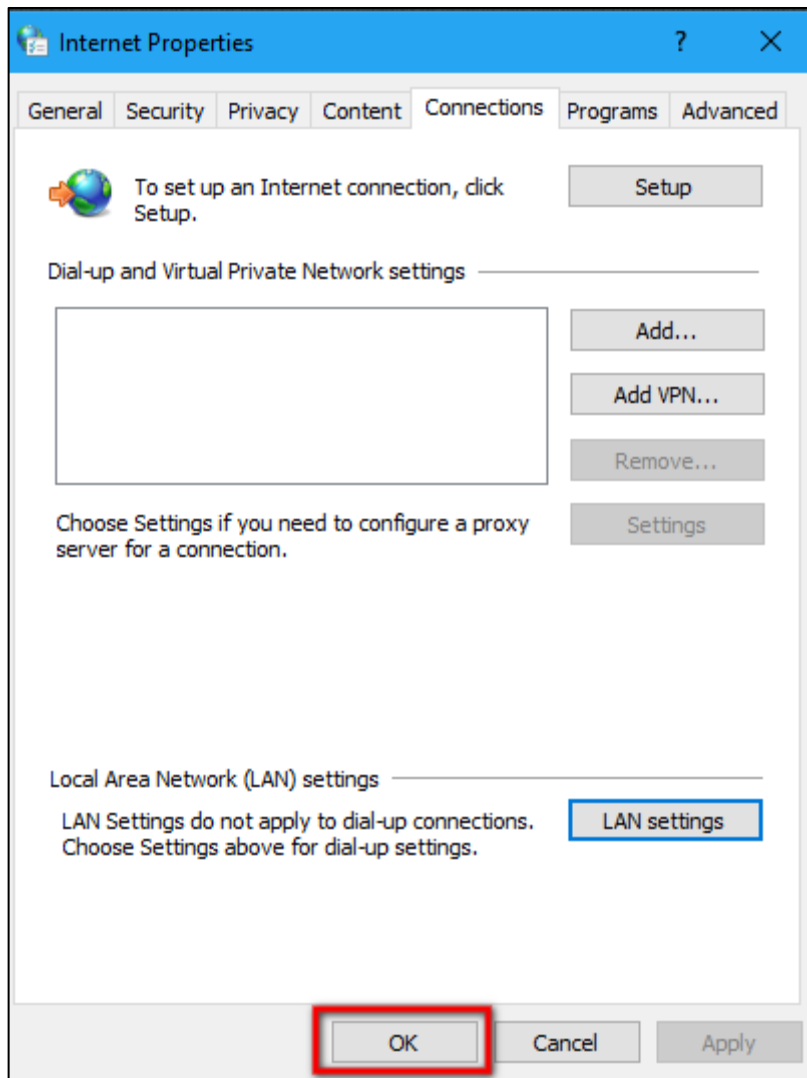
- On checking the box, the Address and Port text boxes becomes active.
- In the address text box, enter the ip address as "127.0.0.1" and in the port text box, enter the port number as "5555".



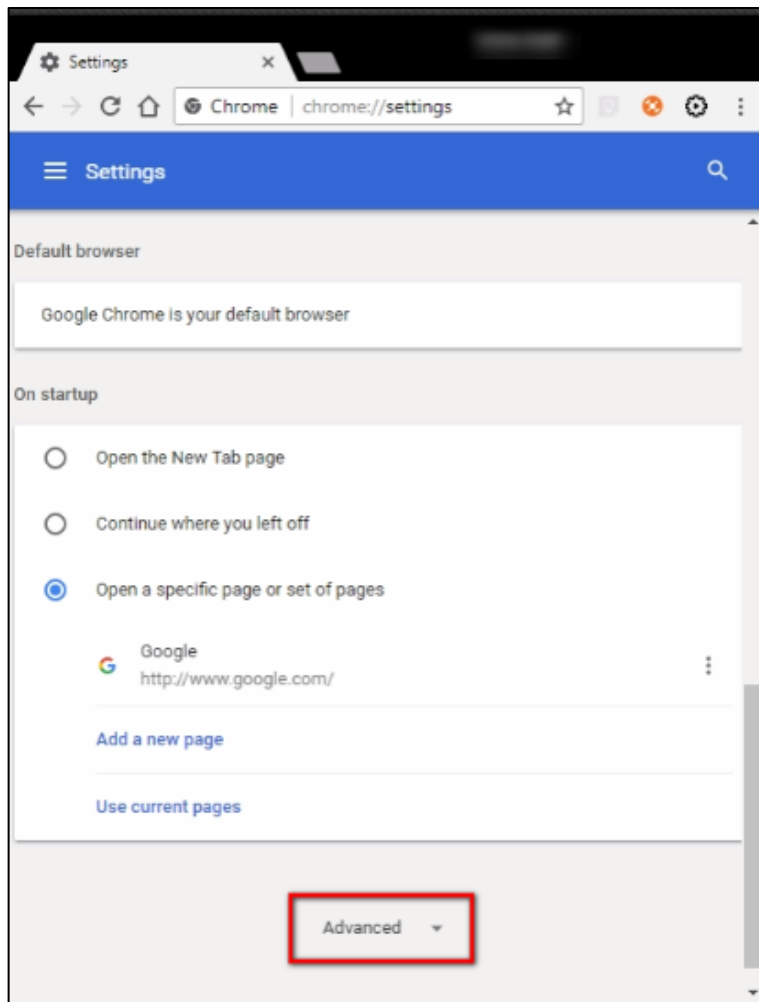
- After entering the address and port number, click Ok.



