



# || **EVERYAPI – USER MANUAL**



Contact:

[sales@testonneed.com](mailto:sales@testonneed.com)

## 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

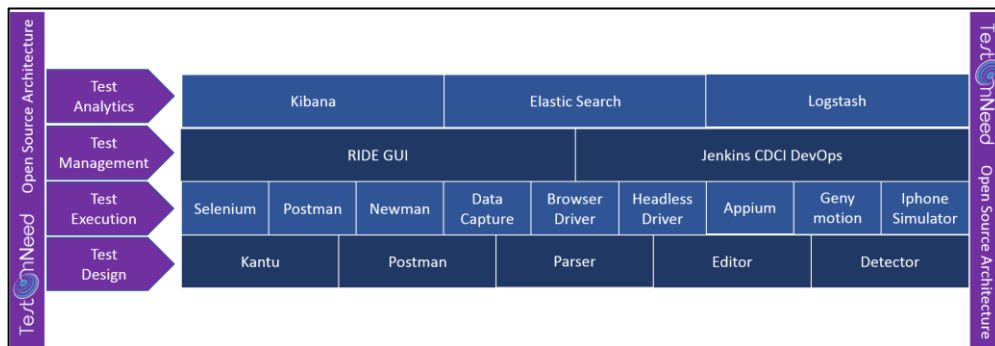
## Table of Contents

1. Introduction .....	3
2. What We Do.....	4
3. Prerequisites .....	5
4. Description of The Website .....	6
5. EveryAPI Tab.....	8
5.1 Develop Test Case .....	9
5.2 Prepare Test Case .....	16
5.3 Execute Test Case .....	18
5.3.1 Execute – Via GUI.....	19
5.3.1.1 Via GUI – API – Functional Testing.....	21
5.3.1.2 Via GUI – API – Automation Testing .....	25
5.3.1.3 Via GUI – API – Load Testing .....	28
5.3.2 Execute - Via TestOps .....	32
5.4 Analyze Results .....	35
6. References.....	37
6.1 Creation of New View .....	37
6.2 Setting up of Proxy in Chrome.....	39
6.3 Changing Folder Location of Postman .....	50

## 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 to the Installation Manual.

- o **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.

- o **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.

- o **Kibana**

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

- o **Jenkins**

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

- o **Selenium**

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

- o **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.

- o **Appium**

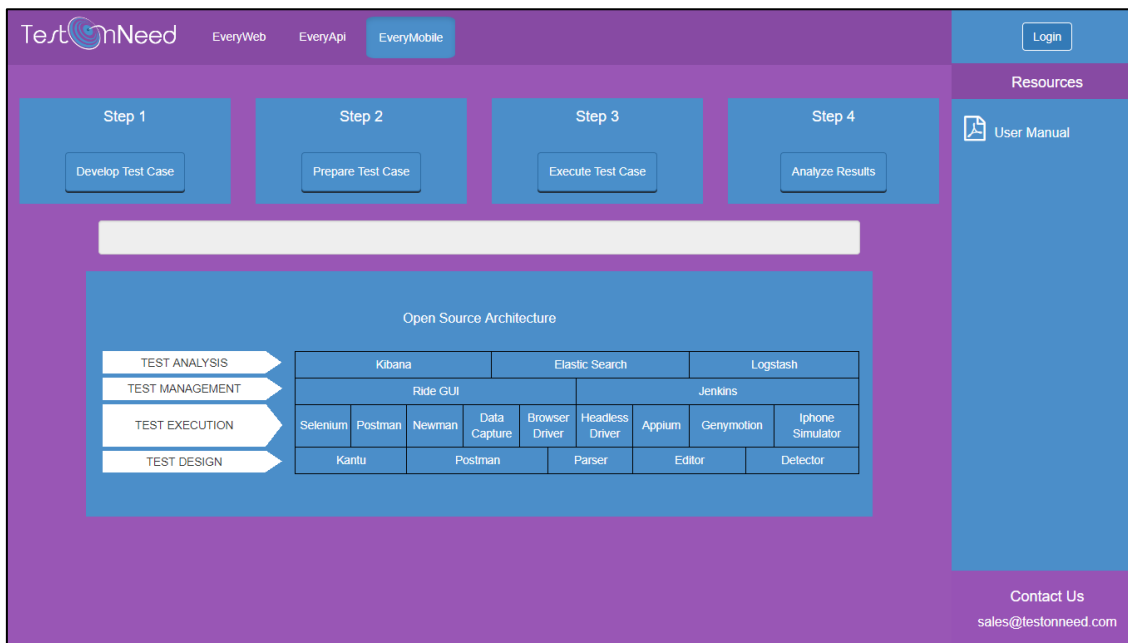
Appium is an automation tool for running scripts and testing native applications and mobile-web applications on android or iOS using a web driver.

