EVERYWEB – USER MANUAL

Contact:

sales@testonneed.com

Follow us:

LinkedIn and Twitter

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
0.1	Sahana Badri	First Draft version	

Review & Approval

Requirements Document Approval History

Approving Party	Version Approved	Signature	Date

Requirements Document Review History

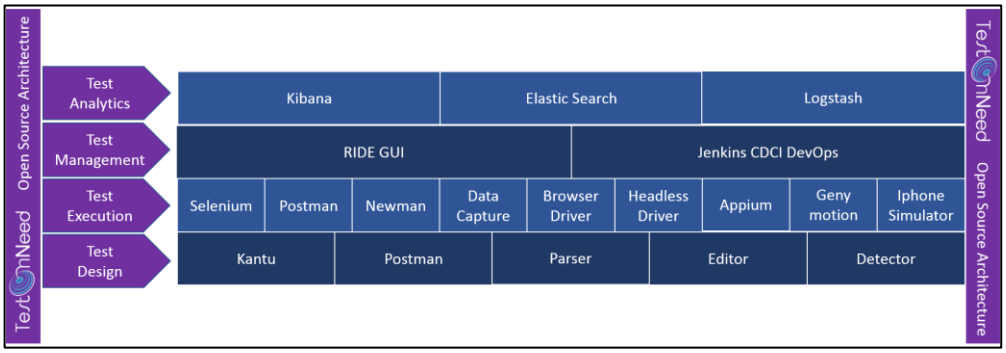
Reviewer	Version Reviewed	Signature	Date

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.

- **Kantu**

  This is used to record the actions performed by user on the Web Browser, thus automating the manual actions and creating the automation test cases.

- **Postman**

  This is used to record and automate the process of posting requests to server and fetching the corresponding responses back while the user will be executing the actions.

- **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.

- **Selenium**

  It is used to automate browser to execute web UI test cases.

- **Browser driver**

  It is used to emulate the user actions on Web Browser UI for executing the automation tests in browsers like Chrome, Firefox, IE etc.

- **Appium**

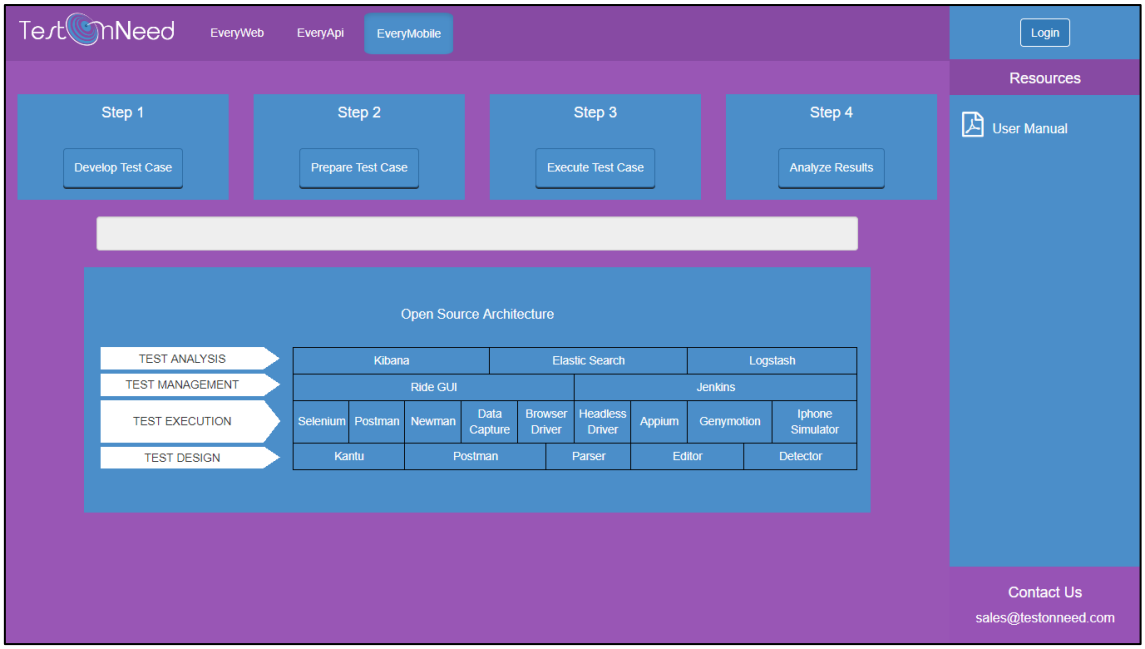
  Appium is an automation tool for running scripts and testing native applications and mobile-web applications on android or iOS using a web driver.

- **Emulator**

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

4. DESCRIPTION OF THE WEBSITE

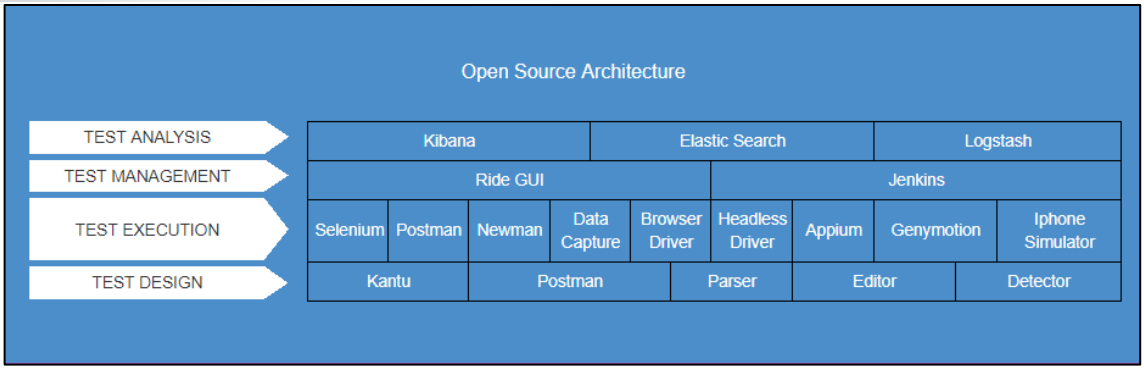
- In order to start the project, there are certain steps to be followed by the user. To know how to start the project, please refer **Quick Start Guide**.
- The outline of the Website looks like below:



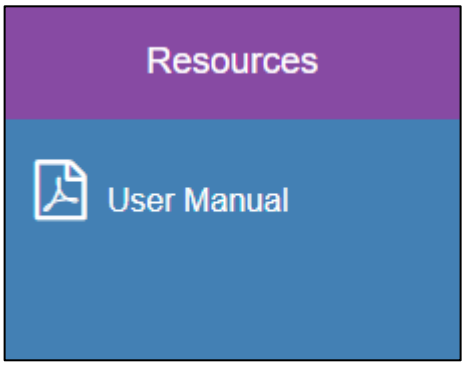
- The application has three tabs namely – EveryWeb, EveryAPI and EveryMobile.

- 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.



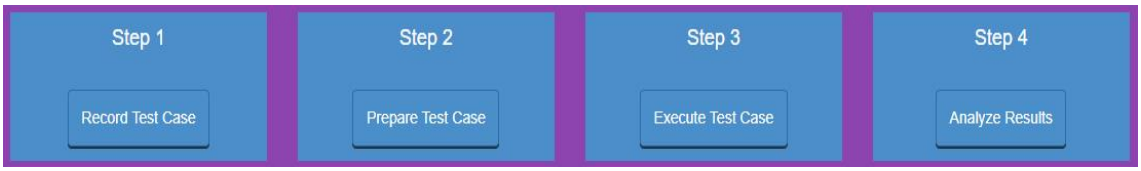
- The user manual guides the user through the web application. It will also change from tab to tab since working of each tab is different. It is a hyperlink, on clicking it, will open the respective document in pdf format in a new tab in the browser.



5. EVERYWEB PAGE

EveryWeb is a web application testing ecosystem using open sources. It is used to develop and execute the test cases for the frontend using browser and recorded backend API calls for web applications.

- The **Prerequisites of EveryWeb** are the following:
  - Advanced users should know Python in order to edit the Selenium scripts.
  - To create new graphs is Kibana, the user should know Selenium.
- 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.



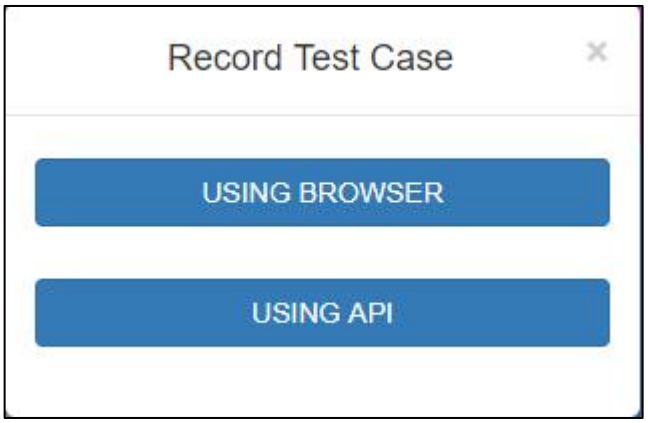
- **Record Test Case**
  - The user can record the frontend functionalities using Kantu browser and backend API's using Postman.
- **Prepare Test Case**
  - In the background, the downloaded JSON file gets converted to its corresponding Python file.
- **Execute Test Case**
  - The user can execute the 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 in detail

5.1 RECORD TEST CASE

The Record Test Case is used to record a scenario which is used for the further testing process. This helps the user to record the browser functionalities and API's that are being called.

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.

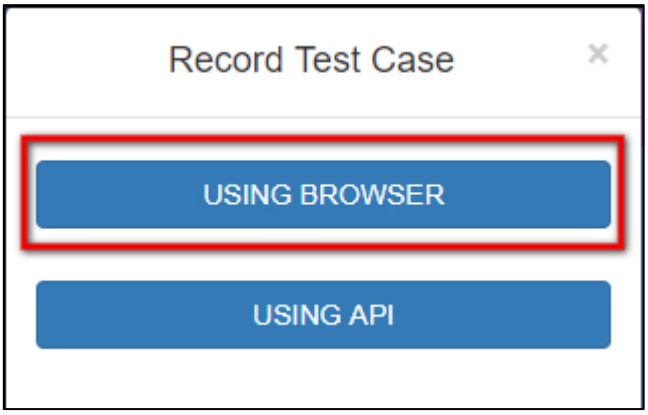


5. From the pop-up, user can choose either of **Using Browser** and **Using API** based on the requirement.

5.1.1 RECORD - USING BROWSER

The user can use this button to record the browser or frontend functionalities using which the user can perform automated testing. The user can record the test case using the Kantu browser. On click of the Using Browser button, the Kantu opens and it will be ready for the recording a new test case. And after recording the test case, the user has to save the test case and download the test case as JSON only. The user should follow the naming conventions that are explained later in this section. The detailed process of the recording using the Kantu browser is explained below.:

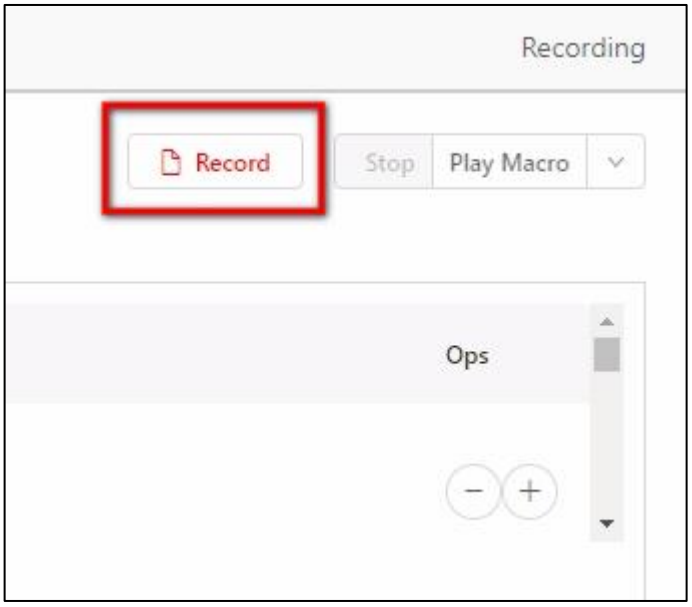
1. Click on **Using Browser**.



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.

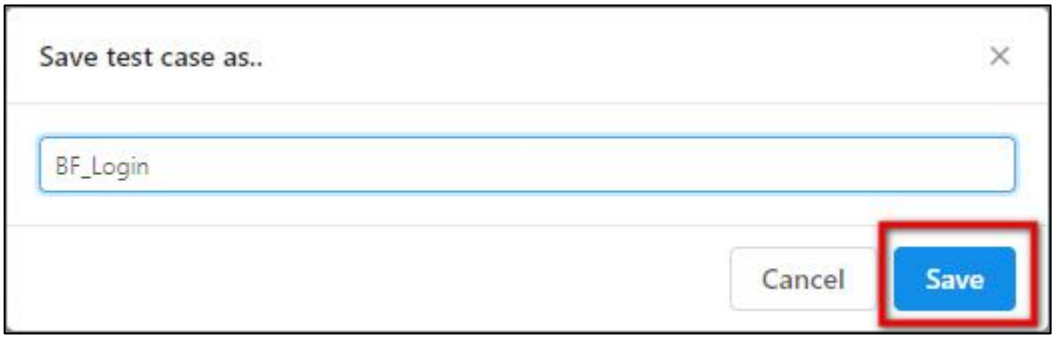


5. Now open the web application that has to be recorded.
6. Perform the steps.
7. To stop click the Record button again. It will change to blue from red.

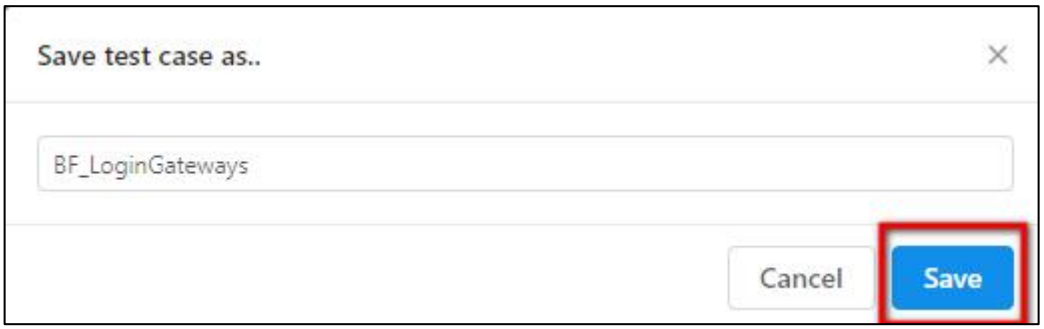
8. Now save the recording by clicking the save button.

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**.

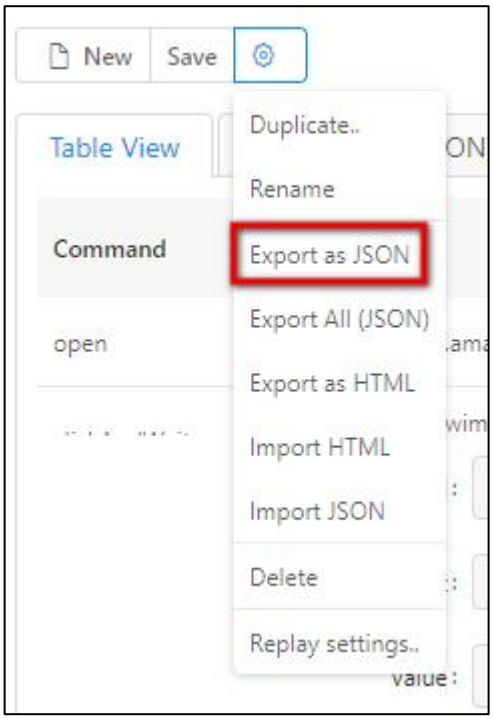


13. Load testing is performance testing technique using which the response of the system is measured under various load conditions. The load testing is performed for normal and peak load conditions. In that case, the test case should be saved as **BL_<TestCase_Name>**. For example,

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**.



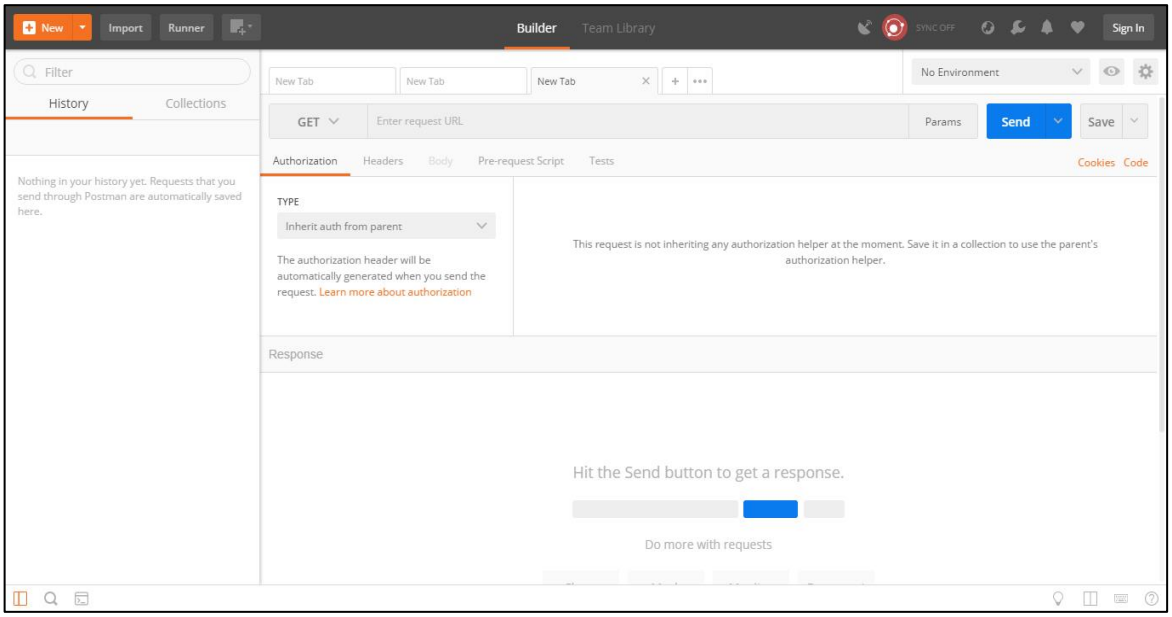
17. Now the JSON file gets downloaded in the default downloads folder.

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

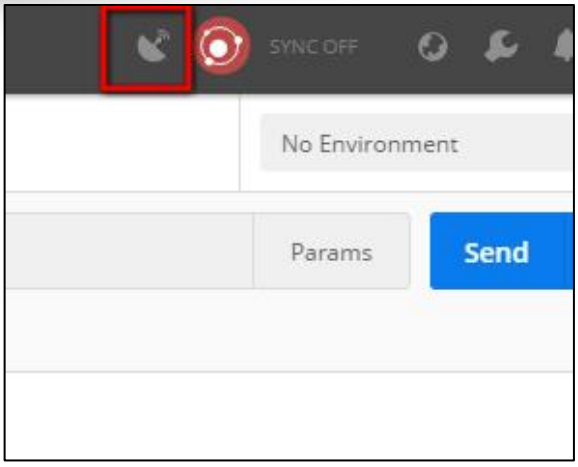
5.1.2 RECORD - USING API

The user can use this button to record the API or services that are being called in the backend. The user can record the test case using the Postman. On click of the Using API button, the new window of the chrome and Postman will open. The changes in the settings that are to be made to the opened chrome window is also explained in the reference section. After the settings are changed, it will be ready for the recording a new test case. And after recording the test case, the user has to save the test case and download the test case as JSON only. The user should follow the naming conventions that have been explained later in this section. The detailed process of the recording using the Postman is explained below.:

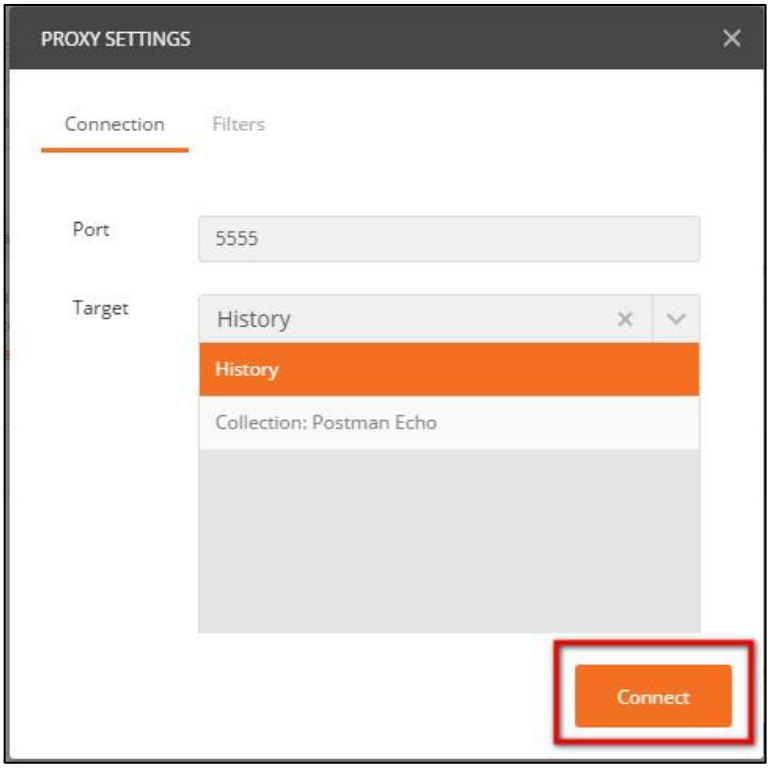
1. The user has to change the folder location of the postman. To change the folder location, click Here.
2. For setting up of proxy in order to record, click Here.
3. On clicking **Using API** button will open the Postman app.



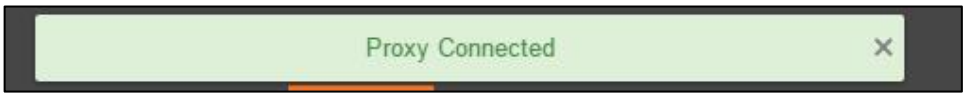
4. Start the proxy in postman application.



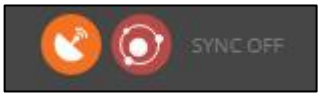
5. On clicking the proxy button, a pop window comes up and click **Connect** on it.



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

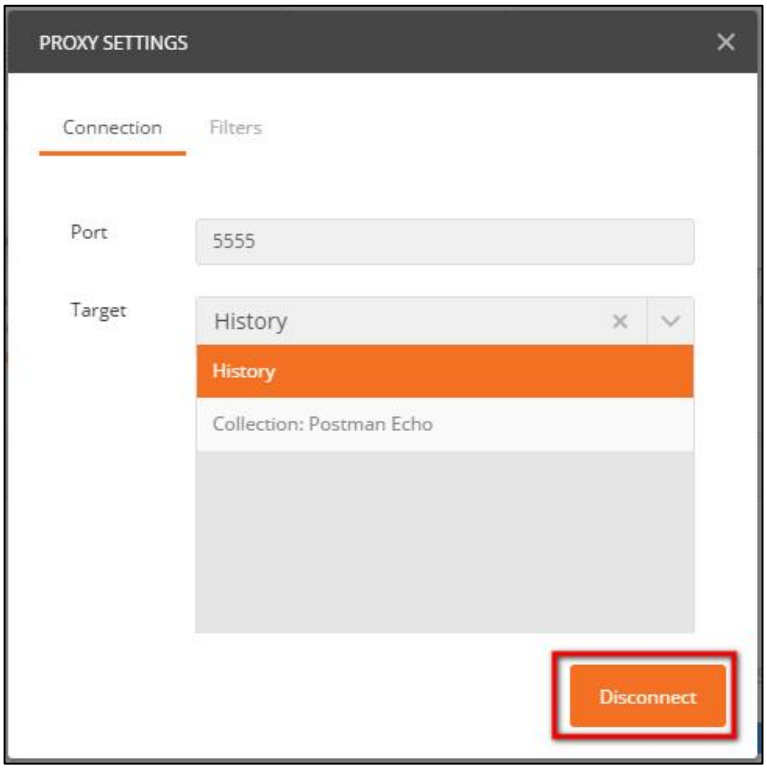


7. If it is connected, the proxy button changes to orange.

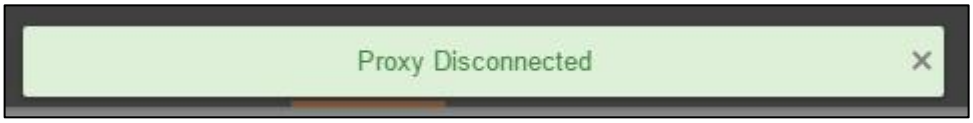


8. Now open the site that has to be record.

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.