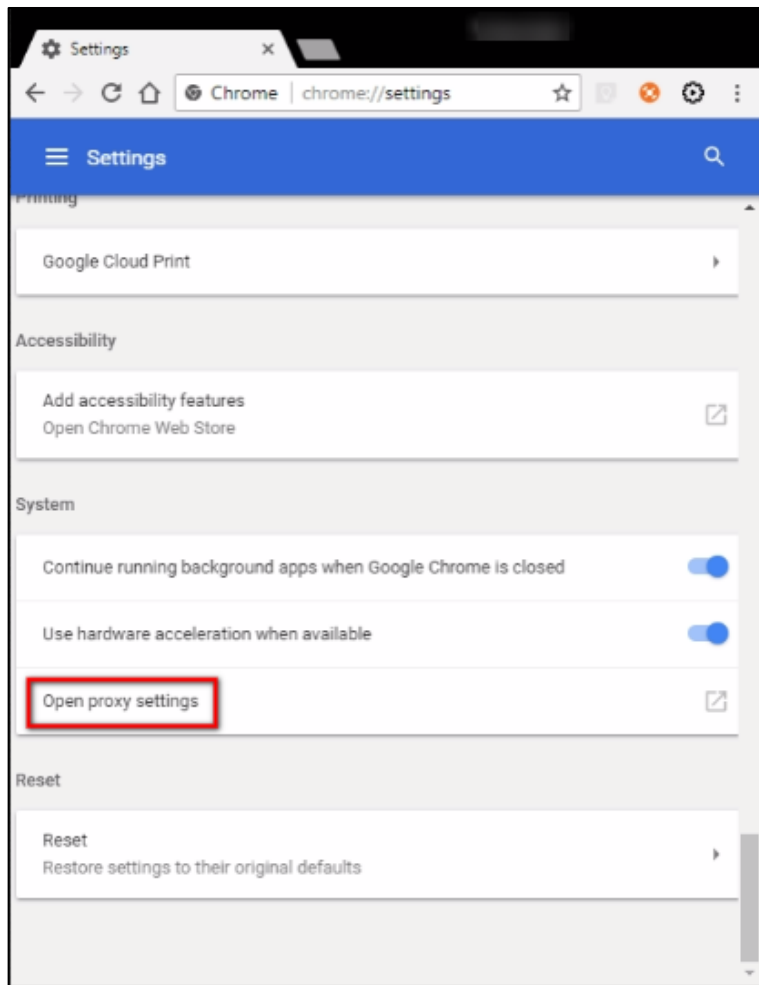
- Again, in the internet properties dialog box, click Ok.