- o **Emulator**

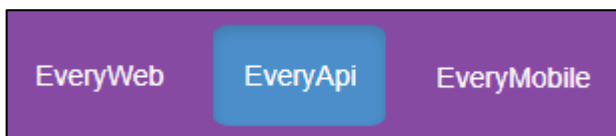
An emulator enables the host system to run software or use peripheral devices designed for the guest system.

## 4. DESCRIPTION OF THE WEBSITE

- The outline of the Website looks like below:



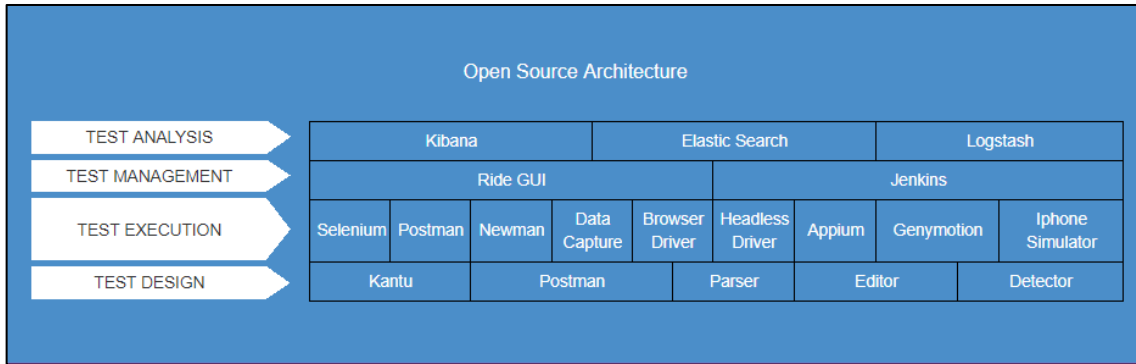
- As the web page opens up, there will be three tabs namely – EveryWeb, EveryAPI and EveryMobile.



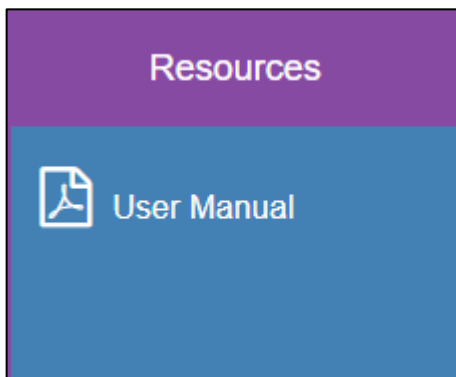
- Things that are common to all the tabs like the stack which contains a blank white space, all the open sources and user manual.
- In the blank space, the user will get know what to be done next by clicking any button.



- 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 installation document and user manual tell the user what and all should have to be installed and how it can be used. It will also change from tab to tab since applications and installation process is different. It is a hyperlink, on clicking it, will open the respective document in pdf format in a new tab in the browser.