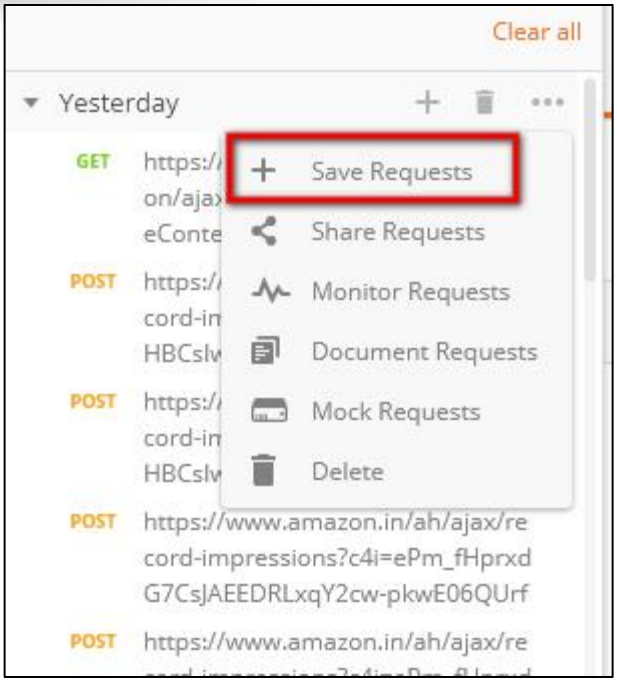


11. Once the disconnect button is clicked, at the top of the page a message will be displayed as **Proxy Disconnected**.

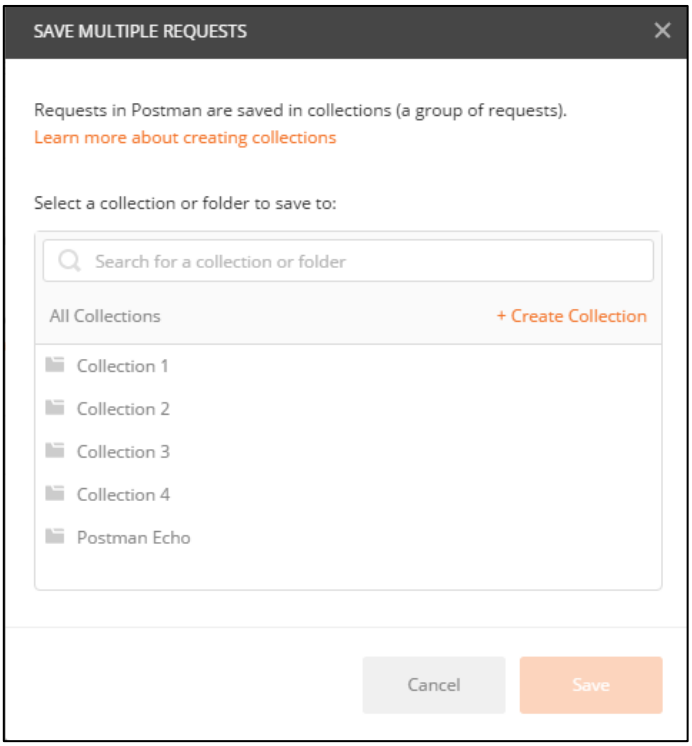


12. After disconnecting the proxy, the proxy button changes back to grey.

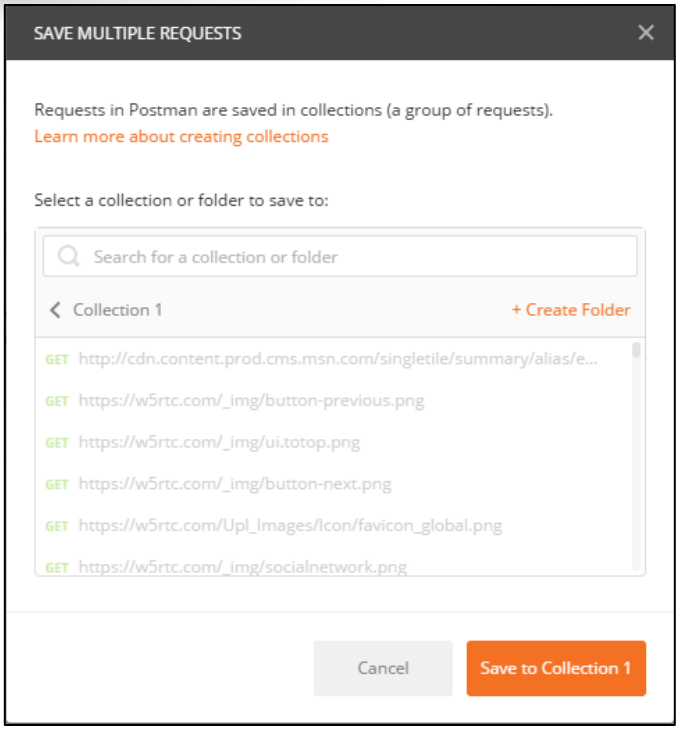
13. In the postman application, click Save Request and save in the appropriate collection.



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.



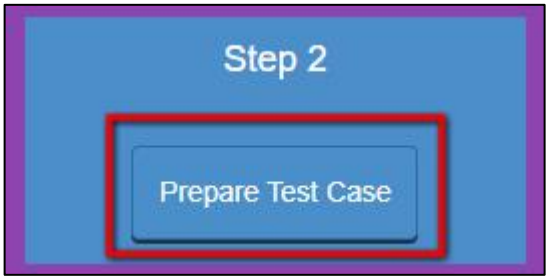
17. Now save the recording based on the type of testing functionality the user has recorded. There are 3 types of testing functionality – **Function, Automation and Load.**
18. Functional Testing is a testing technique that is used to test the features/functionality of the system or Software, should cover all the scenarios including failure paths and boundary cases. In that case, the test case 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 test cases**.
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.

5.2 PREPARE TEST CASE

The user will download the JSON file after recording the test case either from Kantu or Postman that will be used in this step. The user will upload the downloaded JSON file in this step. It will convert the uploaded JSON into its corresponding Python.

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



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

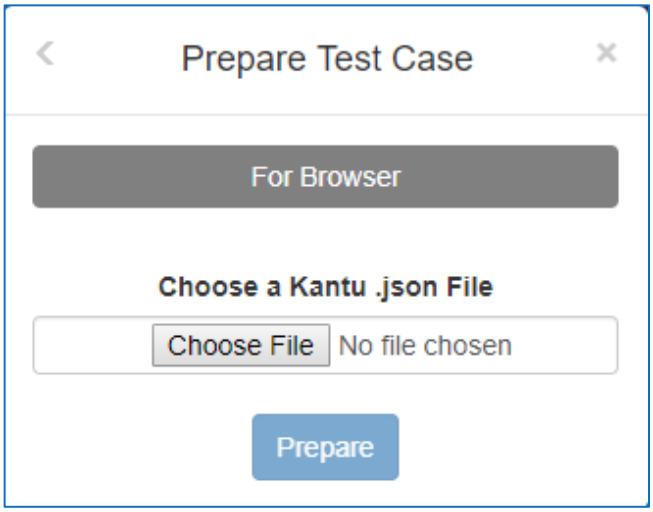
3. To upload the JSON file downloaded using the Kantu browser in the previous step then click **FOR BROWSER**.
4. To upload the JSON file downloaded using the Postman in the previous step then click **FOR API**.
5. From the pop up select either – **FOR BROWSER** or **FOR API** button.

5.2.1 PREPARE – FOR BROWSER:

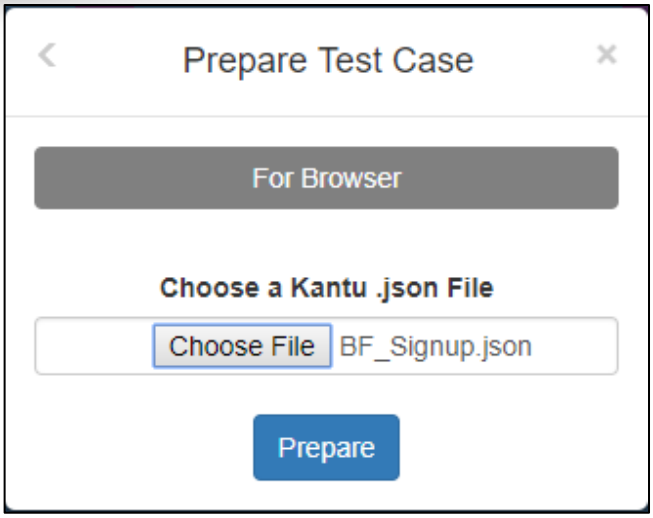
Here the user should upload the JSON file that was downloaded after recording from the Kantu browser and clicks Prepare button. The uploaded JSON files gets converted into its Python file that will be used in the execution of the test case.

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

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

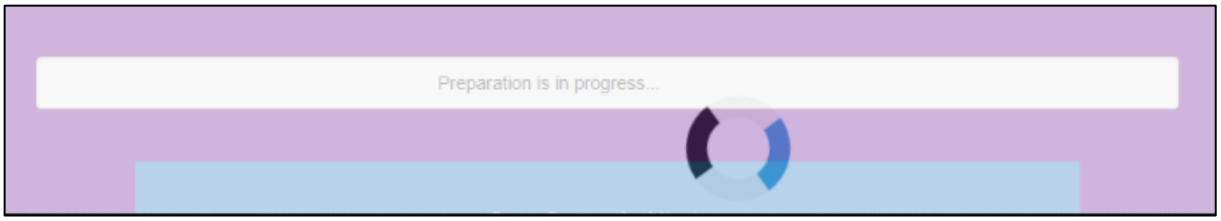


3. Click on the **Choose File** button.
4. Surfing the folder location where saved the JSON file that was downloaded in the previous step.
5. Now upload the JSON file.
6. The user can upload **only one file** at a time.
7. On uploading the JSON file, the **Prepare** button becomes active.

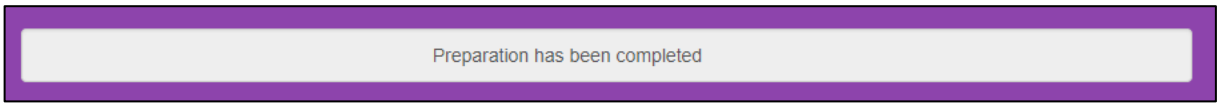


8. On uploading JSON file and clicking the **Prepare** button, the following will happen at the backend.
   - It creates a folder in the name of the test case that was mentioned in the previous step.
   - The folder gets stored in:

     > TON > WebTesting > 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.
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.

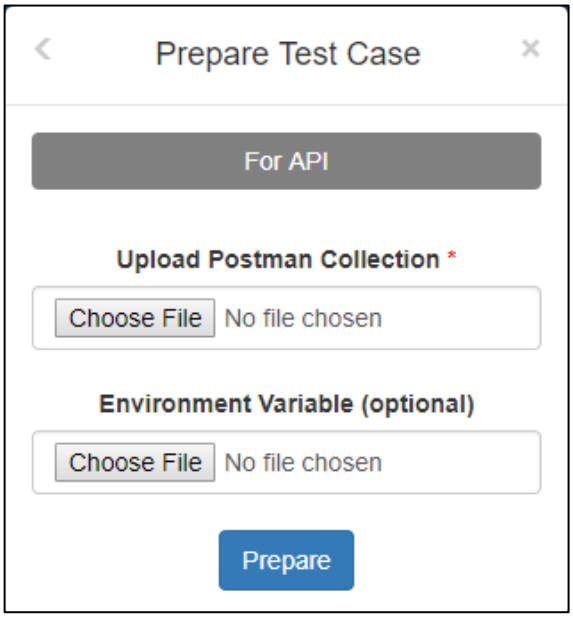


5.2.2 PREPARE – FOR API

Here the user should upload the JSON file that was downloaded after recording from the Postman along with the environmental variables JSON and clicks Prepare button. The environmental variables file must also be a JSON file and it is an optional file. The uploaded JSON files gets converted into its Python file that will be used in the execution of the test case. While converting the downloaded JSON and environmental variables JSON, it will become one JSON.

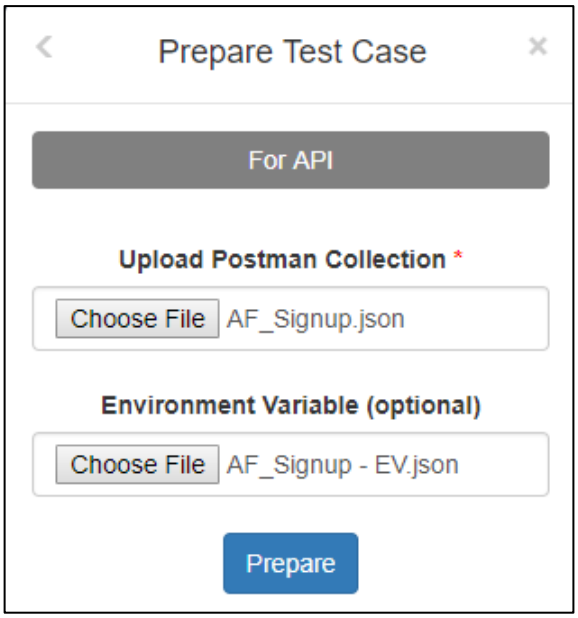
1. Click on the **For API** button.

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.

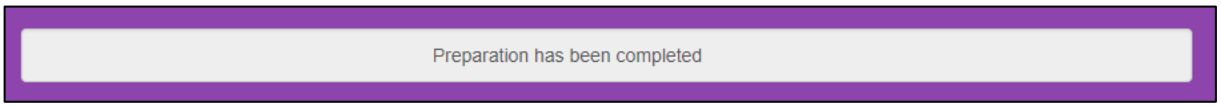
7. On uploading the JSON file, the **Prepare** button becomes active.



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.



5.3 EXECUTE TEST CASE

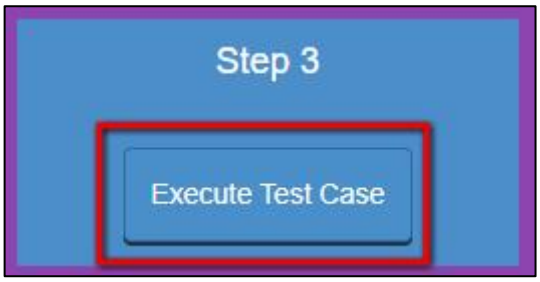
Here the user will be able to execute the test cases being developed in the previous steps. The user can choose any number of test case to execute, passing arguments for configuring the execution with the test management tools. the user will also be able to get the output of the automation execution.

The Execute Test Case button is used for the execution process. The process of execution takes place in two ways – GUI and TestOps.

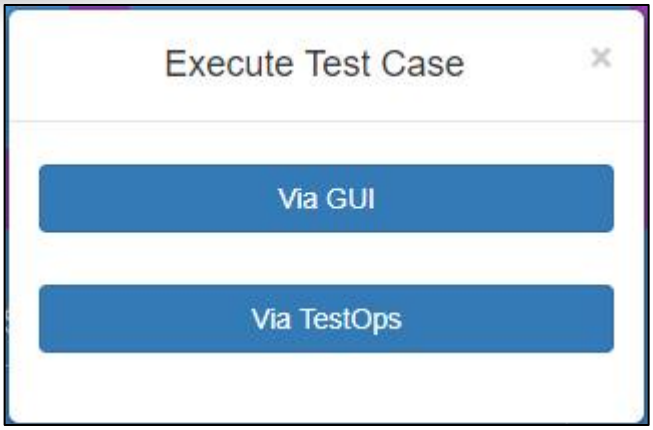
There are three types of testing that can be performed under GUI and TestOps – Functional, Automation and Load Testing.

- 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.
- Automation testing makes use of specialized tools to control the execution of tests and compares the actual results against the expected result.
- 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.

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



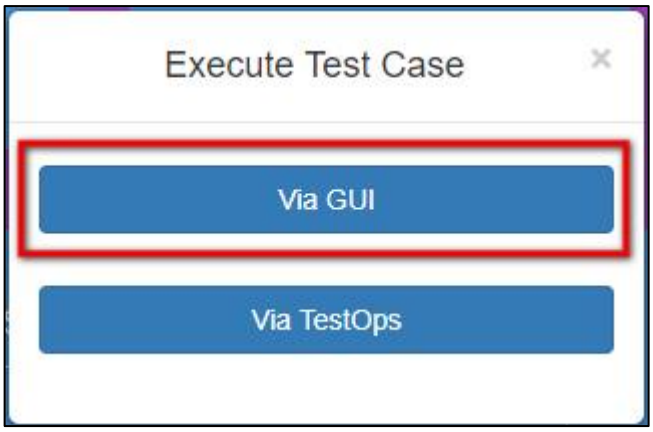
2. On clicking on the button, a pop up appears.
3. From the pop-up, select the button either as **Via GUI, Via TESTOPS.**


Execute Test Case
Via GUI
Via TestOps

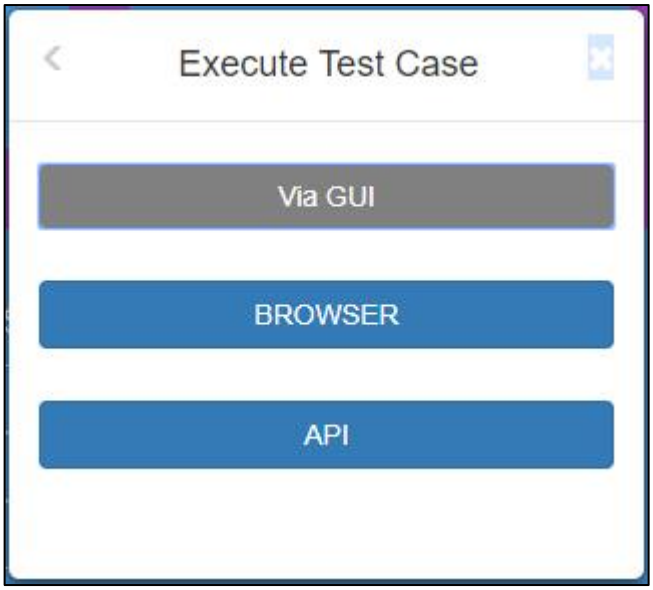
5.3.1 EXECUTE – VIA GUI

Now let's see how to execute the test case using the GUI option.

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



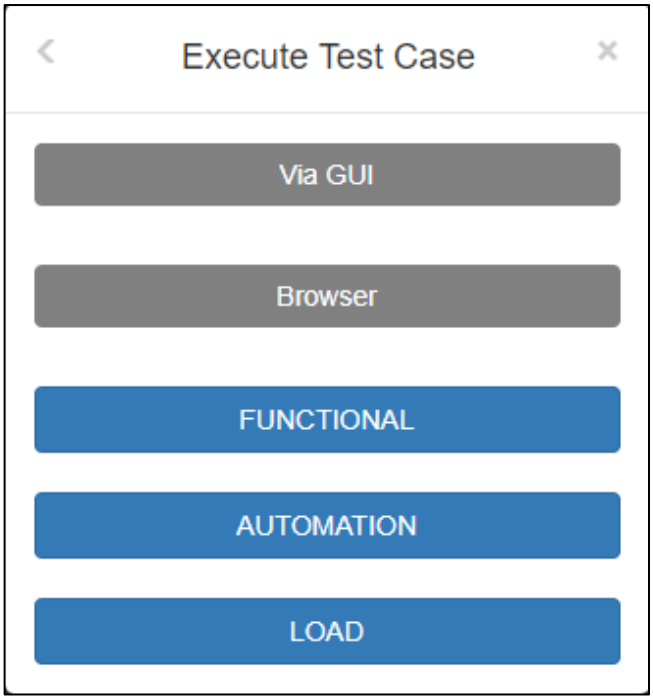
2. On clicking the button, execution through GUI can be done using 2 buttons– **Browser and API**.



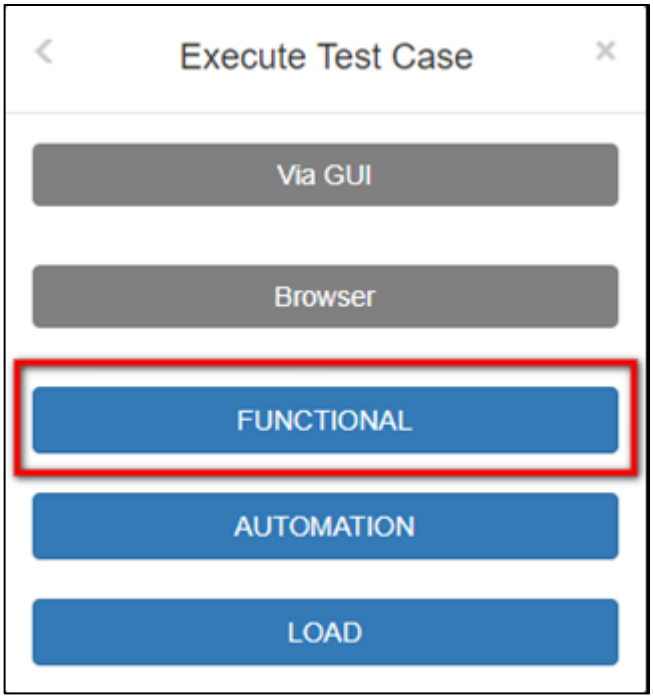
3. Under each button, 3 buttons – **Function, Automation and Load** will be found.