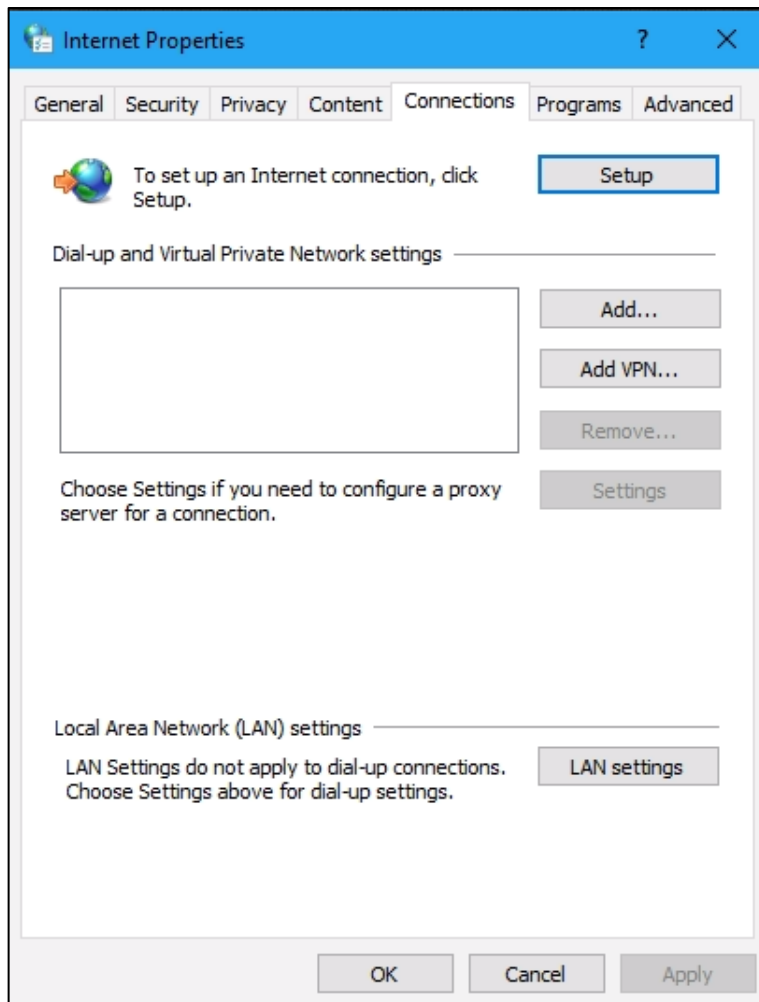
- When the user has finished the recording, the user has to revert back the changes in order to connect to the internet.
- To connect the revert the changes back, open Chrome Settings.
- Scroll down and click Advanced.



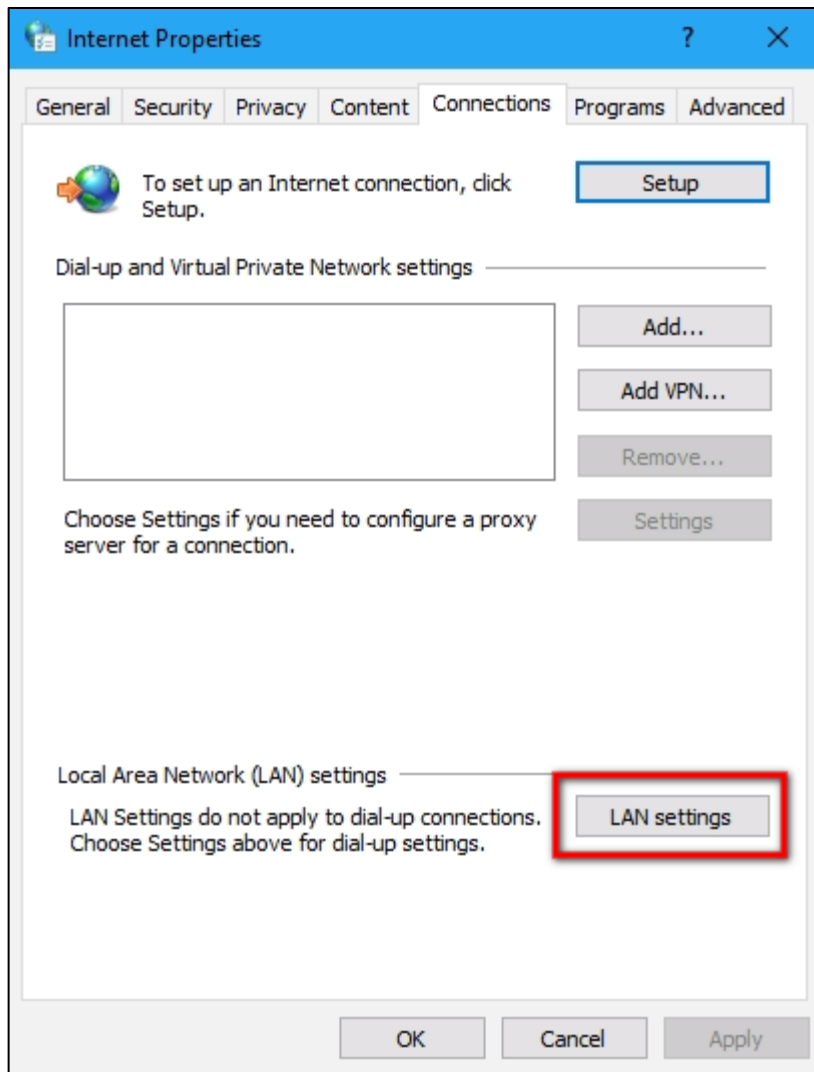
- Again, scroll down and click “Open Proxy Settings”.