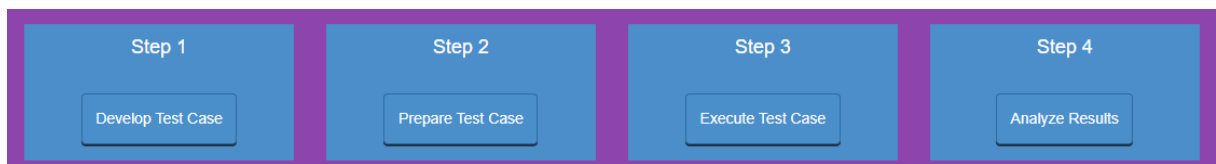
## 5. EVERYAPI TAB

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

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

- o To build the API, the user need to have API document files like RAML.

- This tab has 4 buttons on the top namely –
  1. Develop Test Case
  2. Prepare Test Case
  3. Execute Test Case
  4. Analyze Results



- **Develop Test Case**

The user can capture the backend API calls 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**

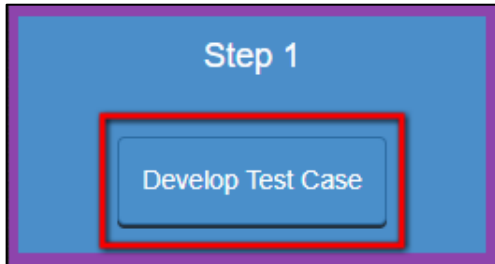
The user can analyze the results visually with the help of graphs

- Each step is explained below

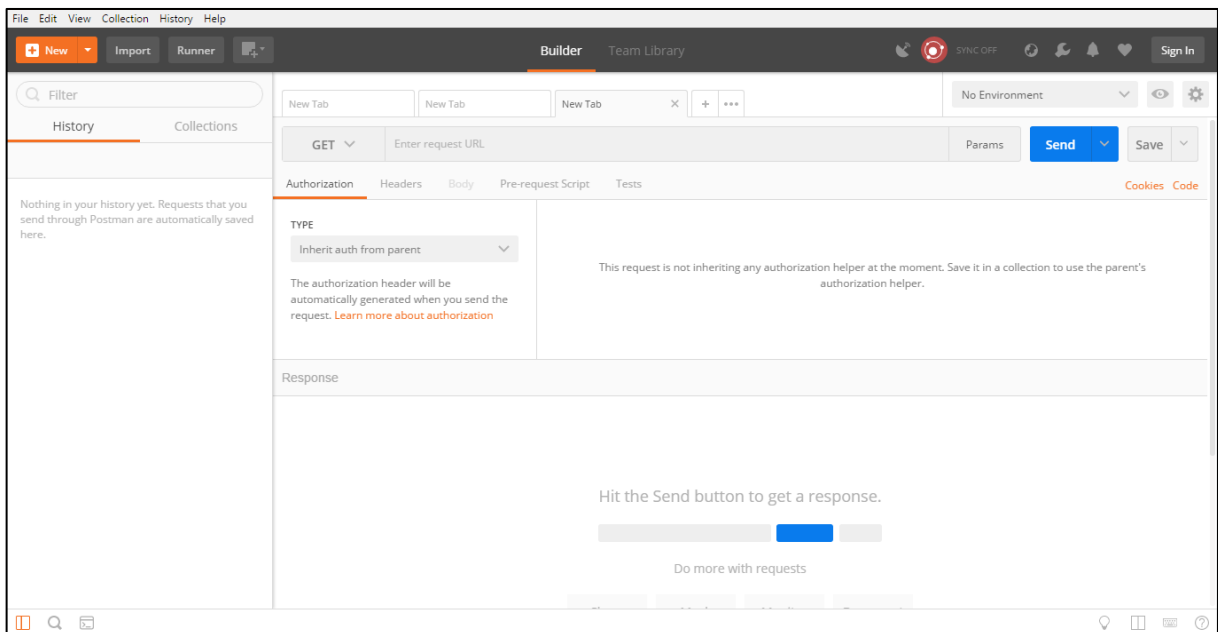
## 5.1 DEVELOP TEST CASE

The Develop Test Case button when clicked will open a new browser window and postman app for the user which can be used to develop the test case for the API.