5.3.1.1 EXECUTE – VIA GUI – BROWSER

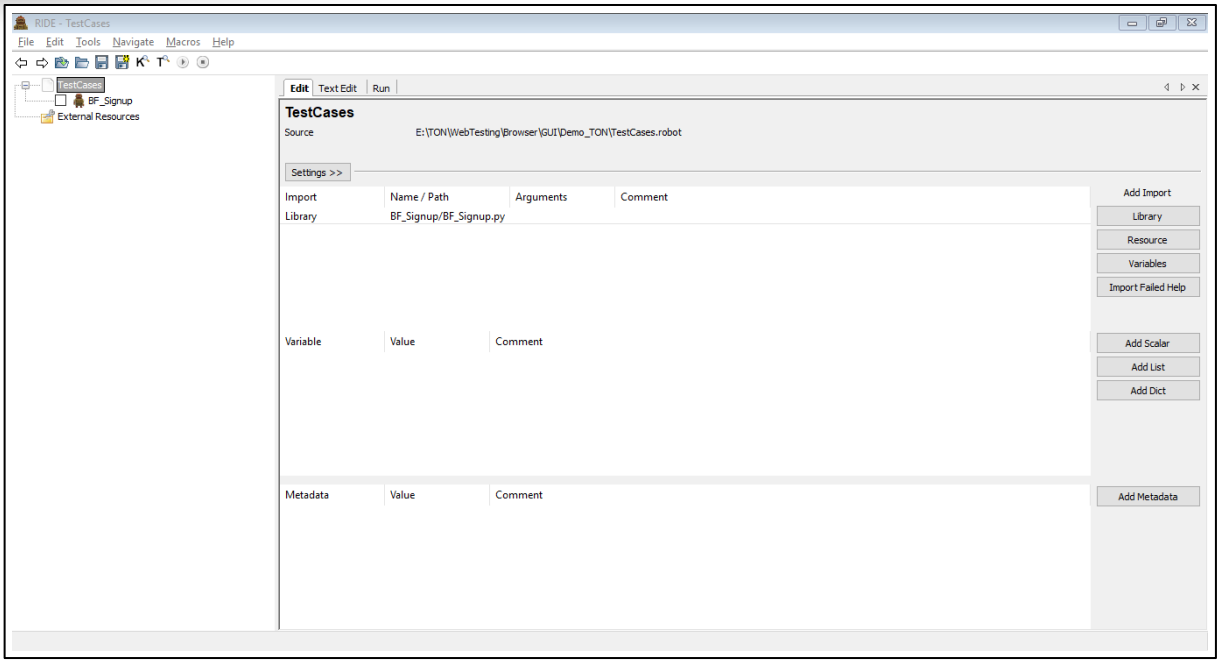
1. On click of the Via GUI button, 3 buttons appear – **Functional, Automation and Load**.



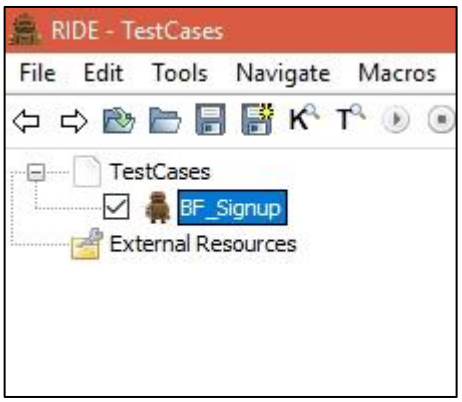
2. To perform functional testing, press **Functional** button.



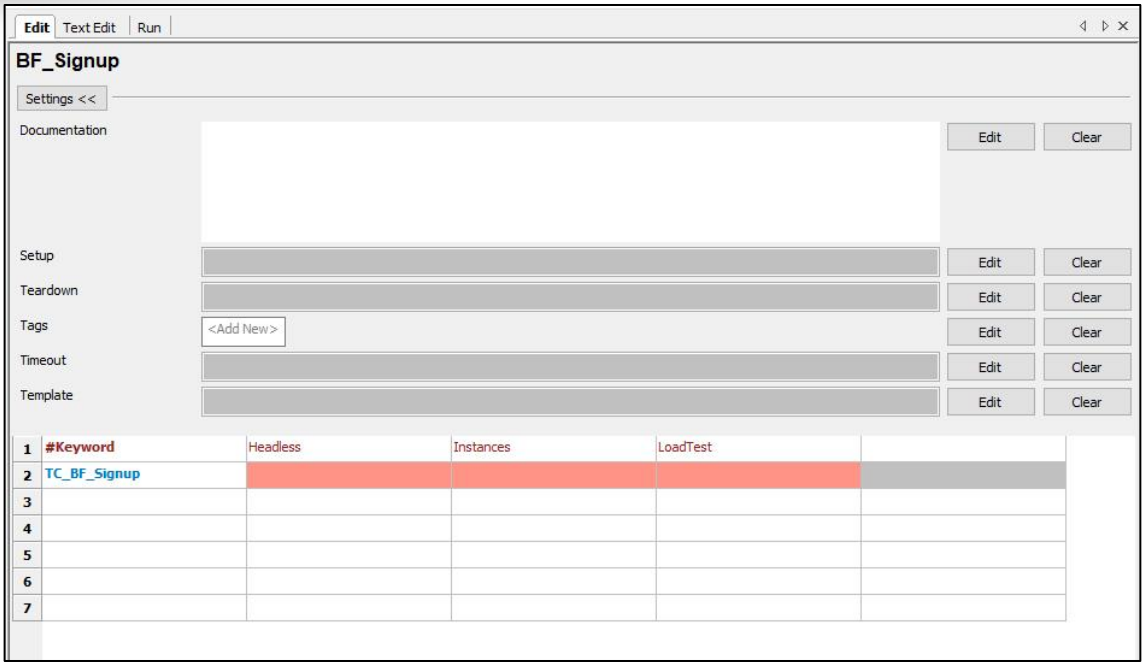
3. On click of the button, **RIDE** will open.



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.



7. The value of the parameters should be:

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

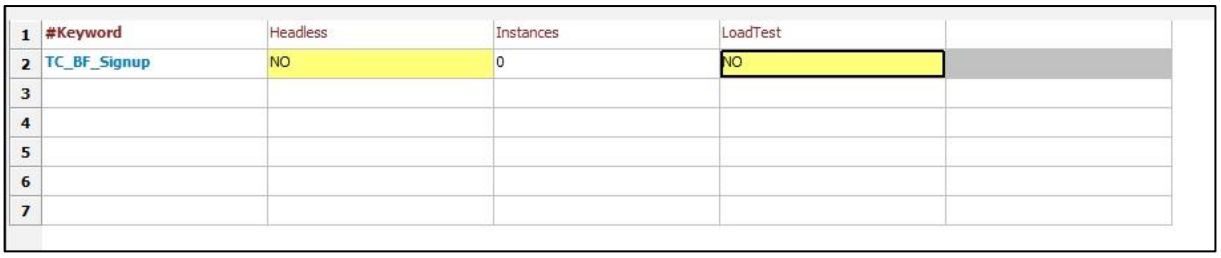
   For Headless = Yes, browser simulation takes place without visible browser.

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

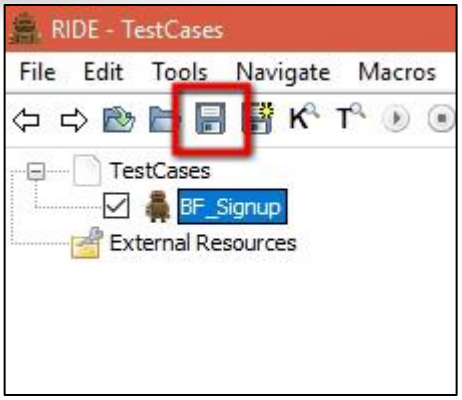
   Instances = 0 (Only valid for Load Test)

   LoadTest = No (Only valid for Load Test)

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



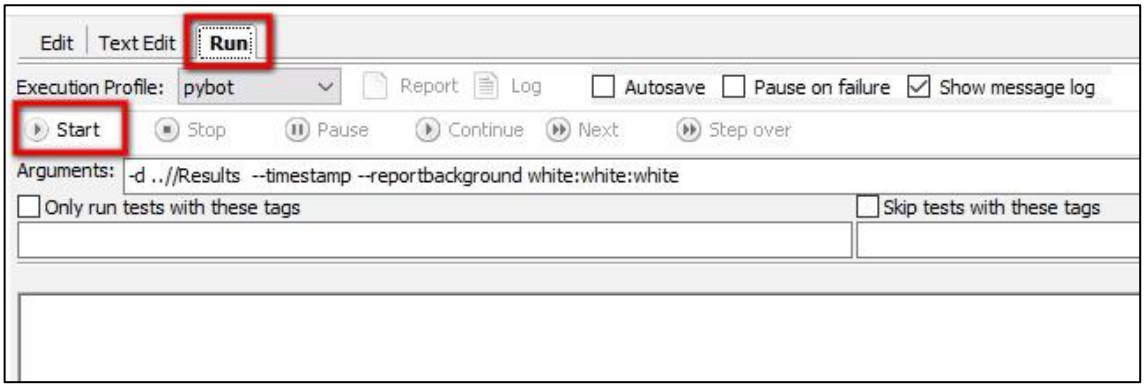
9. Click **Save** button to save the test case.



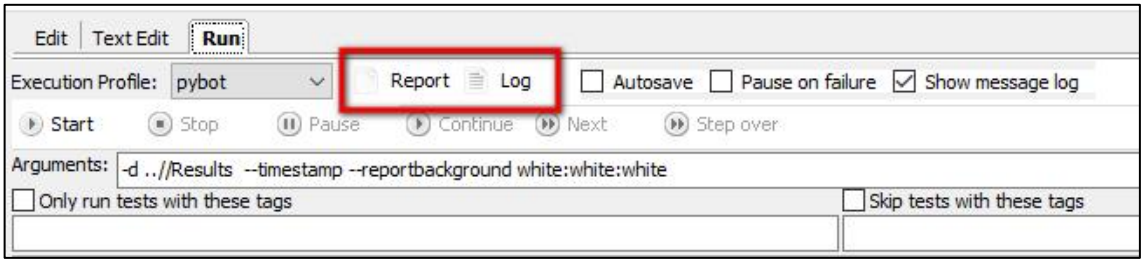
10. Configure the **Arguments** textbox by entering:

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

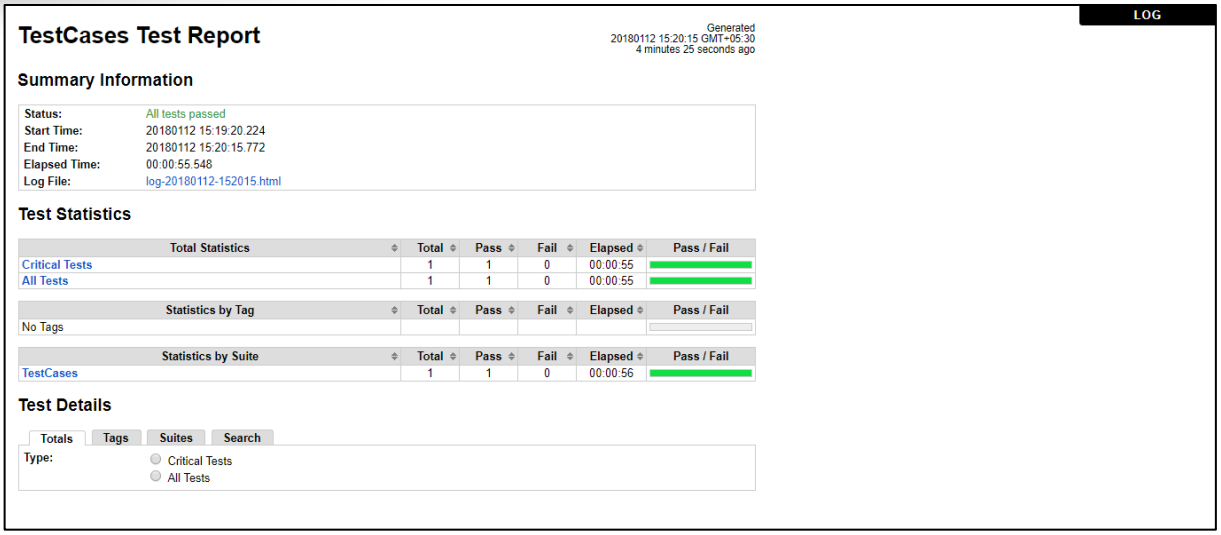
11. Under Run tab, click **Start** button to start the execution.



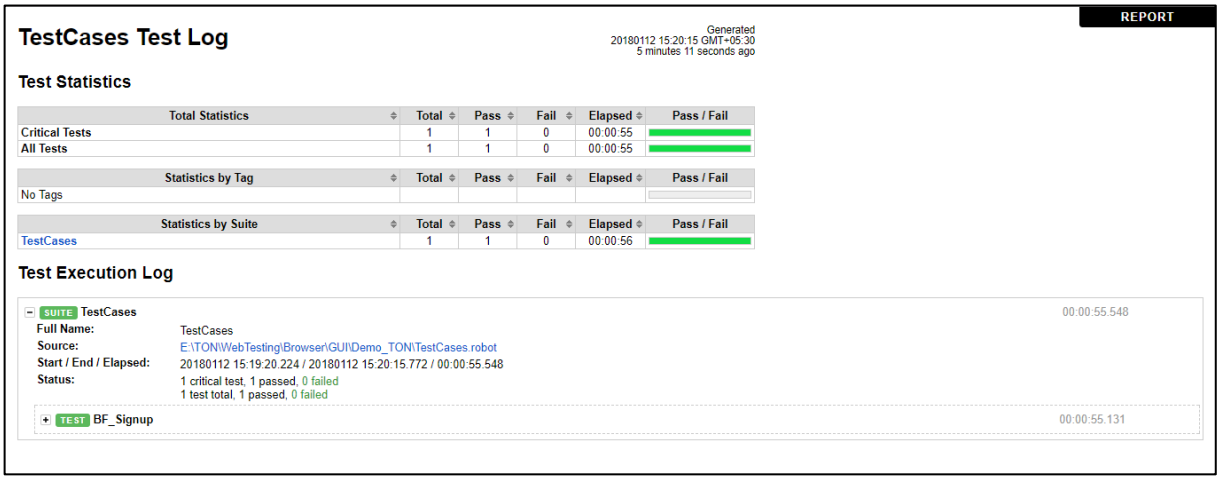
12. Click on the **Log / Report** button to view the result. The button becomes active on executing the test case.



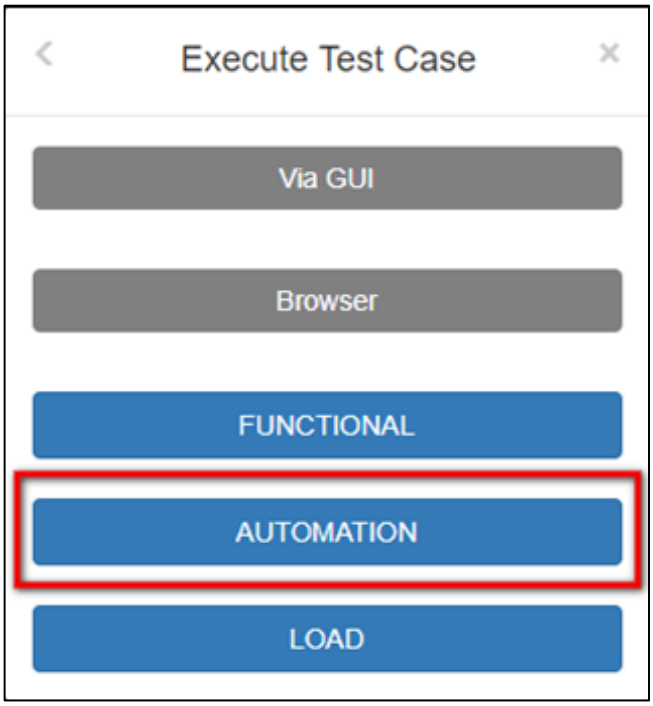
13. On click of the Report button, in the browser, the report appears.



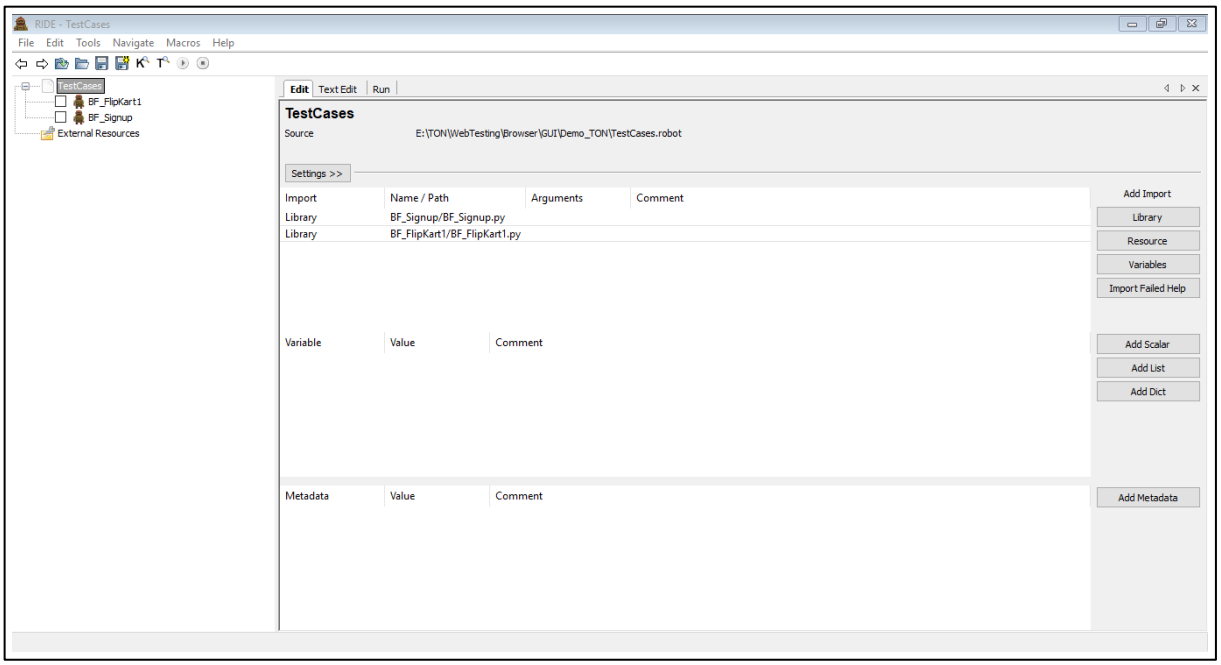
14. In the browser on the top right corner, the log button is there. Clicking on it, will show the log report and vice versa.



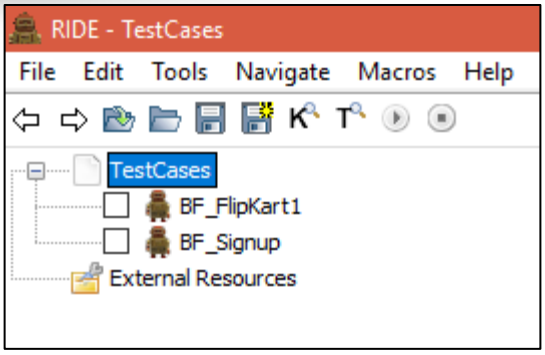
15. To perform automation testing of the JSON file downloaded using Kantu browser, click **Automation** button.



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

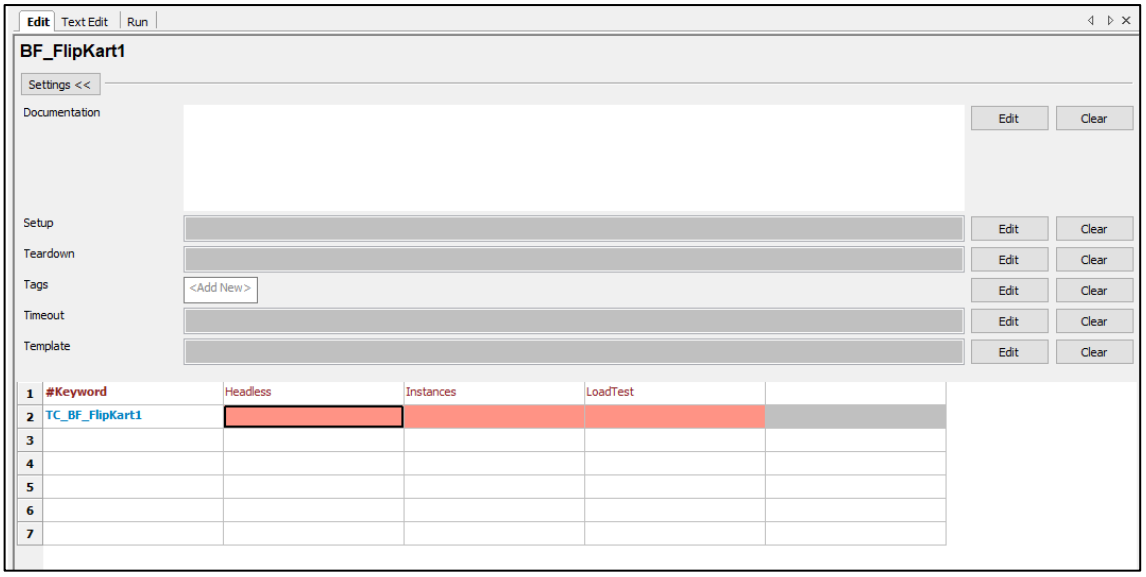


17. On the left panel, select multiple test cases to execute.



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

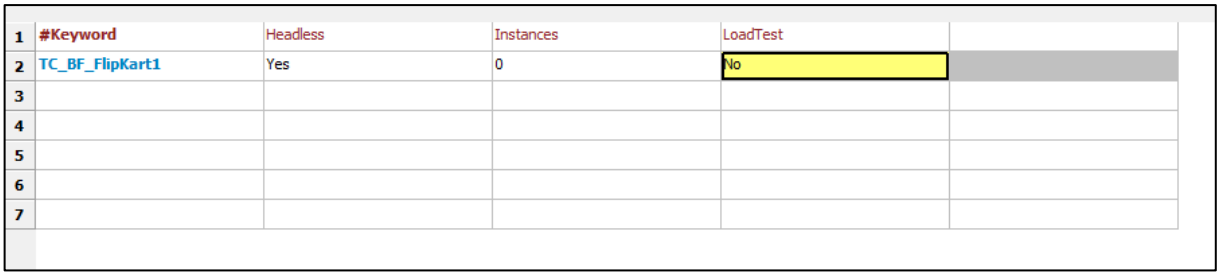
    For Headless = Yes, browser simulation takes place without GUI.

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

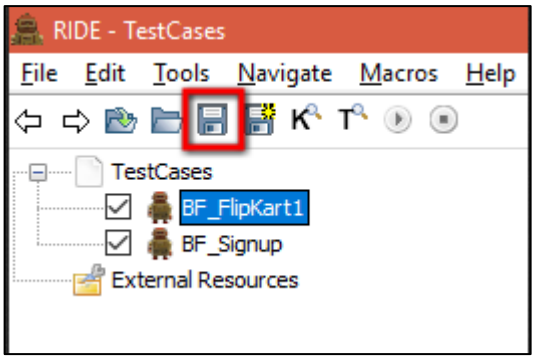
    Instances = 0 (Only valid for Load Test)

    LoadTest = No (Only valid for Load Test)

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



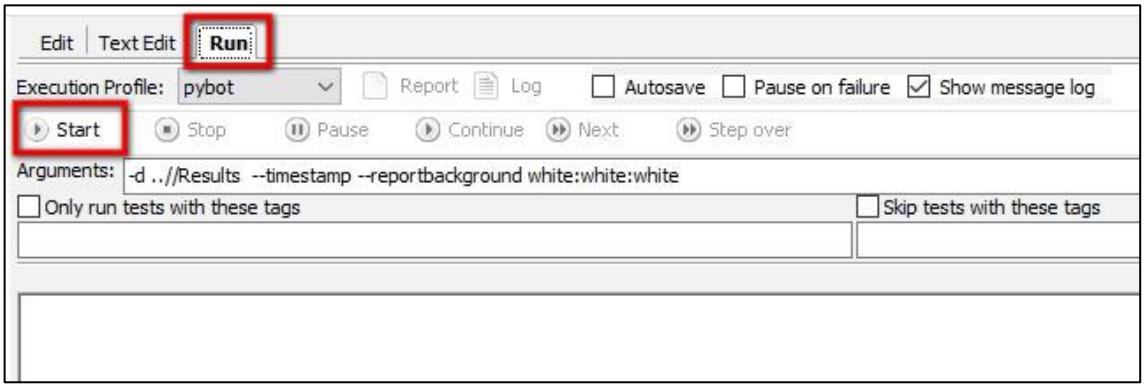
22. Click **Save** button to save the test case.