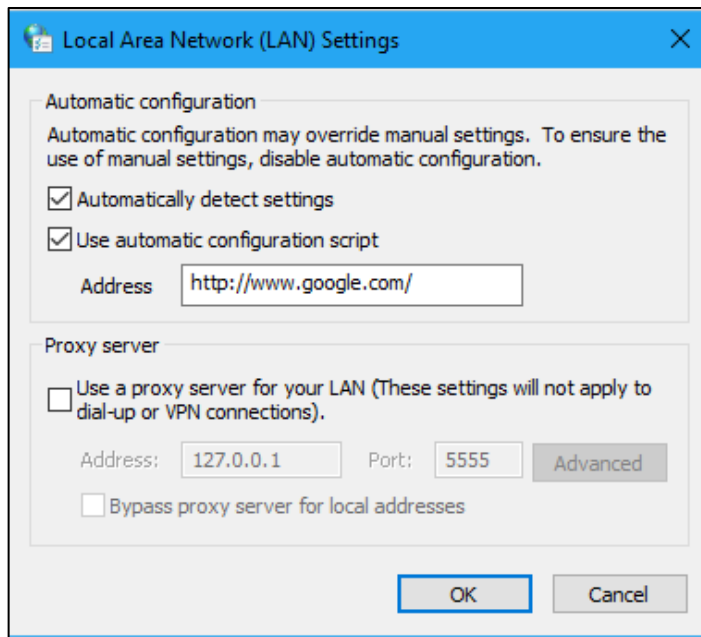
- A dialog box will appear.



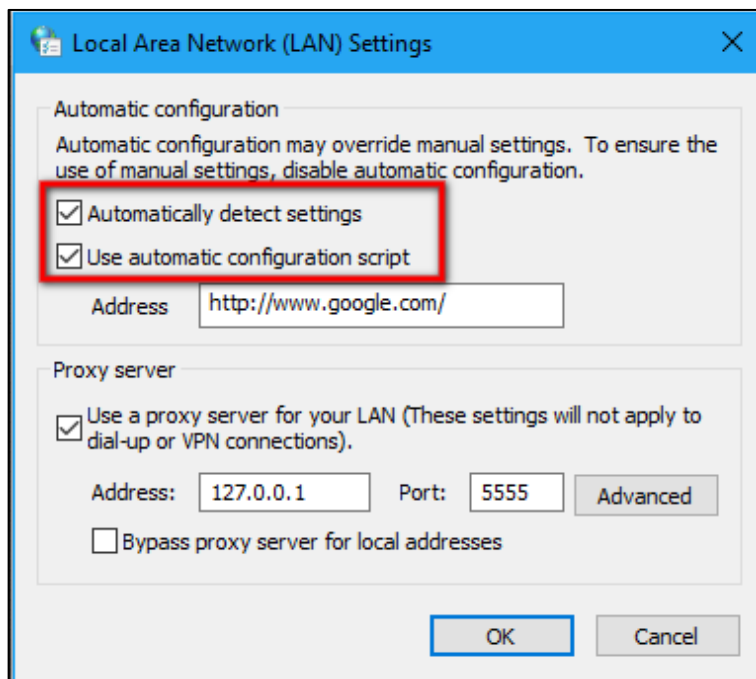
- In the dialog box, click LAN Settings.