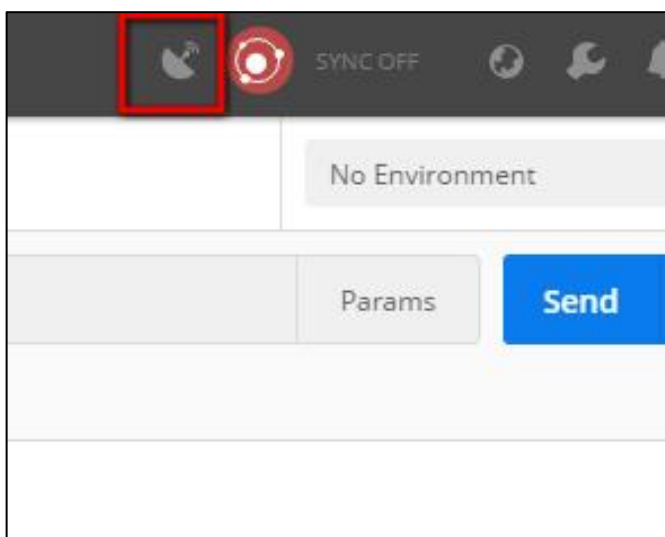
1. Click on **Develop Test Case** button.



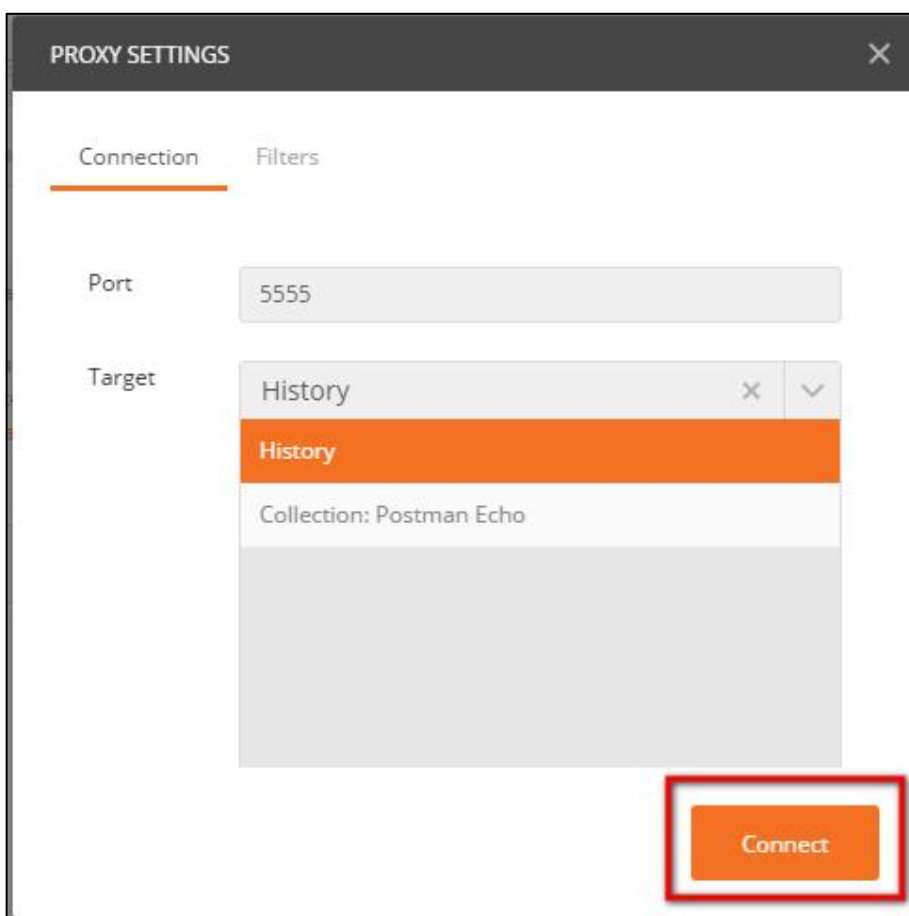
2. The user has to change the folder location of the postman. To change the folder location, click [Here](#).
3. For setting up of proxy in order to record, click [Here](#).
4. On clicking of the button, new browser window and Postman opens up.