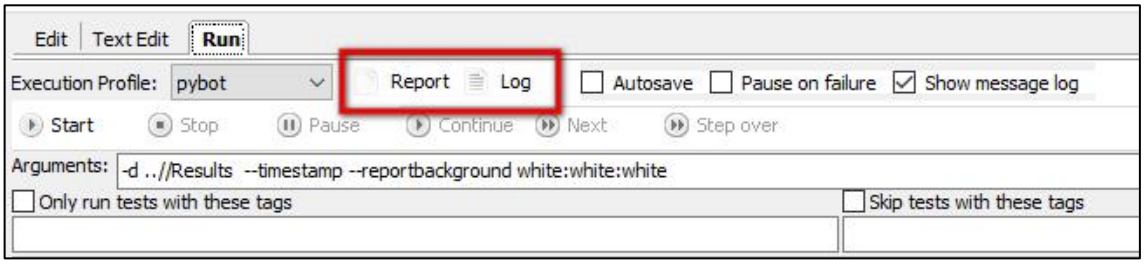
23. Configure the **Arguments** textbox by entering:

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

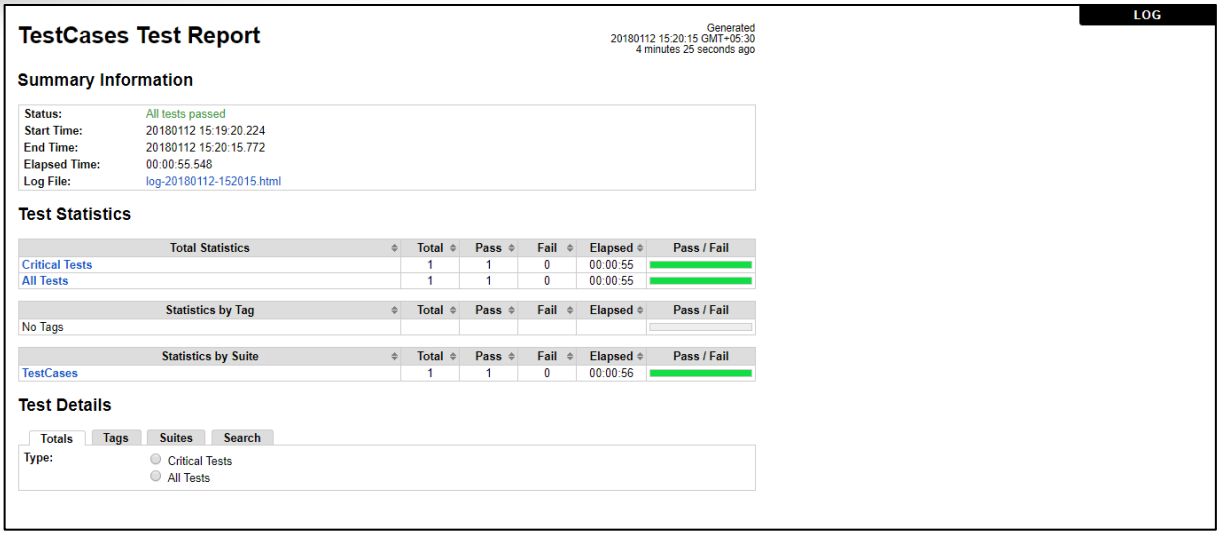
24. Under Run tab, click **Start** button to start the execution.



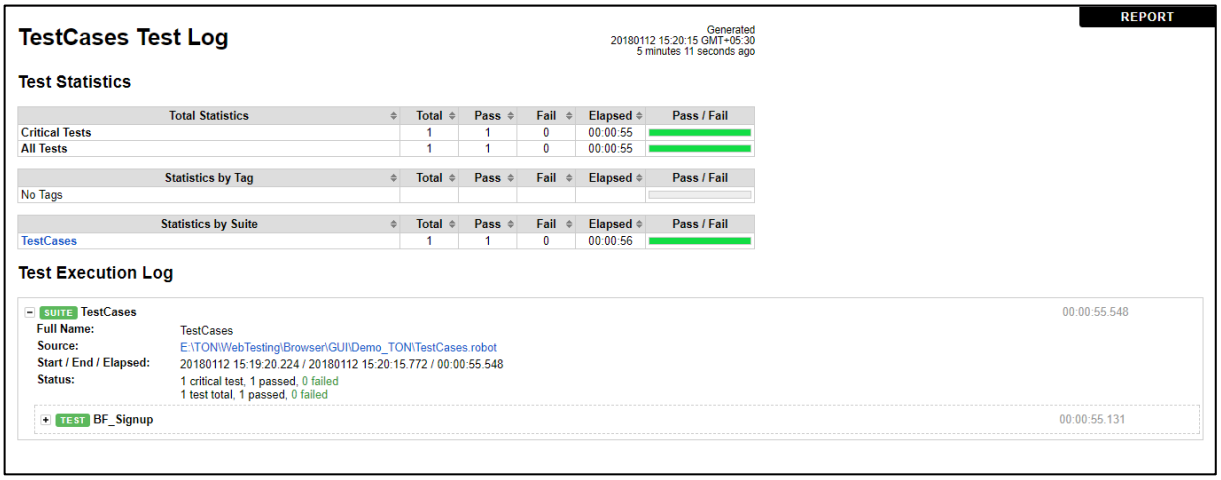
25. Click on the **Log / Report** button to view the result.



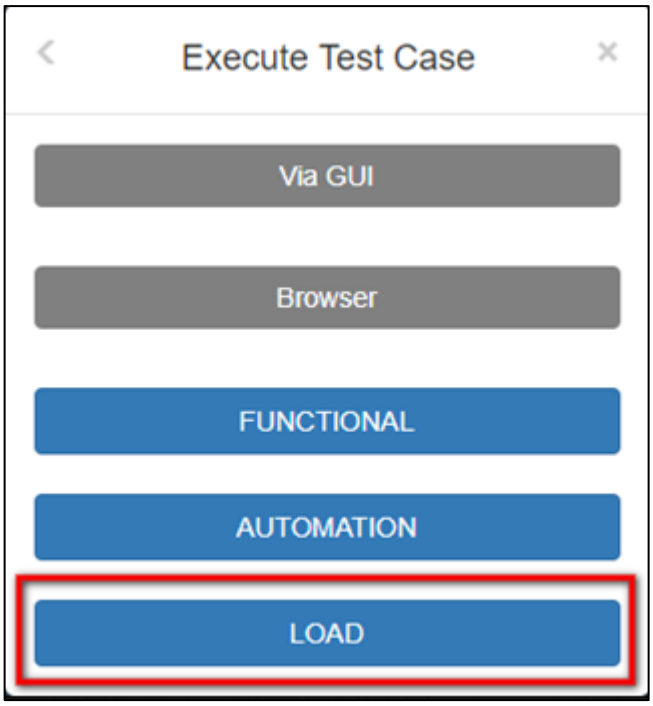
26. On click of the Report button, in the browser, the report appears.



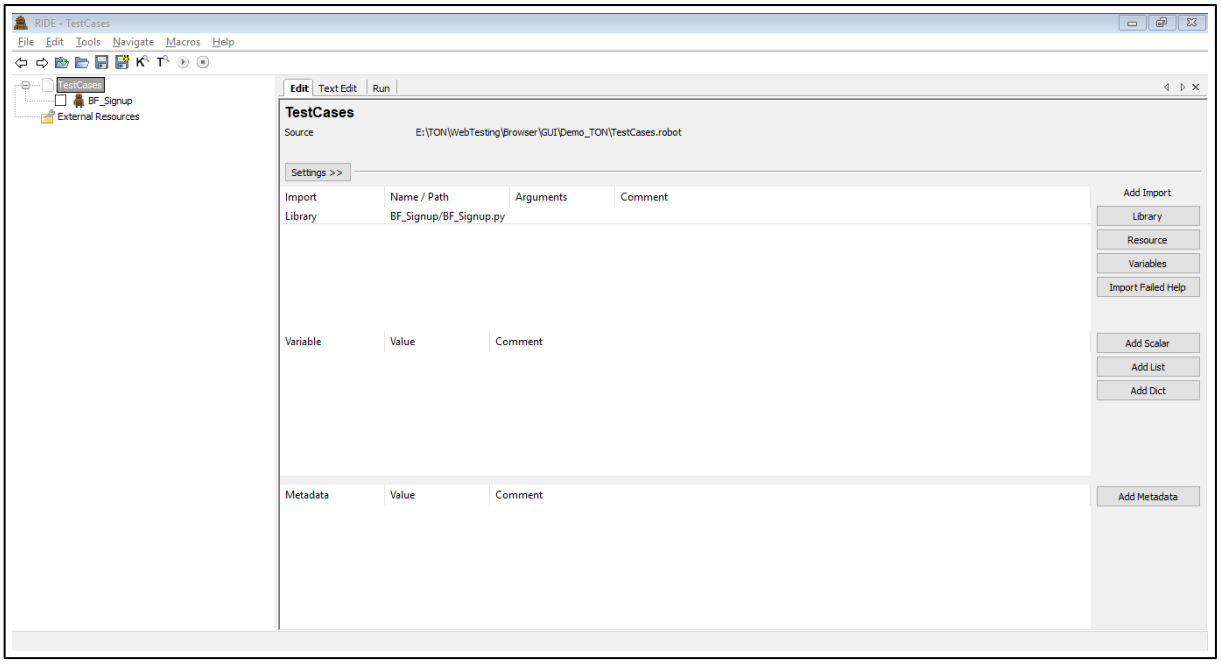
27. In the browser on the top right corner, the log button is there. Clicking on it, will show the log report and vice versa.



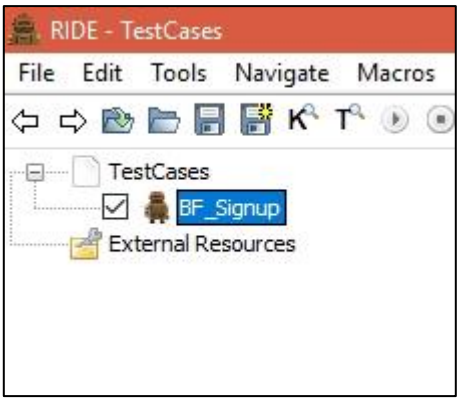
29. To perform load testing of the JSON file downloaded using Kantu browser, click **Load** button.



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

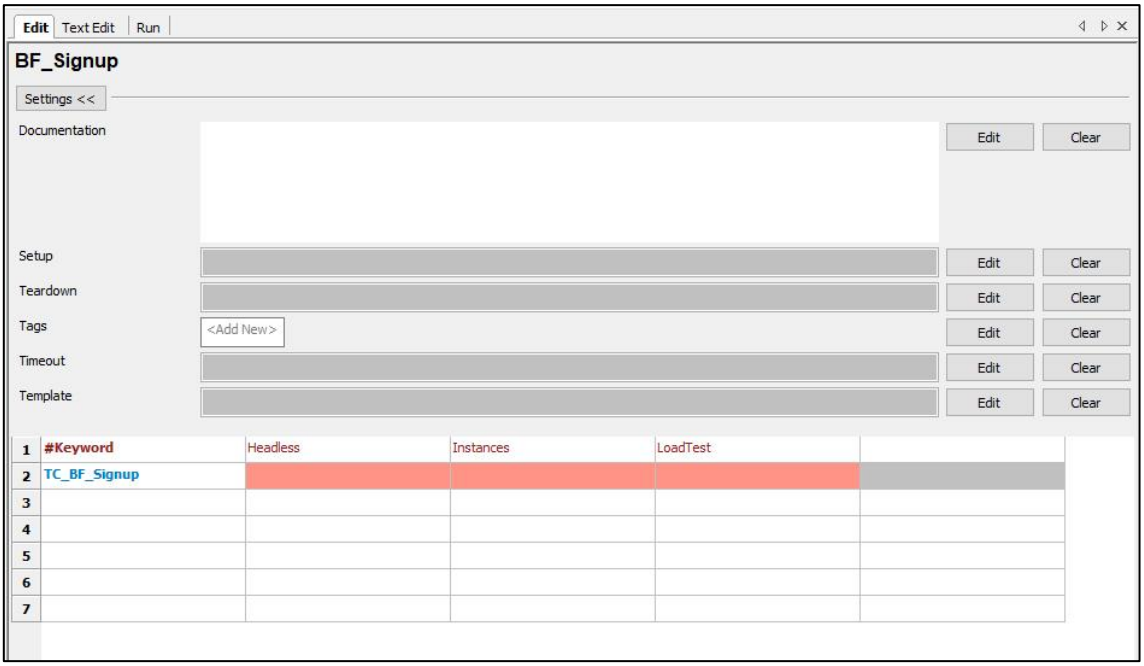


31. On the left panel, select one test case to execute.



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.

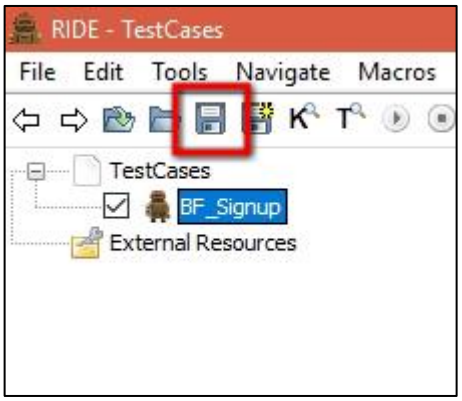
Instances = 0 (Only valid for Load Test)

LoadTest = Yes (Only valid for Load Test)

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

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

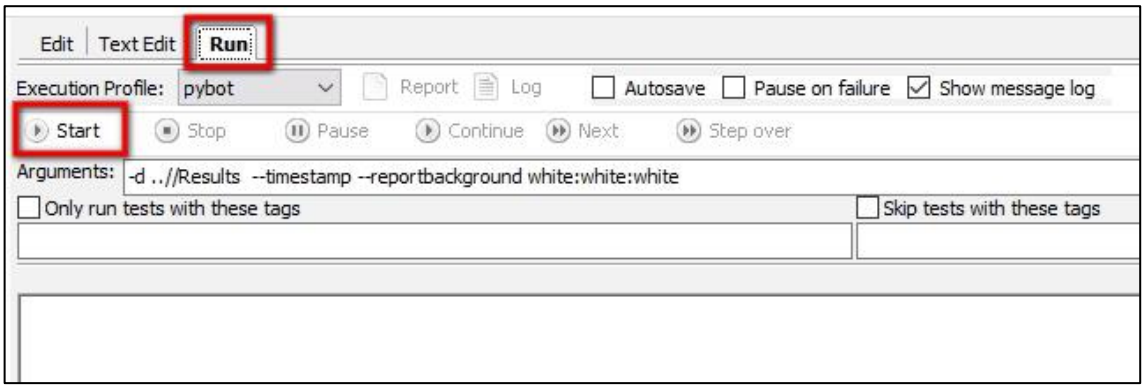
36. Click **Save** button to save the test case.



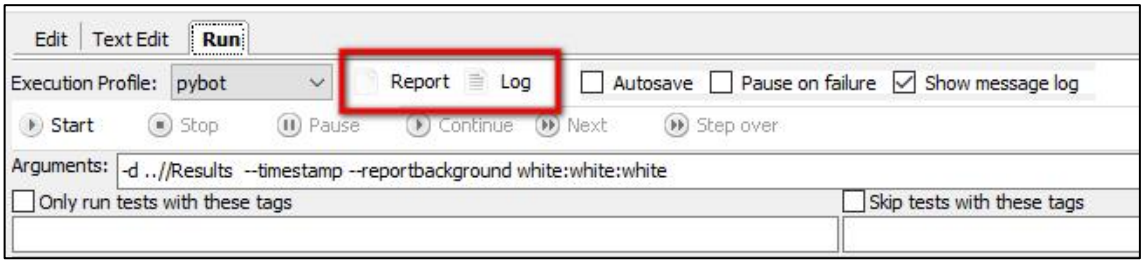
37. Configure the **Arguments** textbox by entering:

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

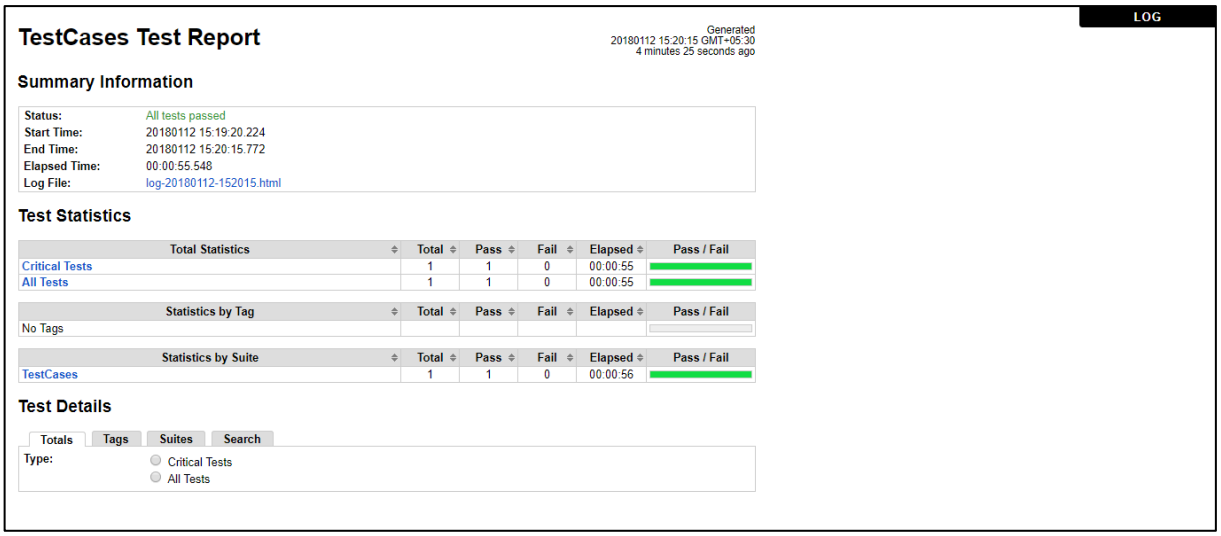
38. Under Run tab, click **Start** button to start the execution.



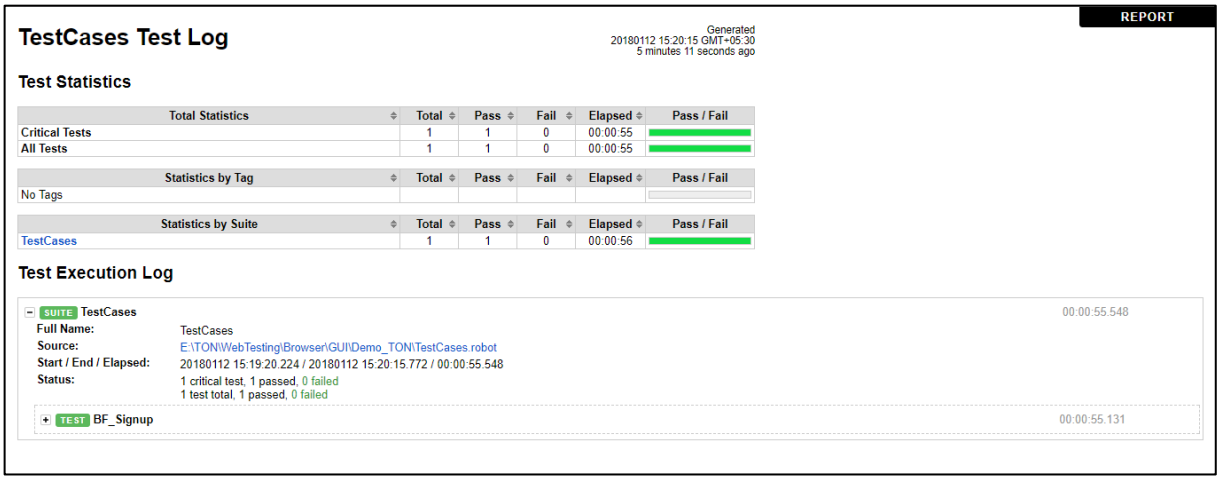
39. Click on the **Log / Report** button to view the result.



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

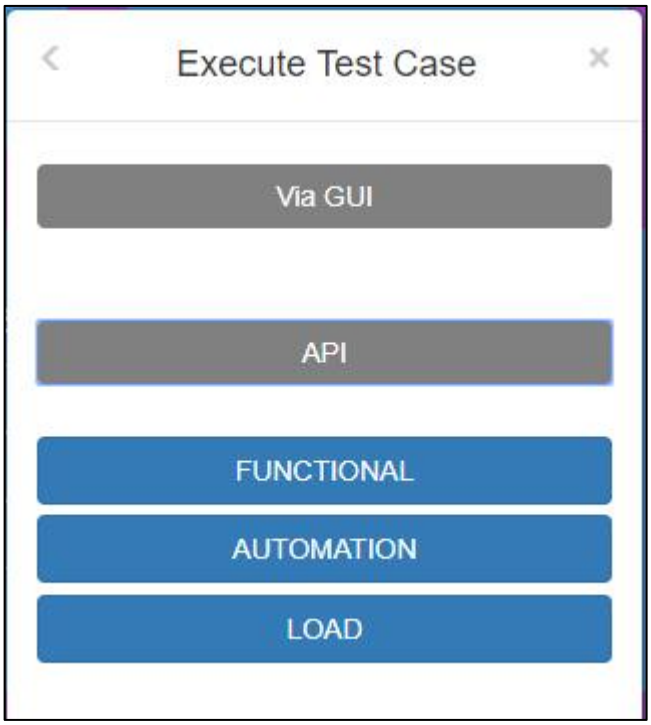


41. In the browser on the top right corner, the log button is there. Clicking on it, will show the log report and vice versa.

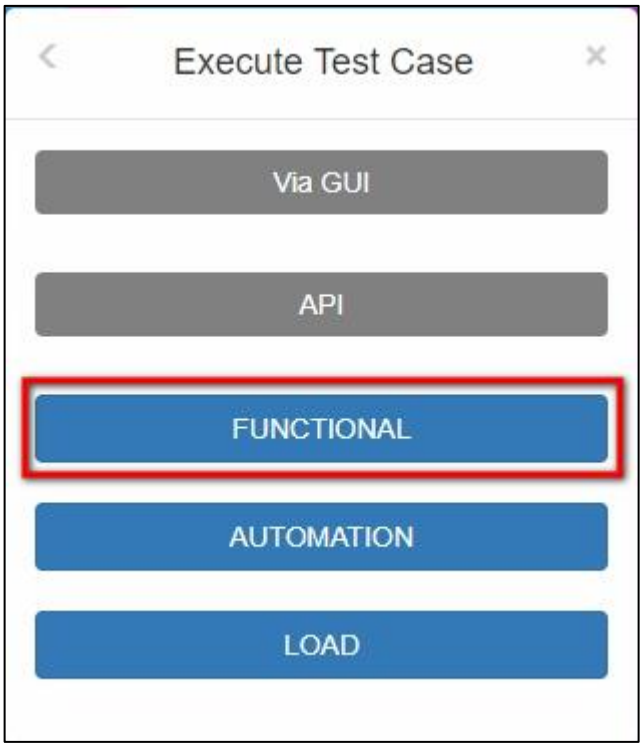


5.3.1.2 EXECUTE – VIA GUI – API

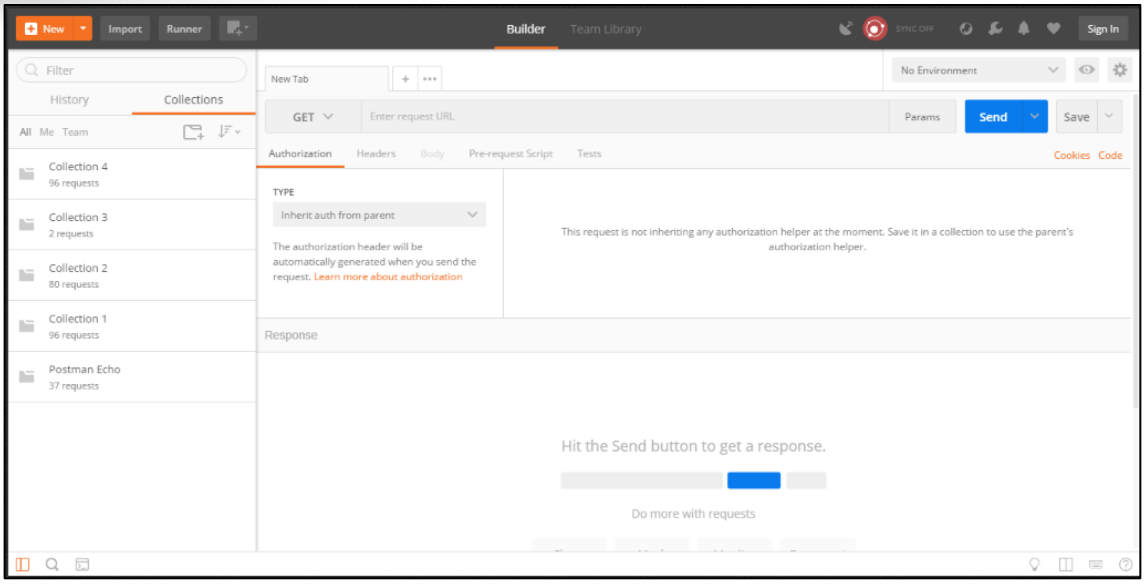
1. On click of the API button, 3 buttons will appear – **Functional, Automation and Load**.