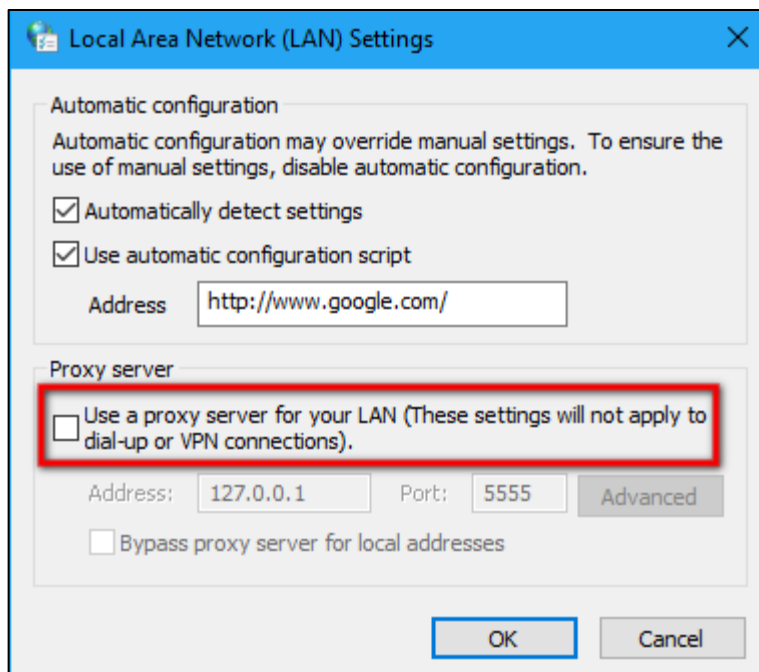
- It will display another dialog box.



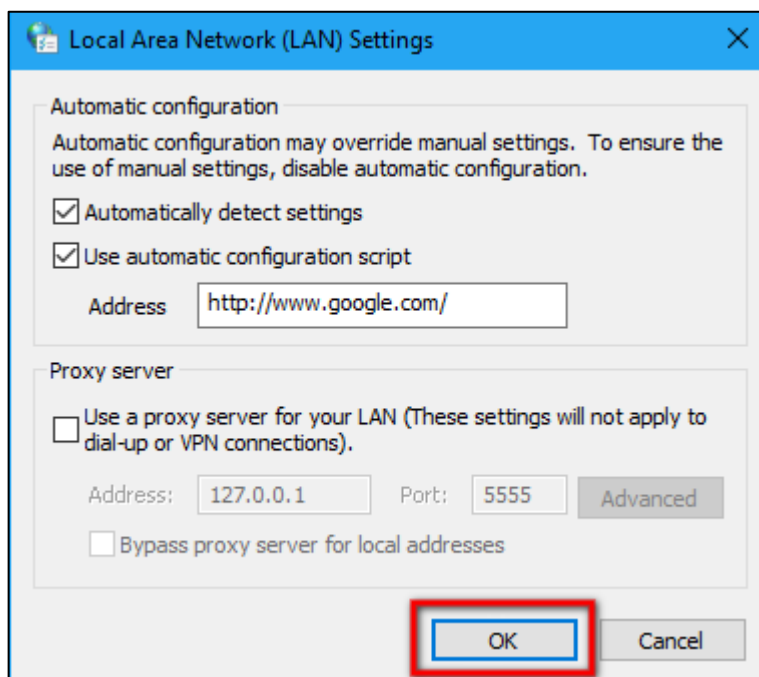
- In the dialog box, check the Automatically detect settings and Use automatic configuration script under Automatic configuration which will be unchecked earlier.