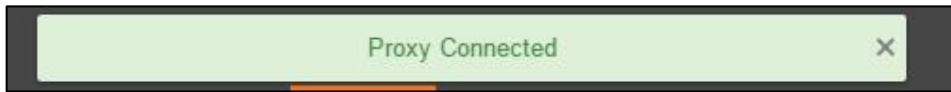
5. Start the proxy in postman application.



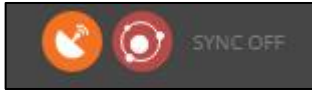
6. On clicking the proxy button, a pop window comes up and click **Connect**.



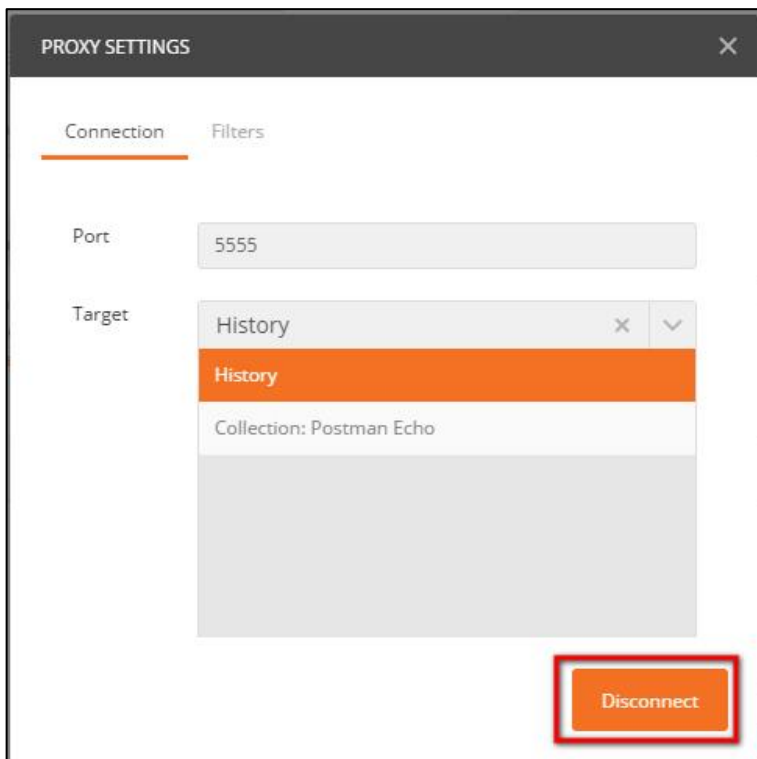
7. Once the connect button is clicked, at the top of the page a message will be displayed as **Proxy Connected** if connected.



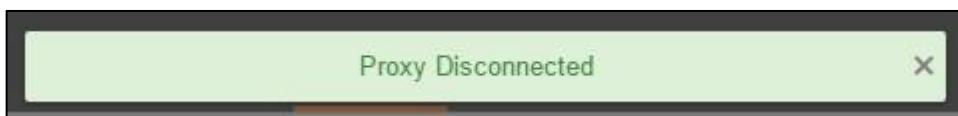
8. If it is connected, the proxy button changes to orange.



9. Now open the site that has to be record.
10. Now the user can perform the steps.
11. 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.