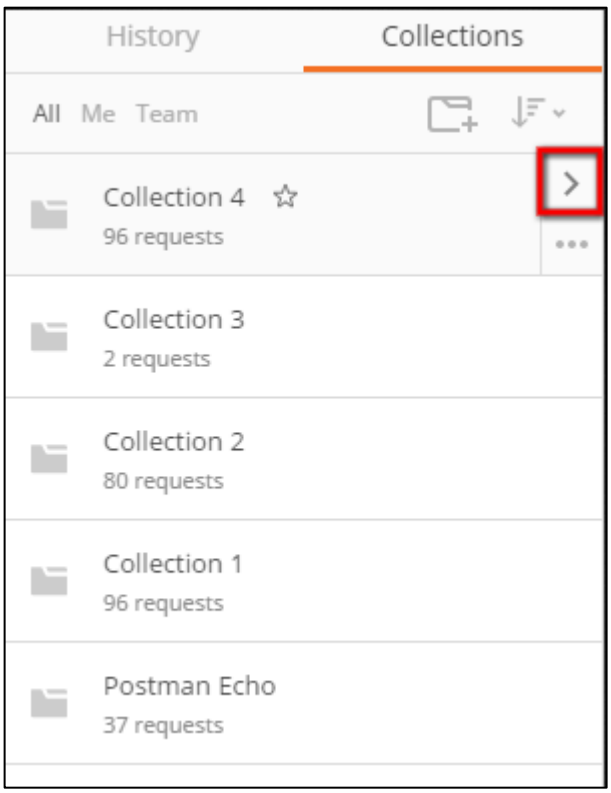
2. To perform functional testing of the recorded scenario, press **Functional** button.



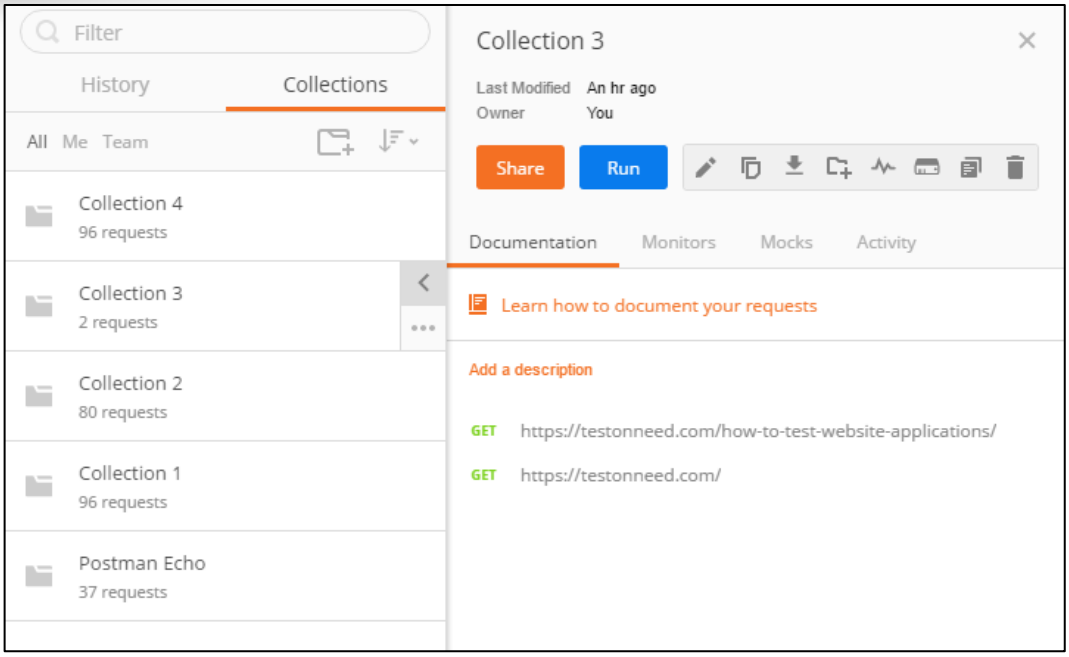
3. On click of the button, Postman will open.



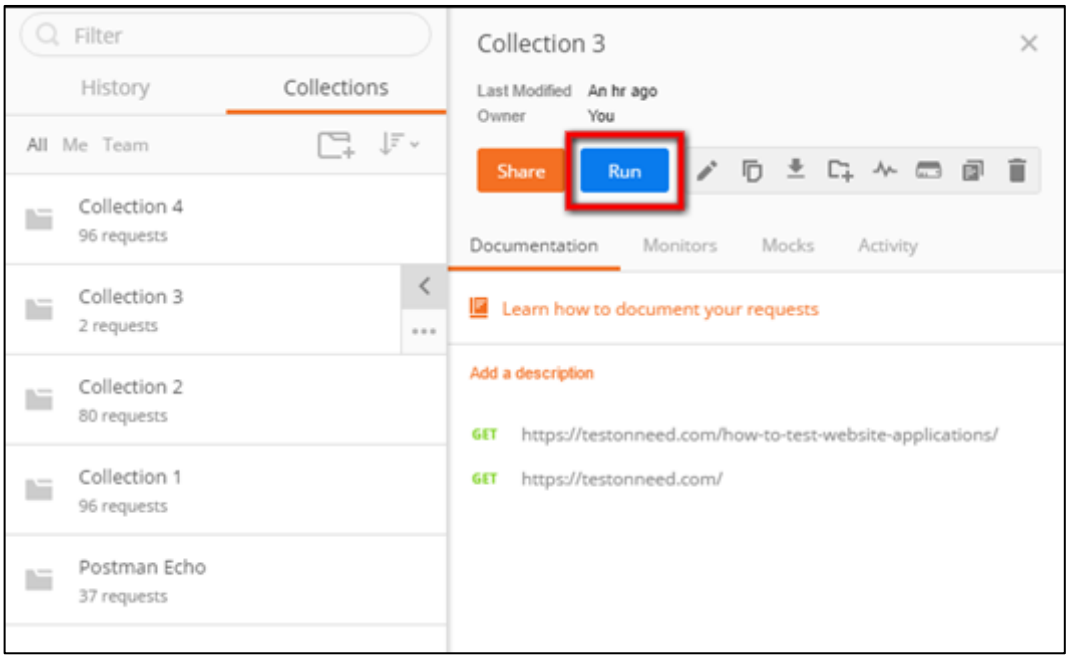
4. On the left, there will be two tabs History and Collections. In the collections tab look for the **Collection** in which the JSON was saved during the **Record Test Case** button.
5. On mouse hover of the **Collection name**, click on the right arrow.



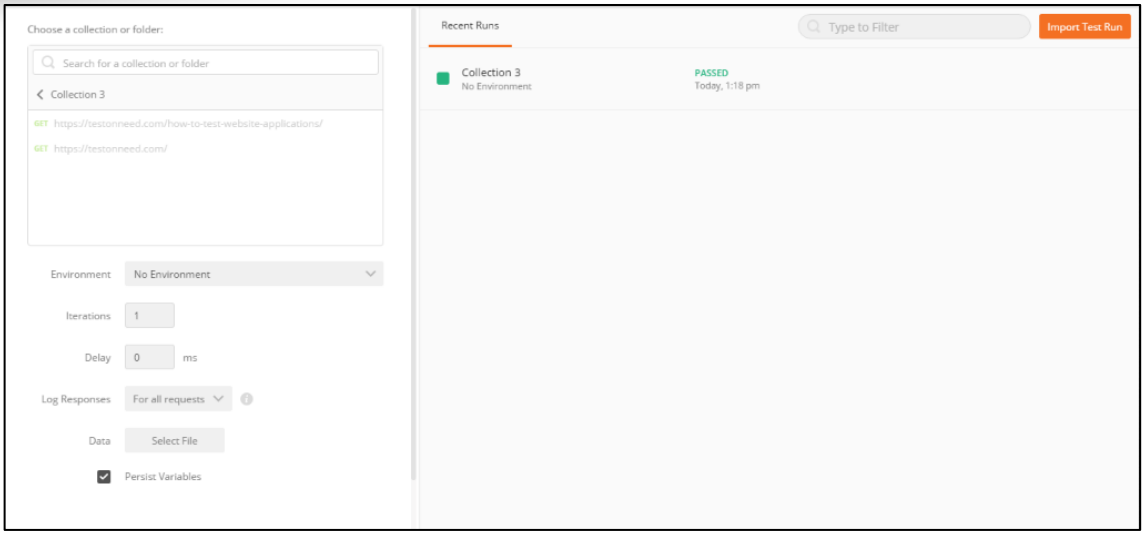
6. On clicking of the arrow button, a small window comes on the right next to the collection selected.



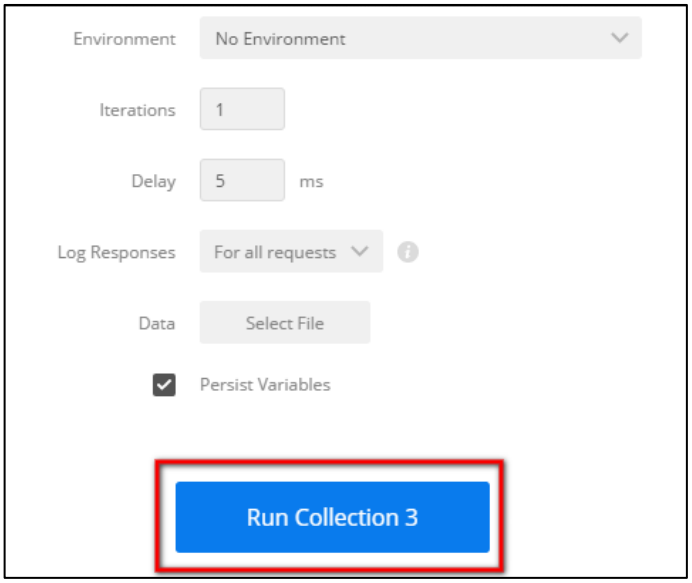
7. From the pop-up, click **Run** button.



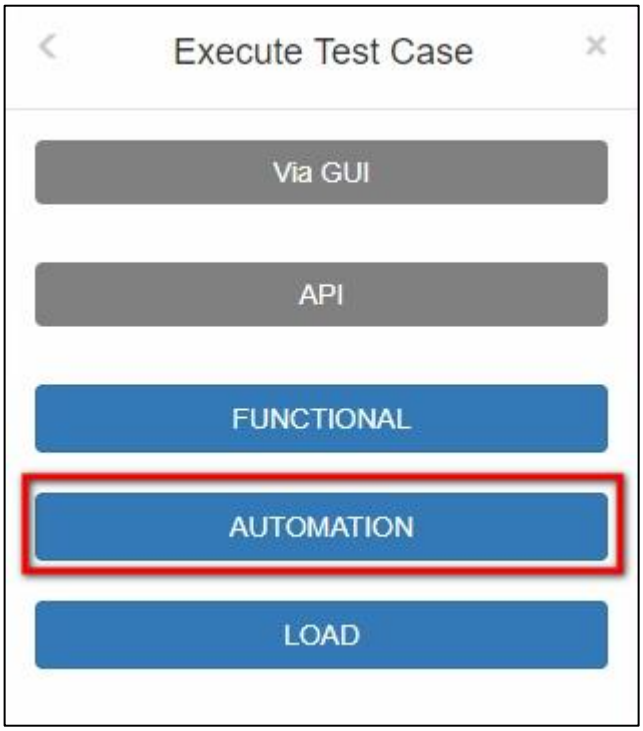
8. On click of **Run** button, a new window will open called **Collection Runner**.



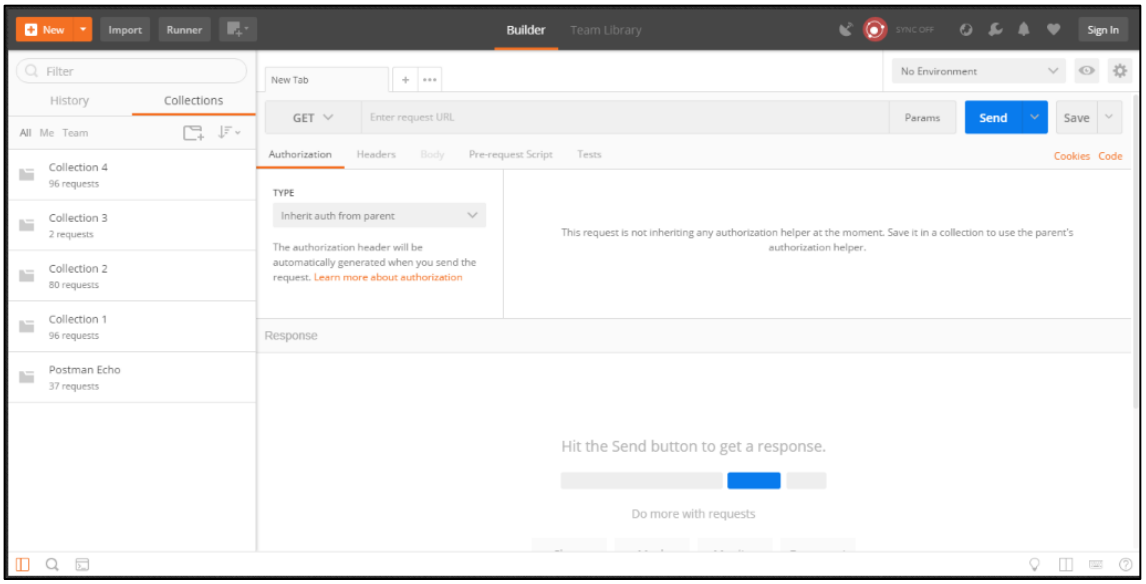
9. In the window, on the left panel, set the parameters and click **Run <Collection_Name>**.



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

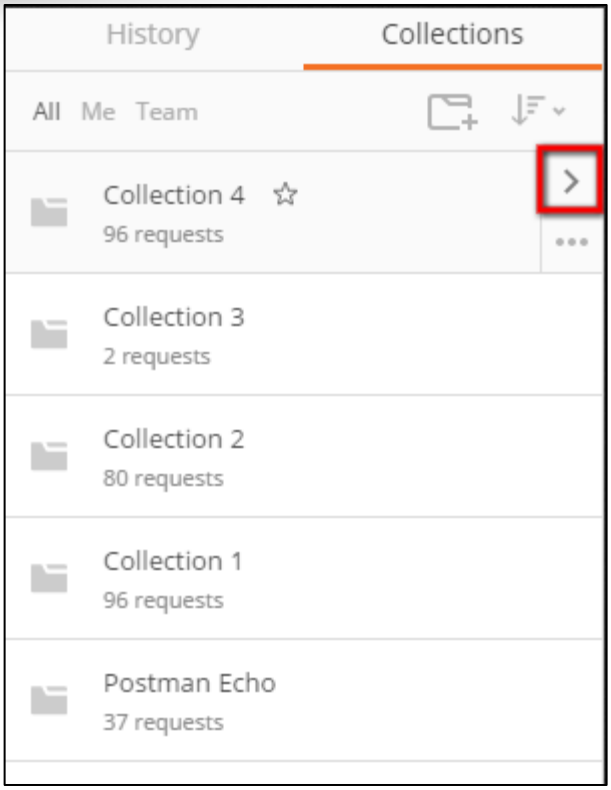


11. On click of the button, Postman will open.

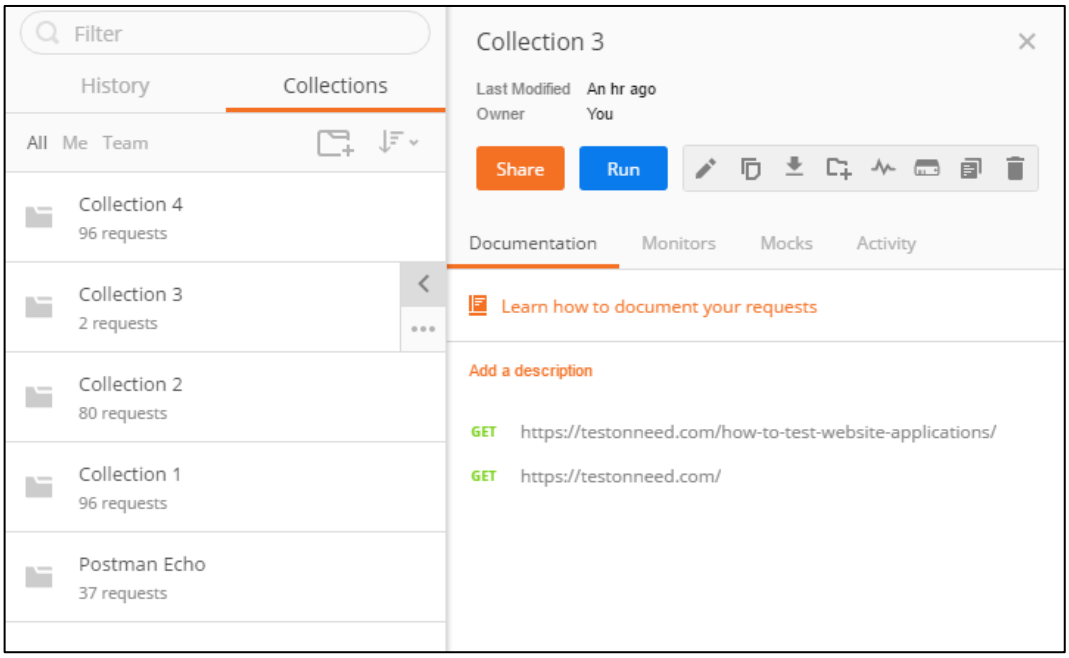


12. On the left, there will be two tabs History and Collections. In the collections tab look for the **Collection** in which the JSON was saved during the **Record Test Case** button.

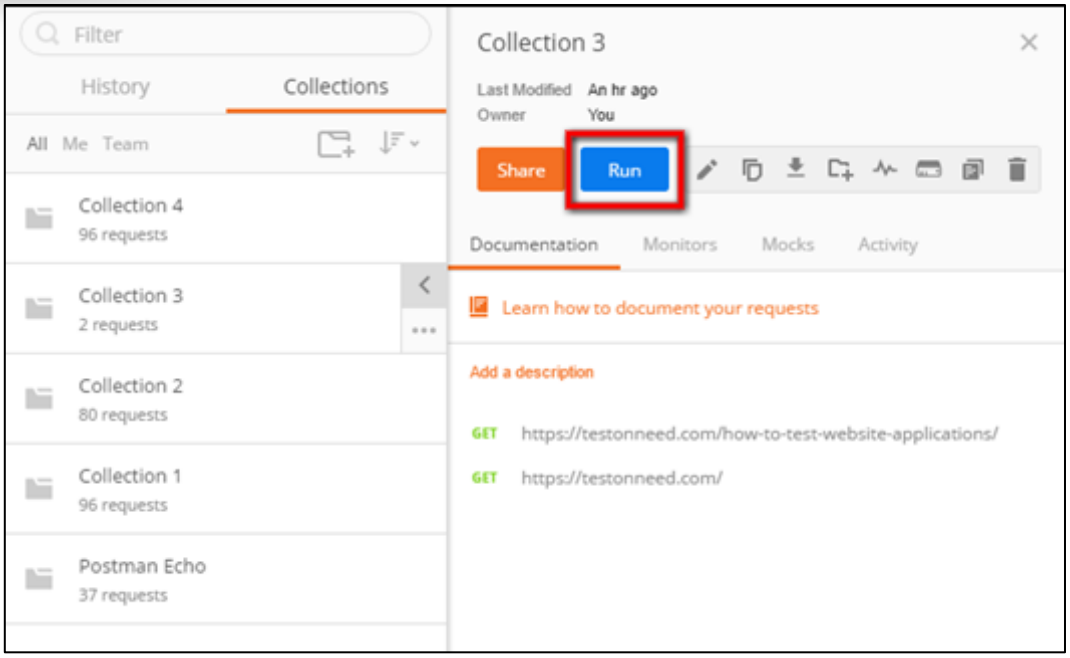
13. On mouse hover of the **Collection name**, click on the right arrow.



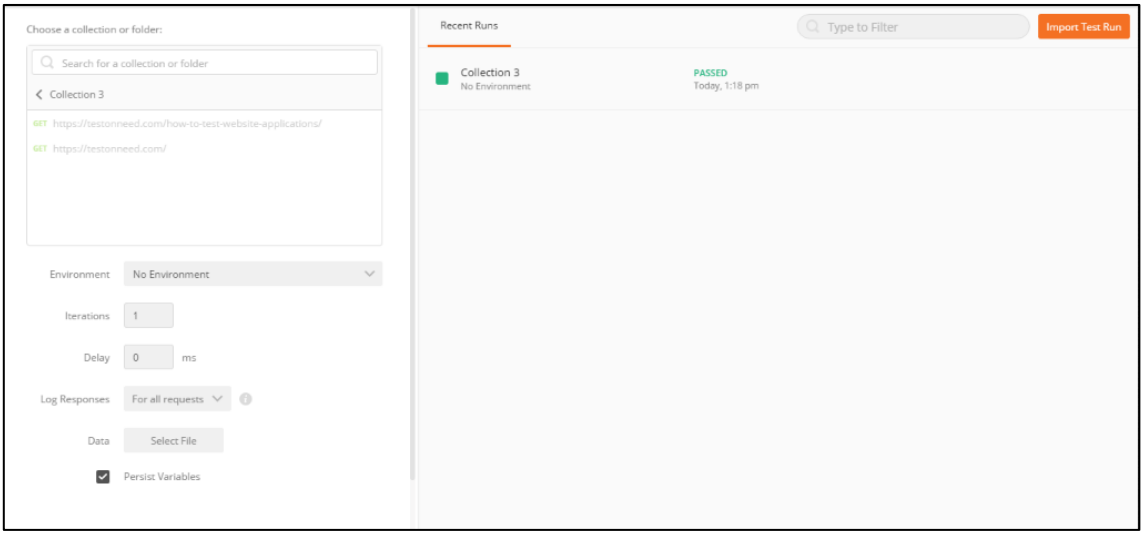
14. On clicking of the arrow button, a small window comes on the right next to the collection selected.