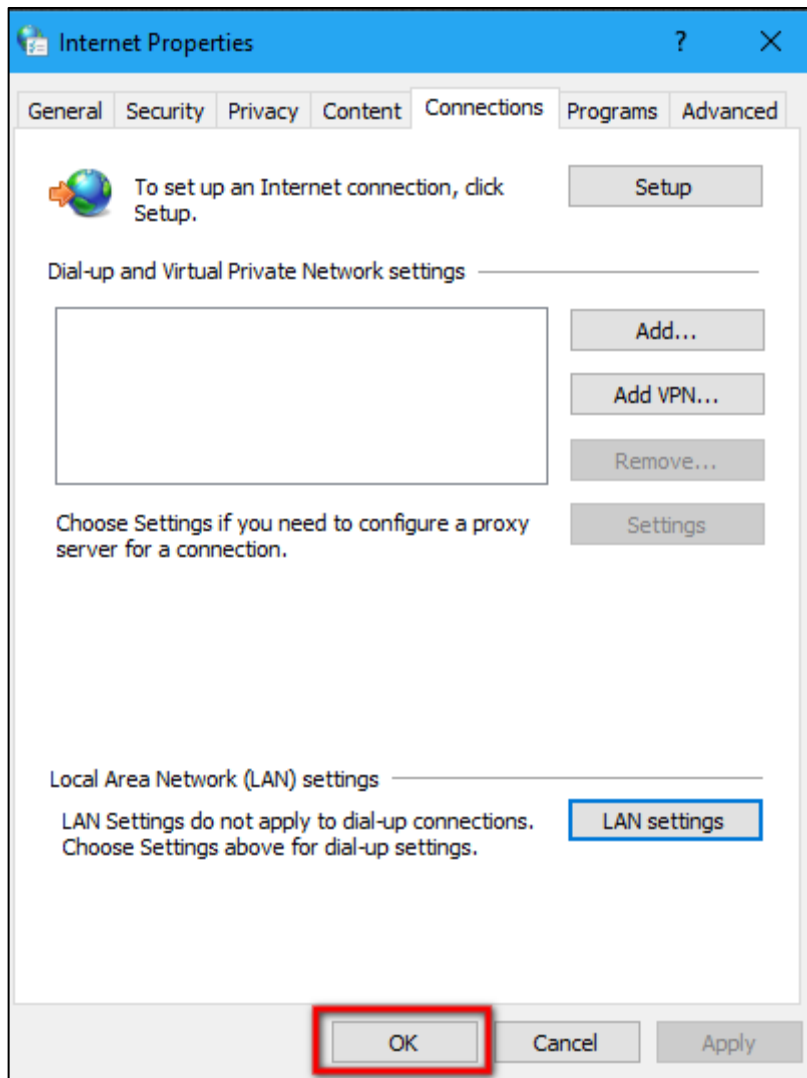
- In the same dialog box, uncheck the box under Proxy server which will be checked earlier.



- After checking and unchecking the boxes, click Ok.

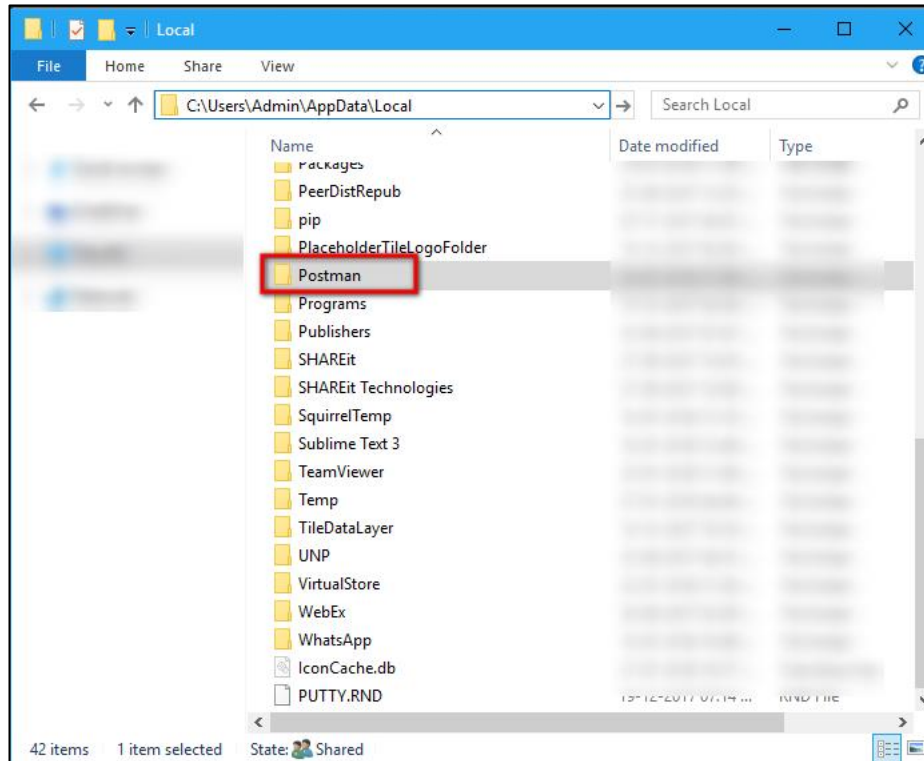


- Again, in the internet properties dialog box, click Ok.



3.4 CHANGING FOLDER LOCATION OF POSTMAN

- The default folder location of Postman will be in C: > Users > <System_Name> > AppData > Local.



- The user has to remove the folder from the above location and paste under C: drive directly.