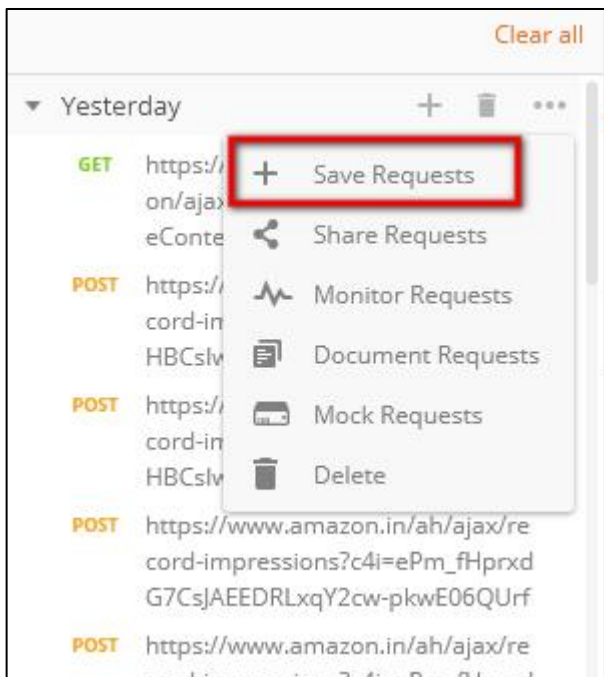
12. Once the disconnect button is clicked, at the top of the page a message will be displayed as **Proxy Disconnected**.



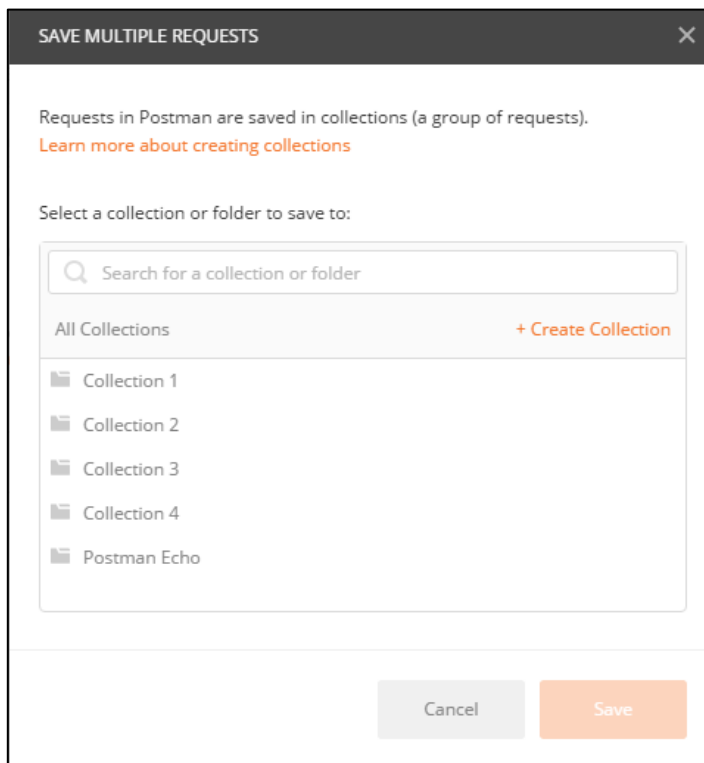
13. After disconnecting the proxy, the proxy button changes back to grey.