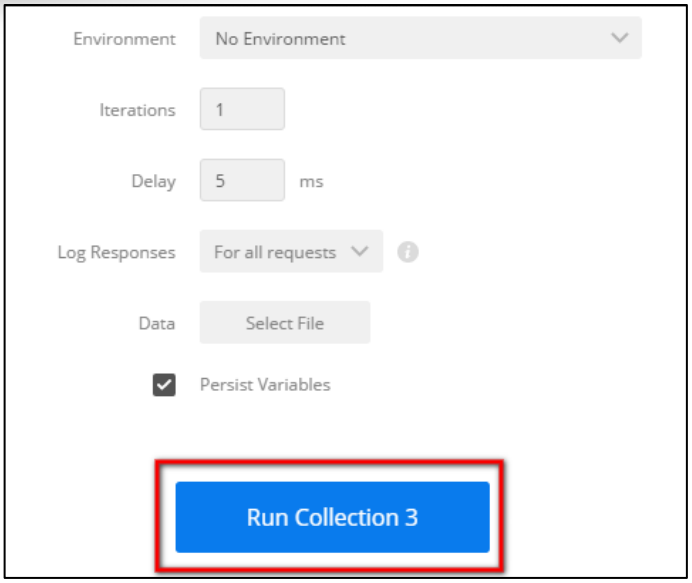
15. From the pop-up, click **Run** button.



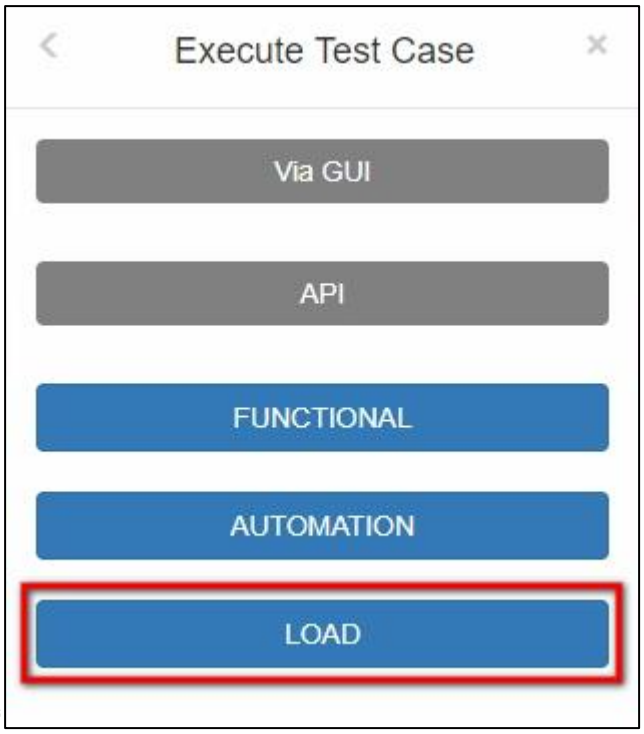
16. On click of **Run** button, a new window will open called **Collection Runner**.



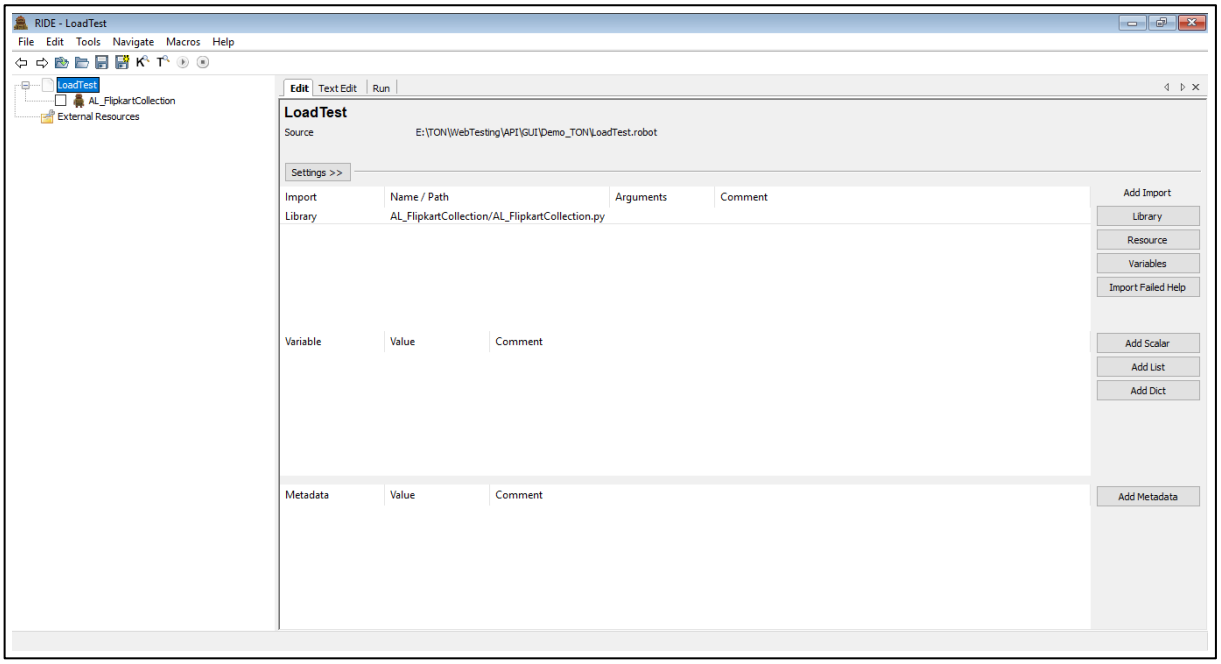
17. In the window, on the left panel, set the parameters and click **Run <Collection_Name>**.


Environment
No Environment
Iterations
1
Delay
5
ms
Log Responses
For all requests
Data
Select File
Persist Variables
Run Collection 3

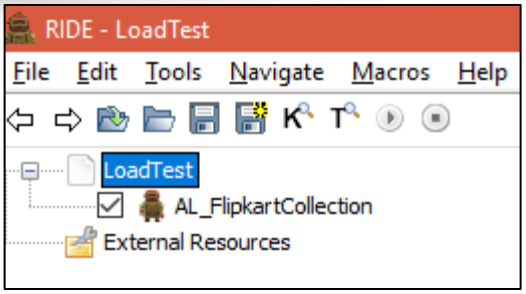
18. To perform load testing of the JSON downloaded using the postman, press **Load** button.



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

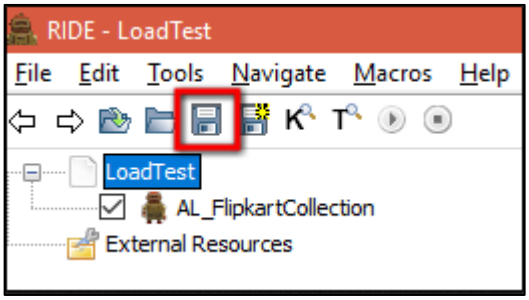


20. On the left panel, select one test case to executed.



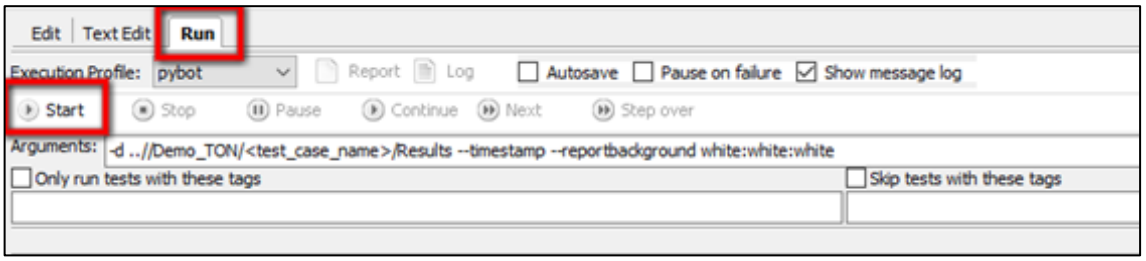
21. To configure the parameter, click on the test case name.
22. On click of the test case name, edit tab changes so that parameters can be configured.
23. The value of the parameters is:

    Iterations, VirtualUsers and RampUP Period.
24. Configure the parameters – **Iterations, VirtualUsers and RampUP Period**.
25. Click **Save** button to save the test case.

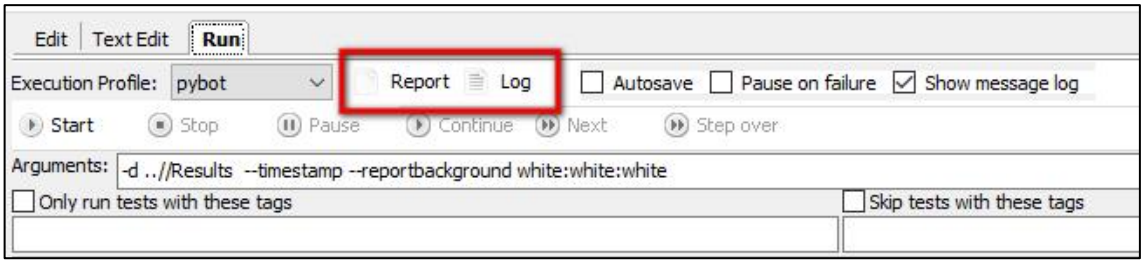


26. Configure the **Arguments** textbox by entering:

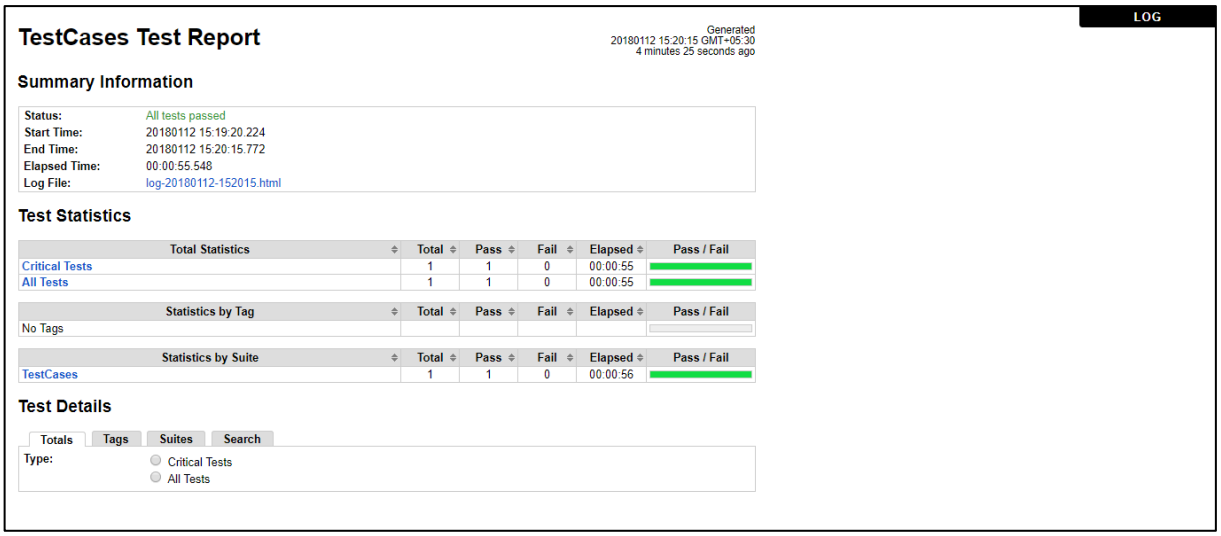
    -d ..\\Demo_TON\<test_case_name>\Results --timestamp --reportbackground white:white:white
27. Under Run tab, click **Start** button to start the execution.



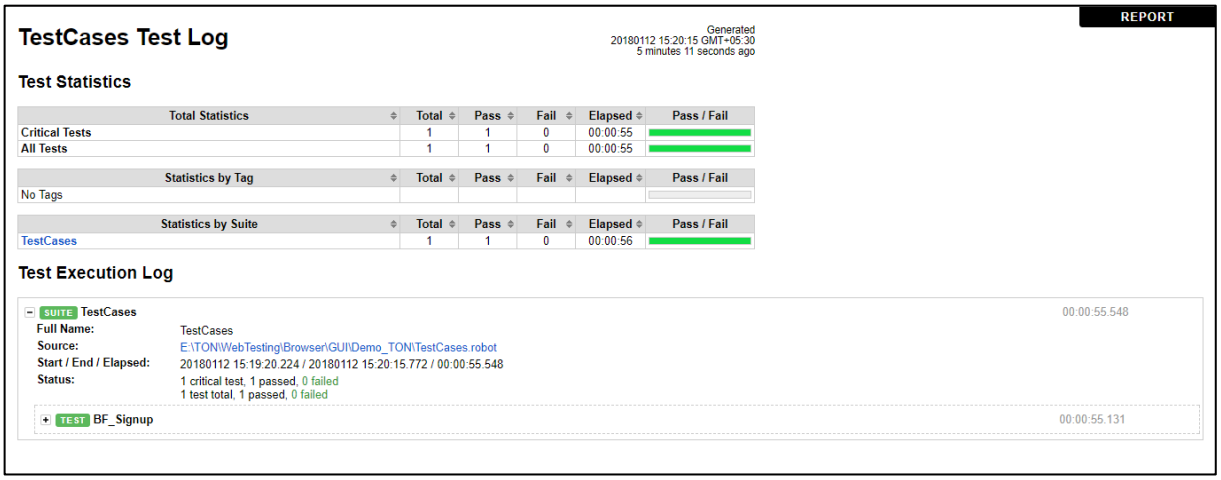
28. Click on the **Log / Report** button to view the result.



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



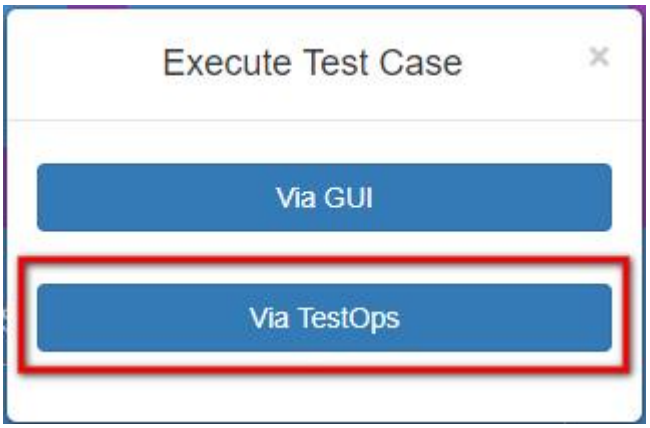
30. In the browser on the top right corner, the log button is there. Clicking on it, will show the log report and vice versa.



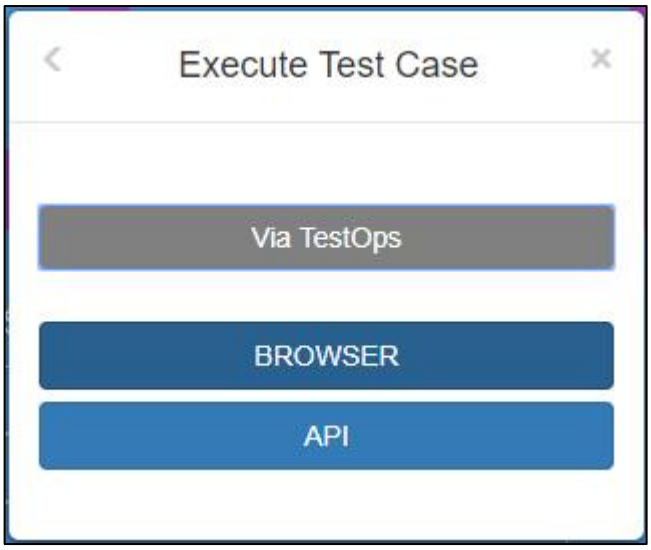
5.3.2 EXECUTE - VIA TESTOPS

Now let's see how to execute the test case using the TestOps option.

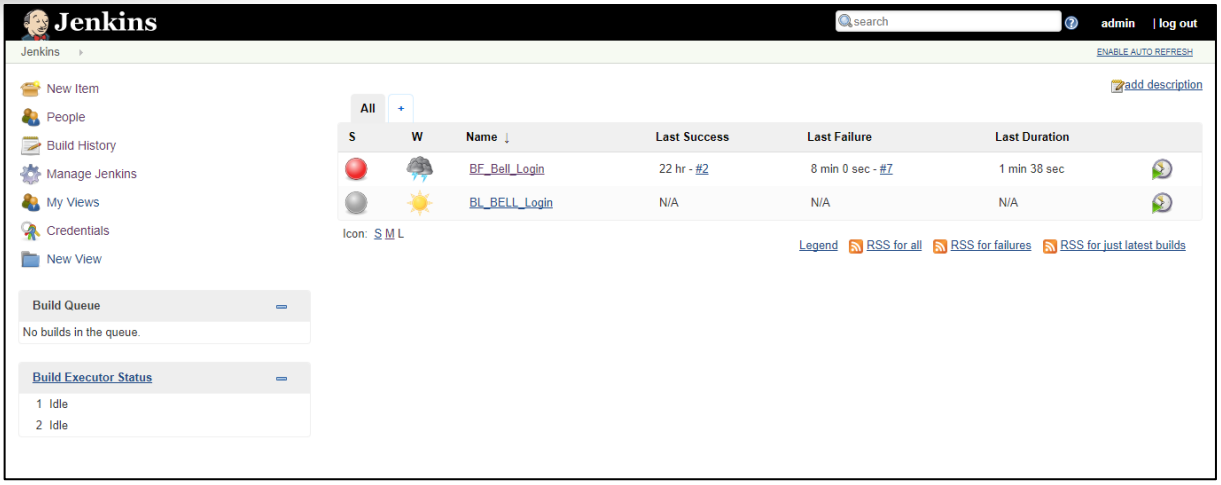
1. Click the Via TestOps button.



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



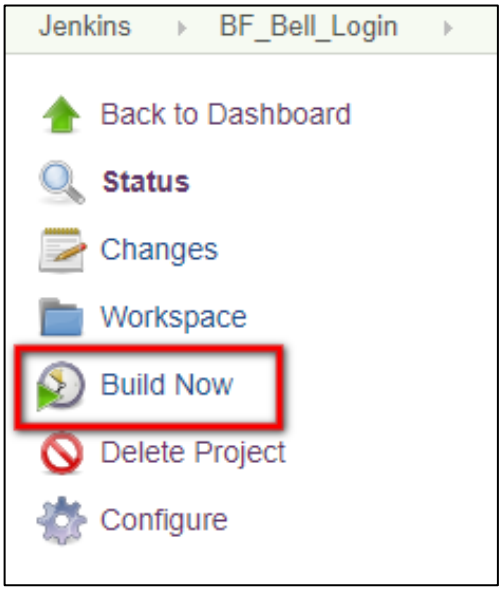
3. On click of any one button, the **Jenkins** will open.



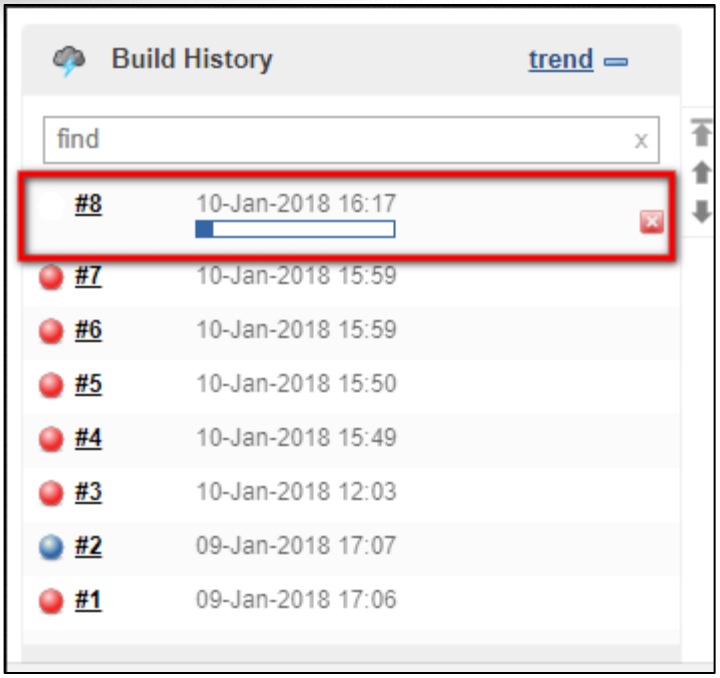
For creation of new View and Job, refer the Reference Section below.

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

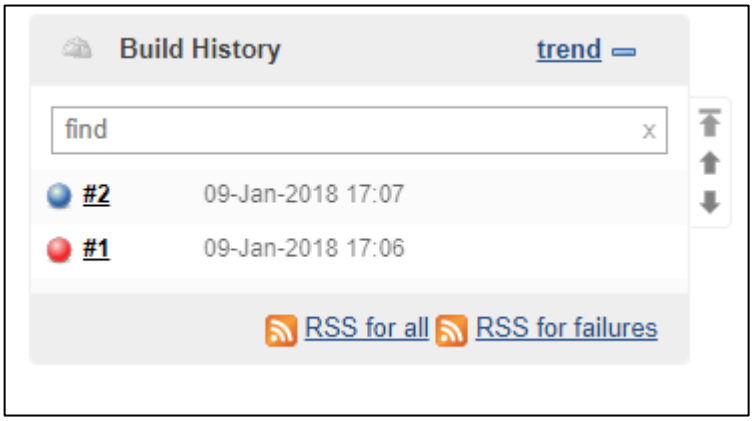
5. On the left side, select Build Now.



6. 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:

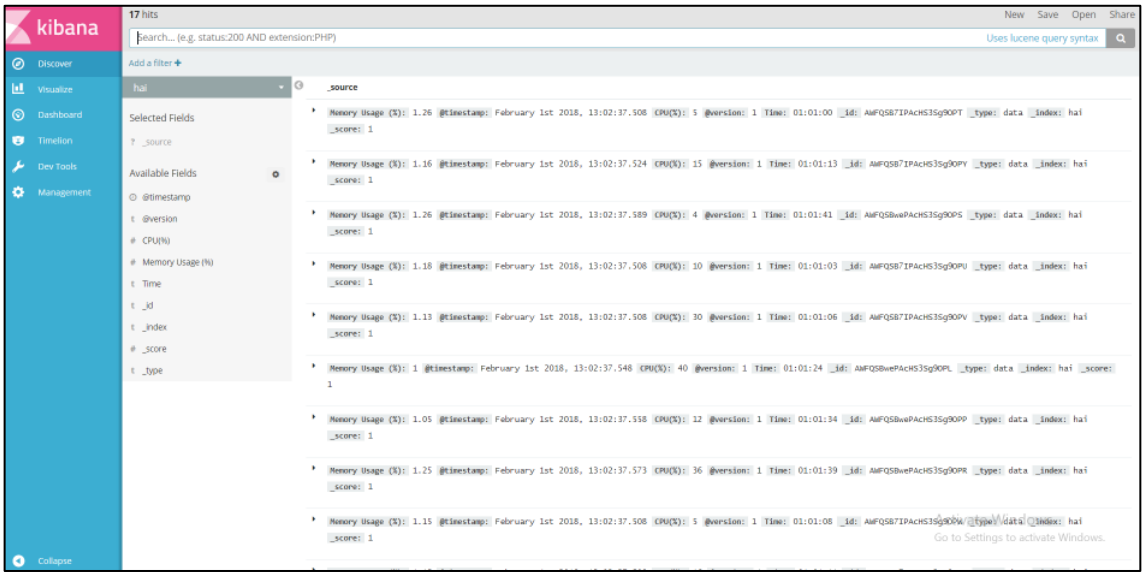
> TON > WebTesting > Browser > TestOps > Demo_TON > <Job_Name_Folder>

5.4 ANALYZE RESULTS

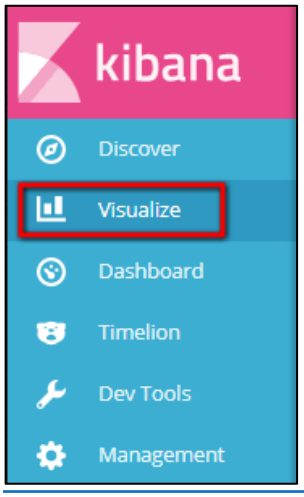
The **Analyze Results** button is used to analyze the final result of the test case that had been recorded and executed in the previous steps. The final result will be displayed in the form of graphs and charts.

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

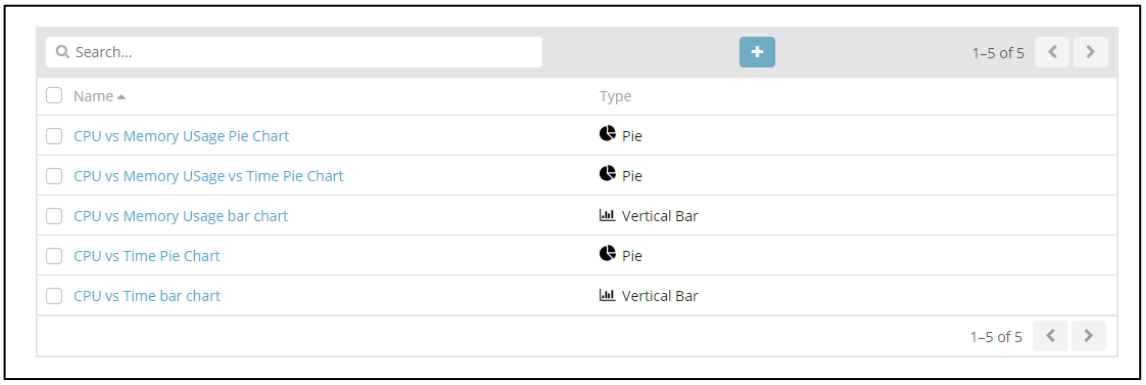
2. On clicking on the **Analyze Results** button, Kibana opens up in a new window.



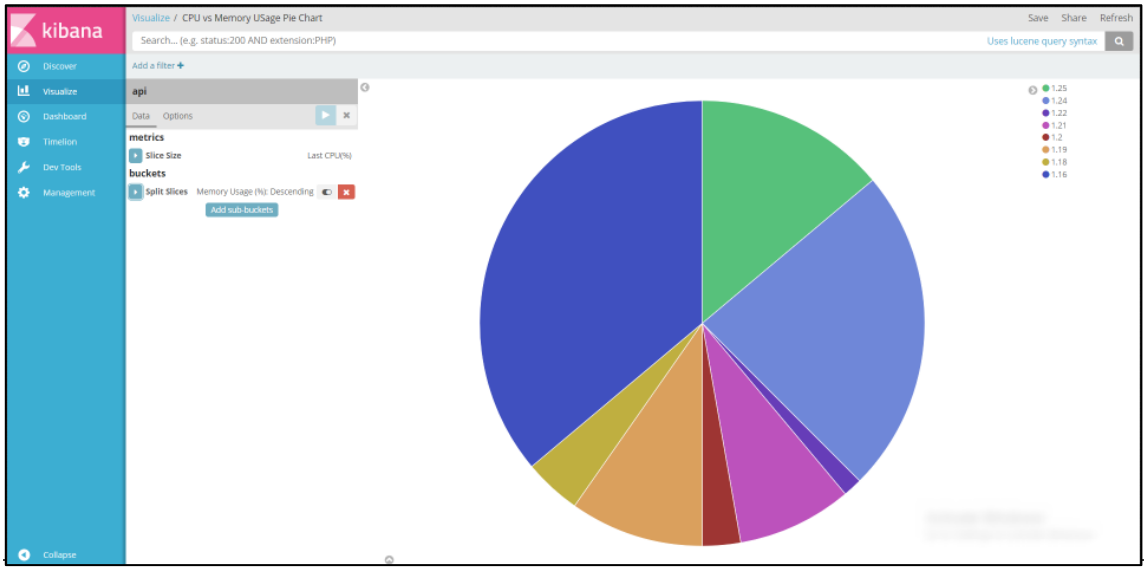
3. On the left panel, click Visualize.



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



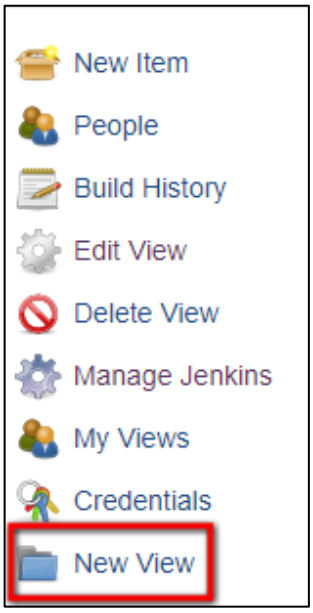
5. On click of the chart name, the user can see the graph based on the type of the chart. (The example shows for Pie Chart)



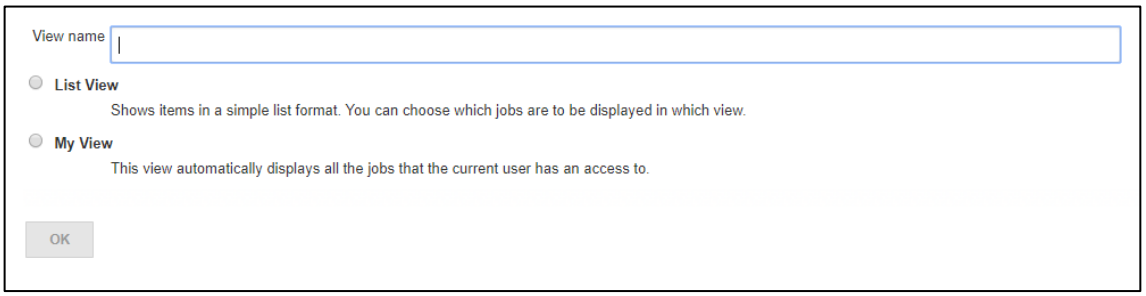
6. REFERENCES

6.1 CREATION OF NEW VIEW

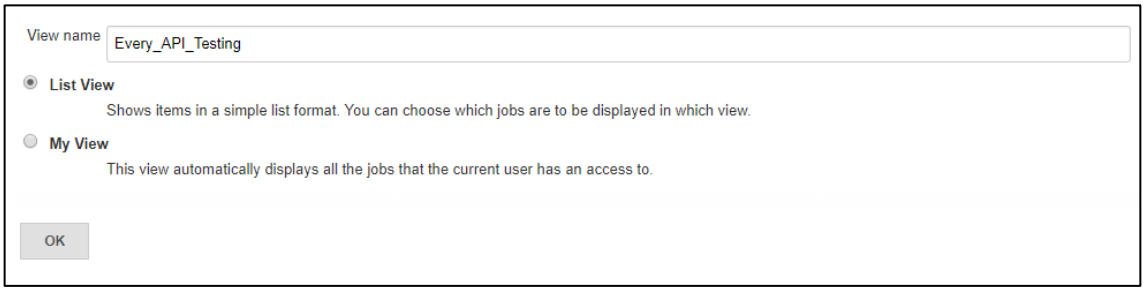
- To create a new view, on the left panel, click **New View**.



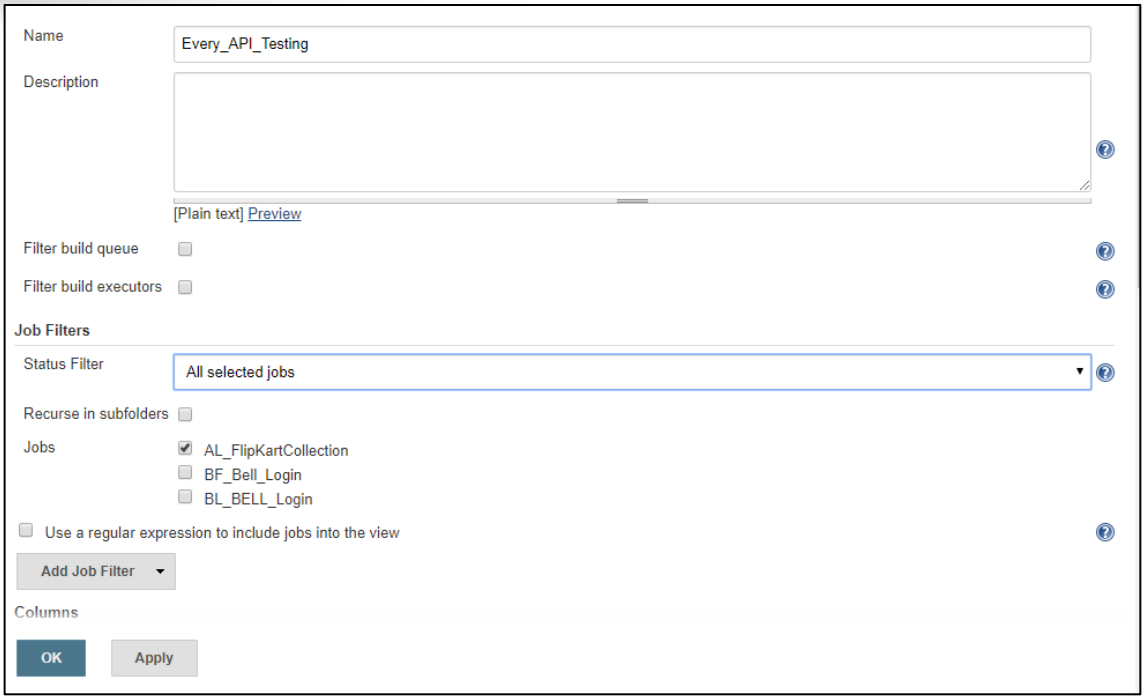
- On clicking, give the name of the tab as **Every_API_Testing** and select the type of view.



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



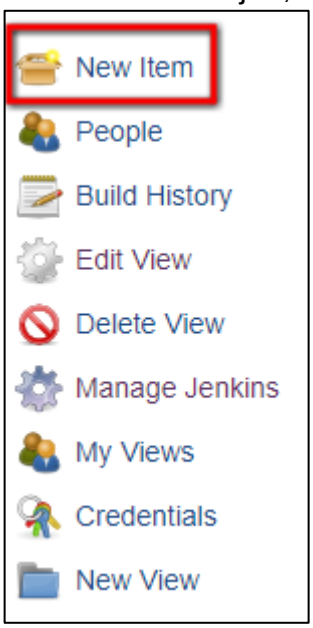
- In the next page, select the jobs that are to be added to this tab.



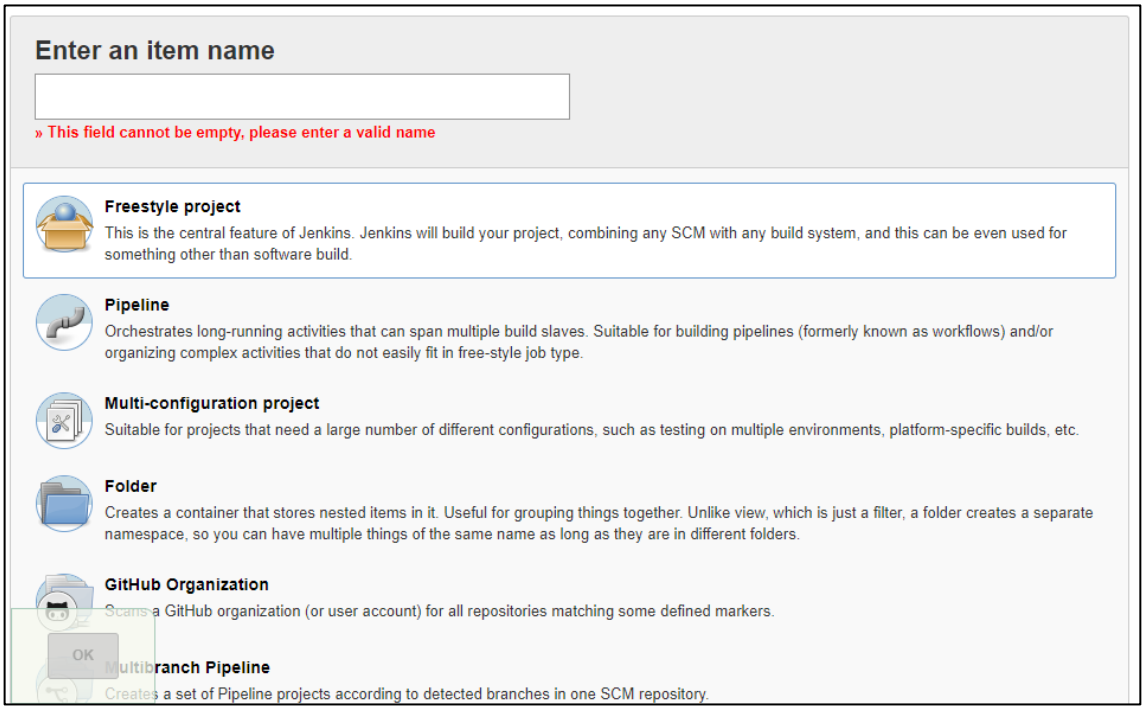
- Once done, click Apply and Ok to save and Close the creation.

6.2 CREATION OF NEW JOB

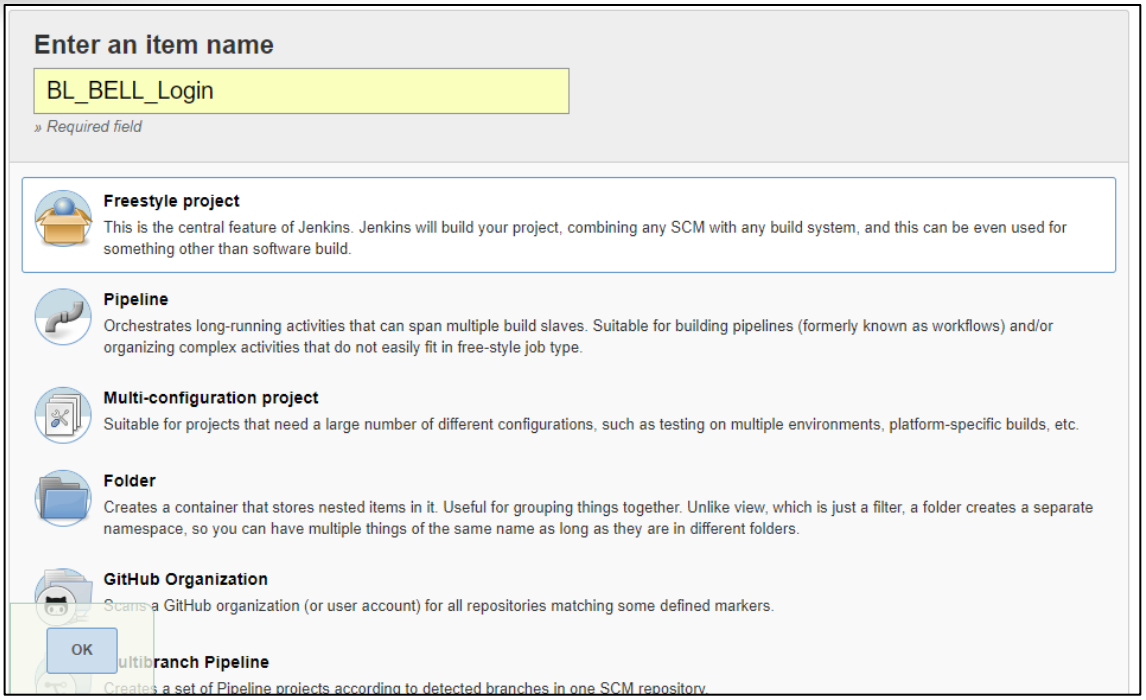
- To create a new job, on the left panel, click **New Item**.



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

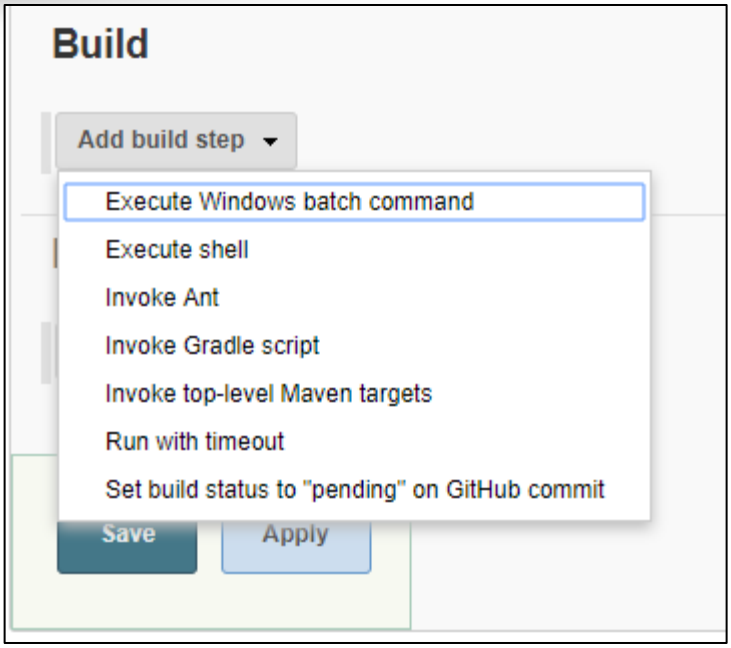


- Then select the type of the job that is to be created.
- Click Ok to save the job.



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

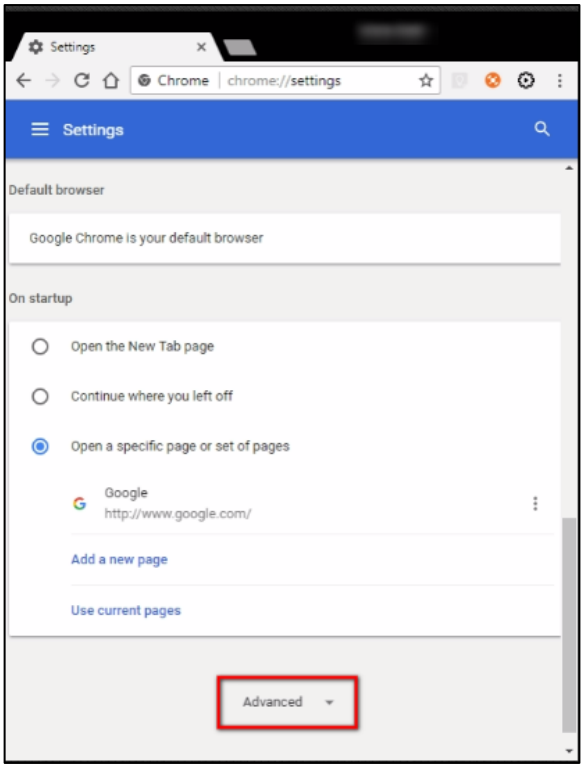
- Under build, click Add Build Setup drop down.



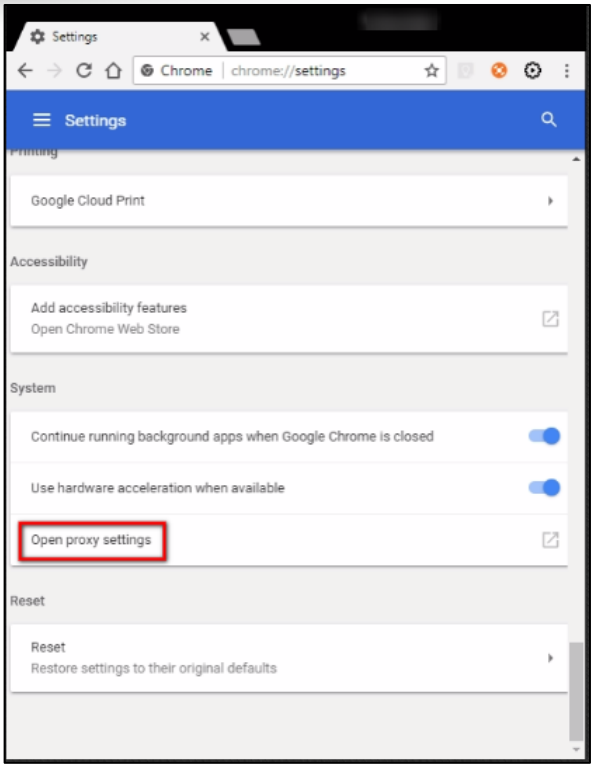
- From the drop down, select **Execute Windows Batch Command.**
- On selecting, a text area appears.

6.3 SETTING UP OF PROXY IN CHROME

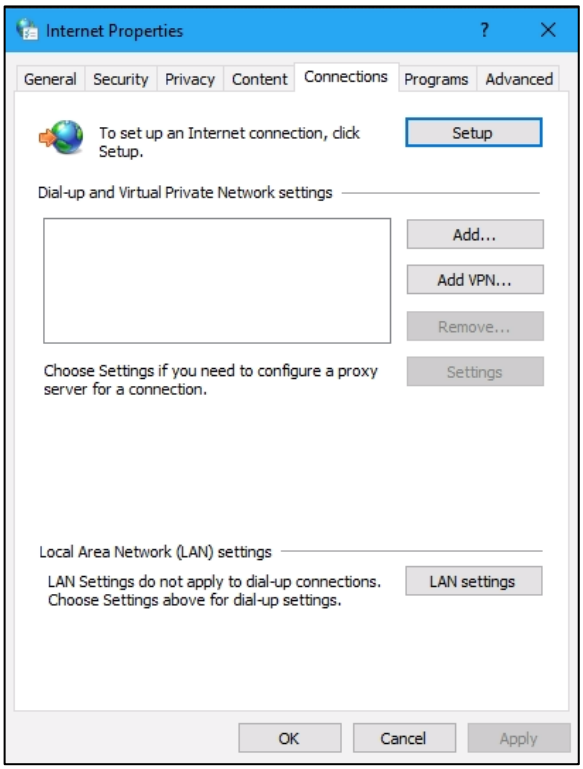
- In the chrome, open Chrome settings.
- Scroll down and click Advanced.



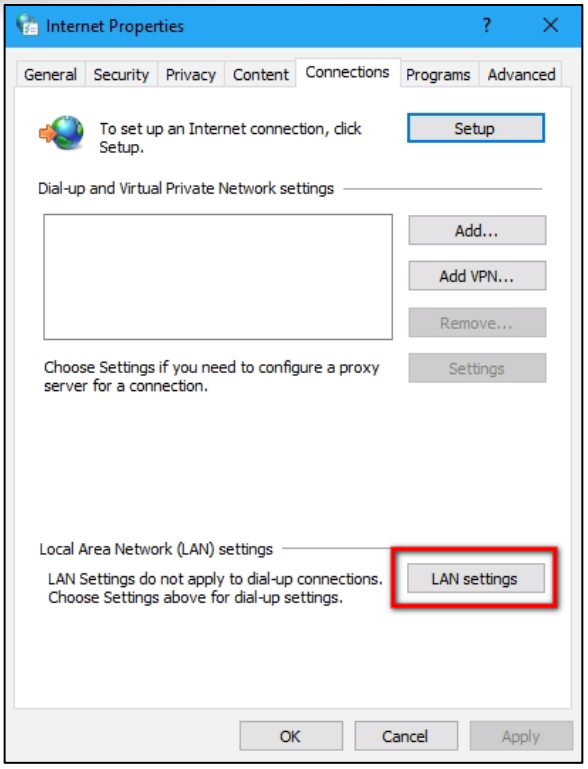
- Again, scroll down and click “Open Proxy Settings”.



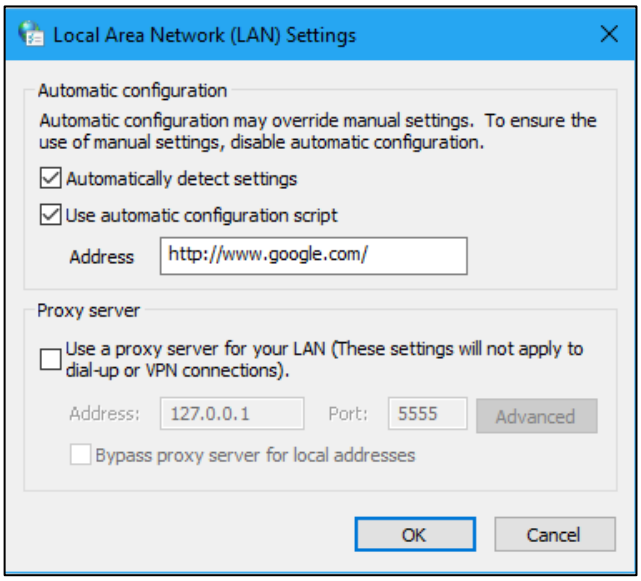
- A dialog box will appear.



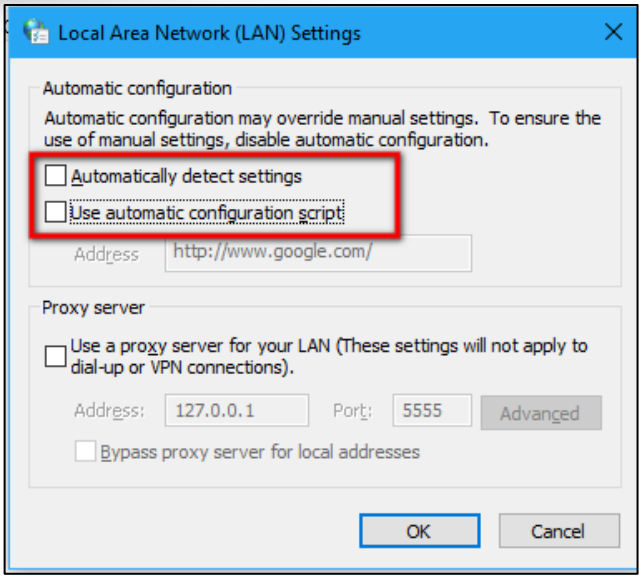
- In the dialog box, click LAN Settings.



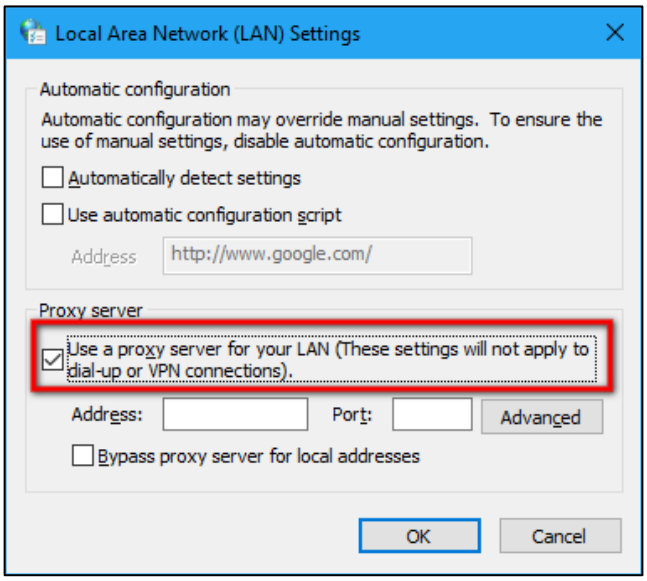
- It will display another dialog box.



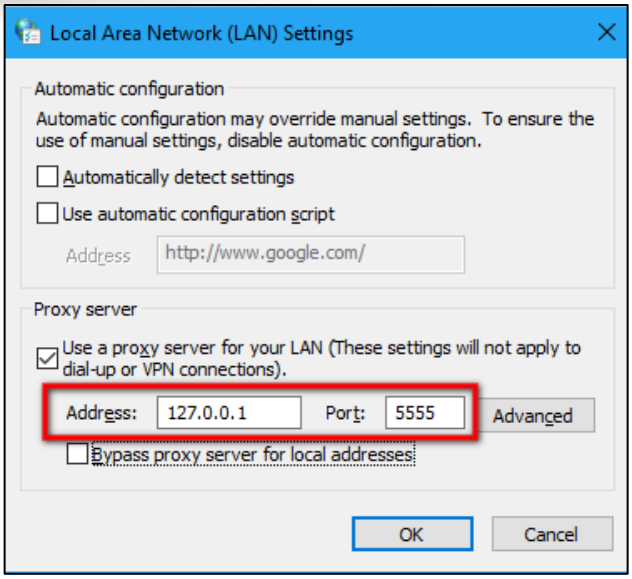
- In the dialog box, uncheck the Automatically detect settings and Use automatic configuration script under Automatic configuration.



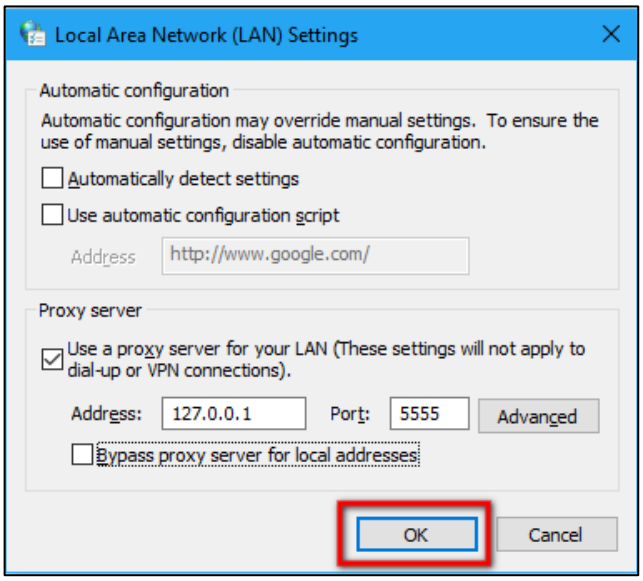
- In the same dialog box, check the box under Proxy server.



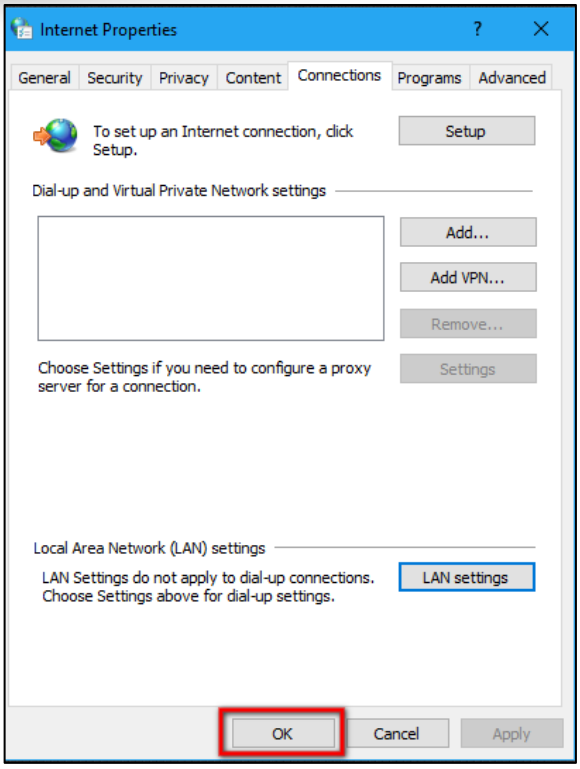
- On checking the box, the Address and Port text boxes becomes active.
- In the address text box, enter the ip address as “127.0.0.1" and in the port text box, enter the port number as “5555”.



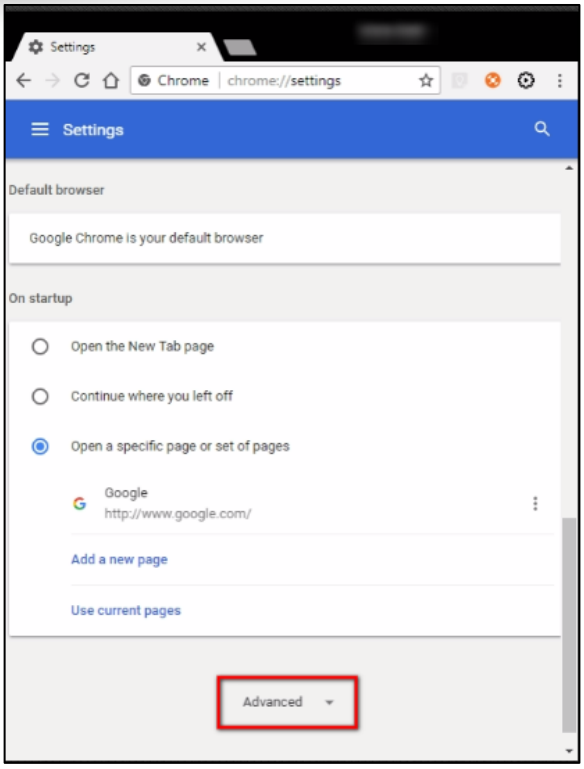
- After entering the address and port number, click Ok.



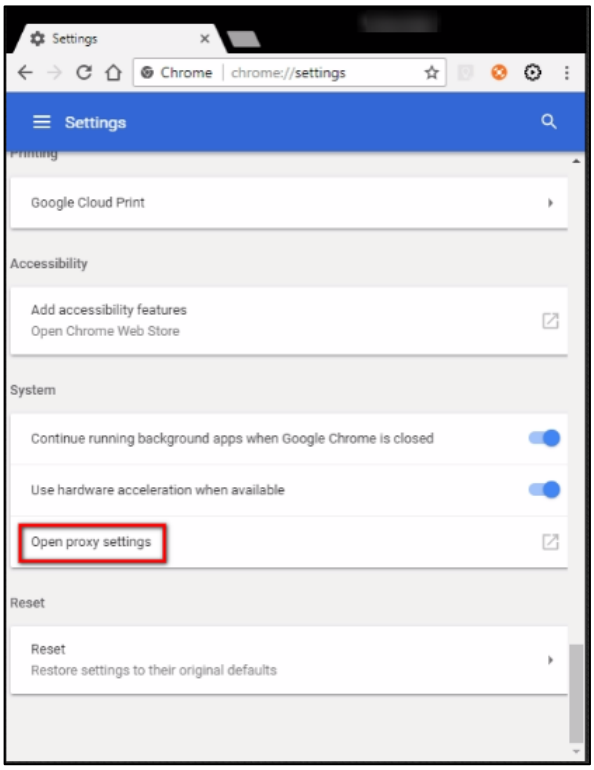
- Again, in the internet properties dialog box, click Ok.



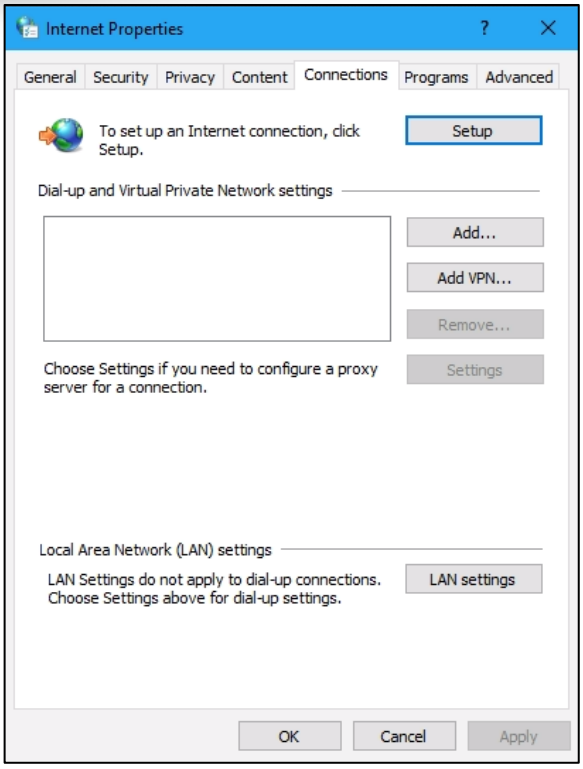
- When the user has finished the recording, the user has to revert back the changes in order to connect to the internet.
- To connect the revert the changes back, open Chrome Settings.
- Scroll down and click Advanced.



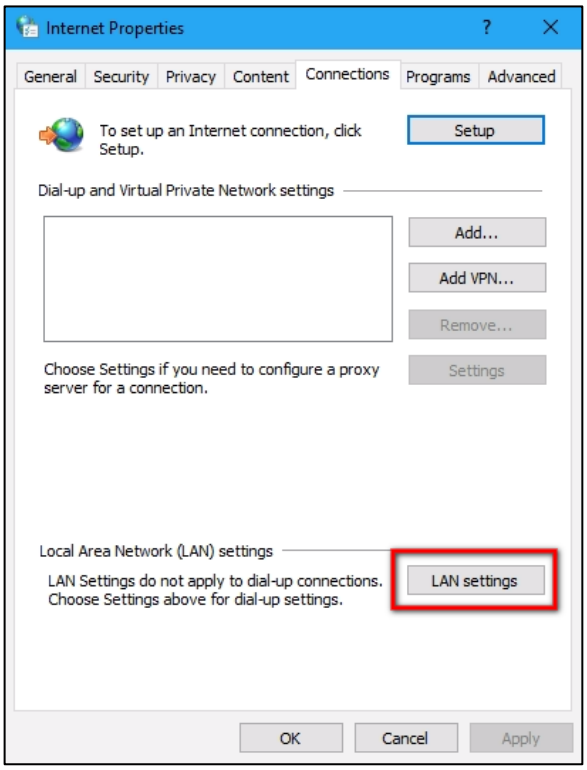
- Again, scroll down and click “Open Proxy Settings”.



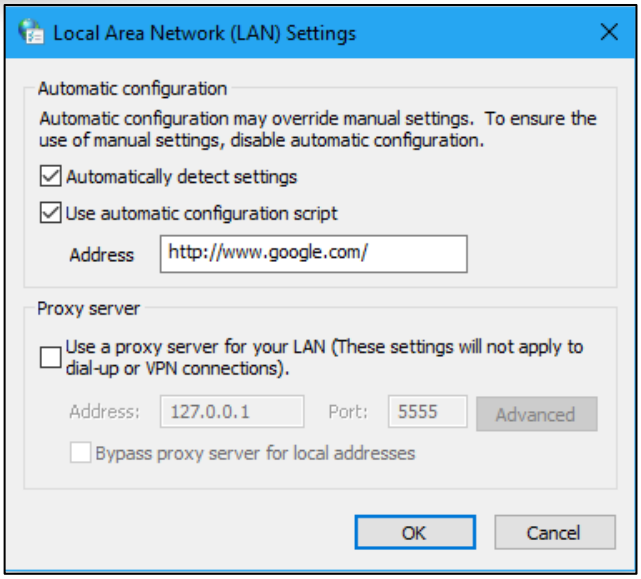
- A dialog box will appear.



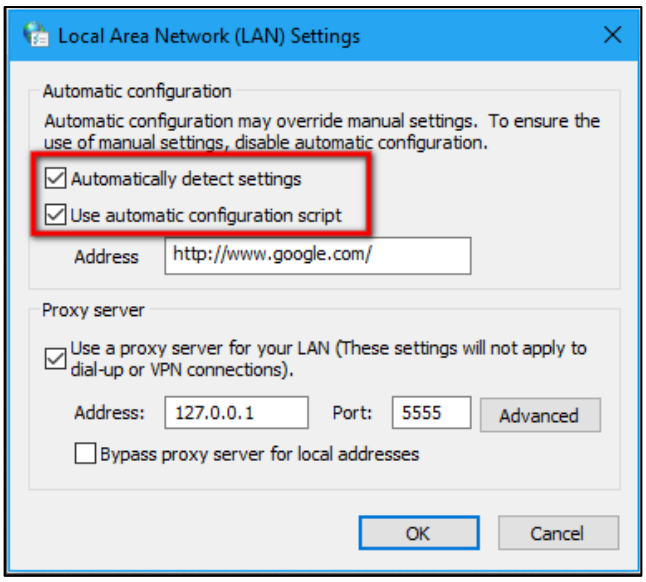
- In the dialog box, click LAN Settings.



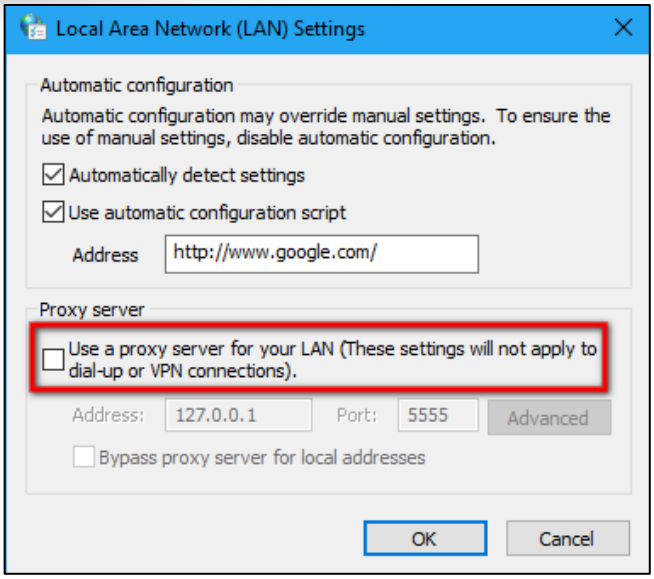
- It will display another dialog box.



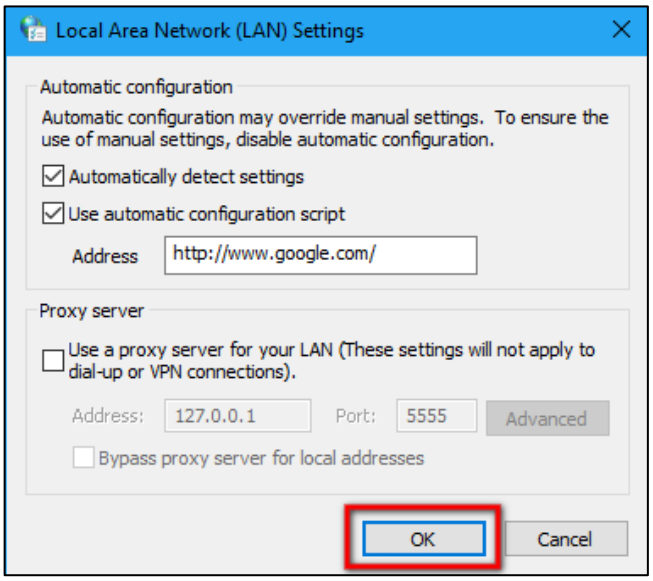
- In the dialog box, check the Automatically detect settings and Use automatic configuration script under Automatic configuration which will be unchecked earlier.



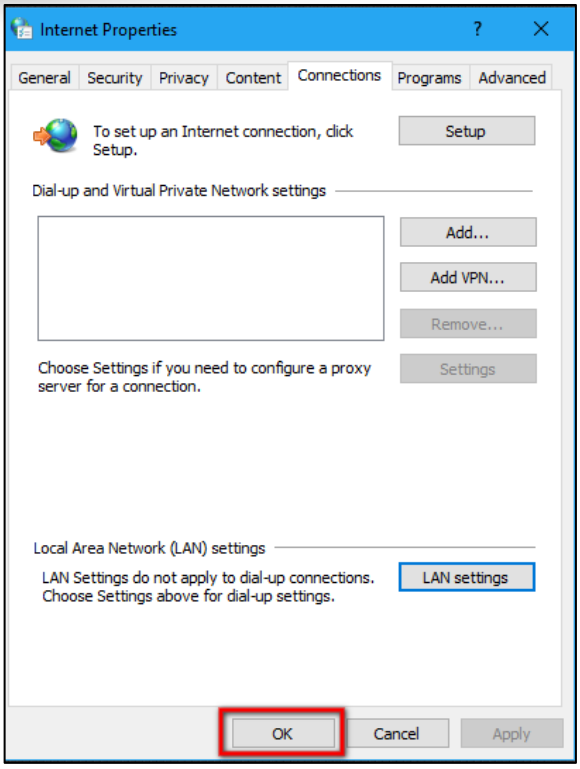
- In the same dialog box, uncheck the box under Proxy server which will be checked earlier.



- After checking and unchecking the boxes, click Ok.

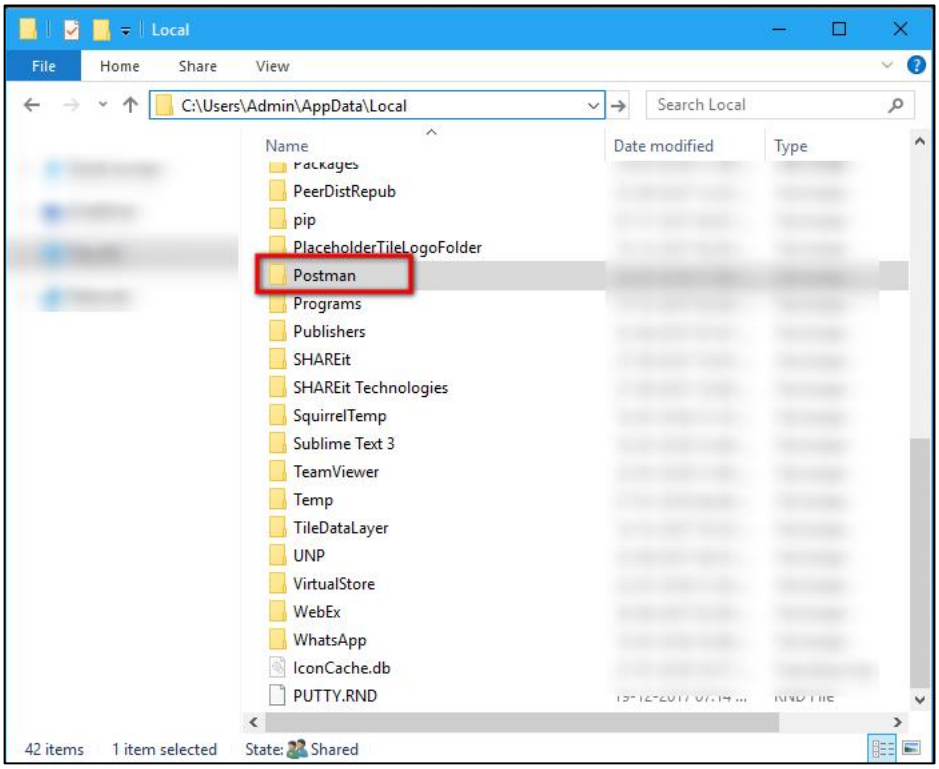


- Again, in the internet properties dialog box, click Ok.


Internet Properties
General
Security
Privacy
Content
Connections
Programs
Advanced
To set up an Internet connection, click Setup.
Setup
Dial-up and Virtual Private Network settings
Add...
Add VPN...
Remove...
Choose Settings if you need to configure a proxy server for a connection.
Settings
Local Area Network (LAN) settings
LAN Settings do not apply to dial-up connections. Choose Settings above for dial-up settings.
LAN settings
OK
Cancel
Apply

6.4 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.