14. In the postman application, click Save Request and save in the collection in which the user want to.

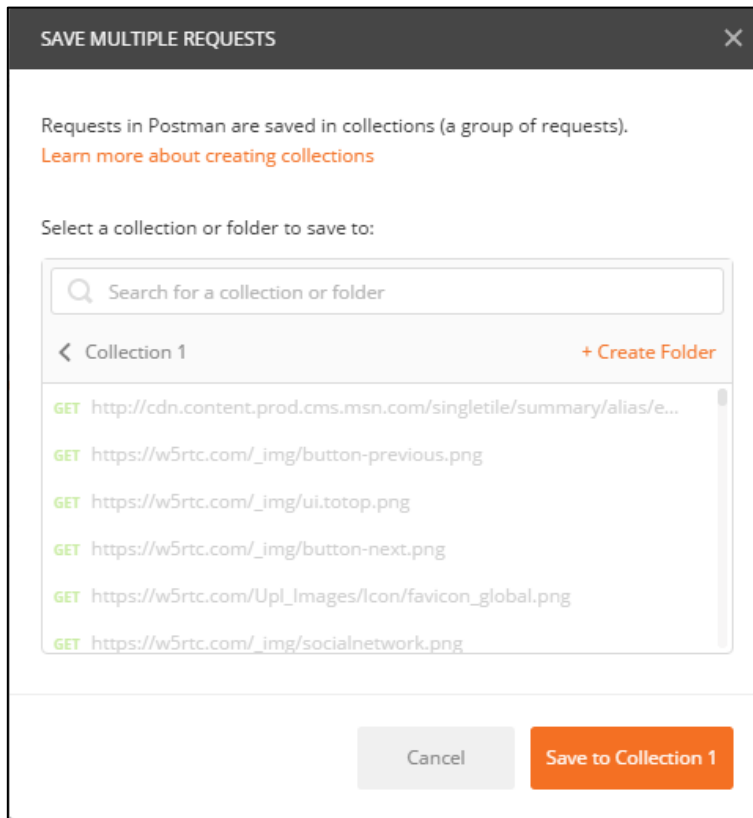


15. On clicking of the Save Requests button, a dialog box appears.



16. 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.

17. On click of the collection name, the Save button becomes active.



18. 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**.
19. 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**.
20. 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\_TestVNF\_Ve-Vnfm\_<Test Function>\_001** For example, if the user has recorded the scenario of login, then the user has to save the test case as **AF\_TestVNF\_Ve-Vnfm\_Login\_001**.
21. 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\_TestVNF\_Ve-Vnfm\_<Test Function>\_001**. 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\_TestVNF\_Ve-Vnfm\_LoginGateways\_001**.

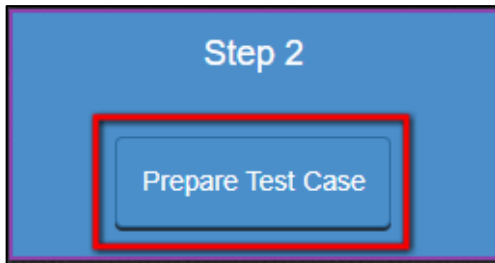
22. Load testing is performance testing technique using which the response of the system is measured under various load conditions. The load testing is performed for normal and peak load conditions. In that case, the test case should be saved as **AL\_TestVNF\_Ve-Vnfm\_<Test Function>001**. For example, 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\_TestVNF\_Ve-Vnfm\_Login\_001**.
23. 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.**

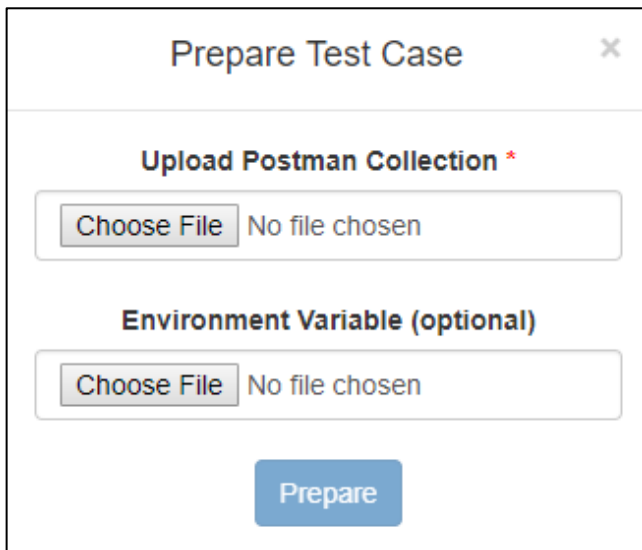


## 5.2 PREPARE TEST CASE

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



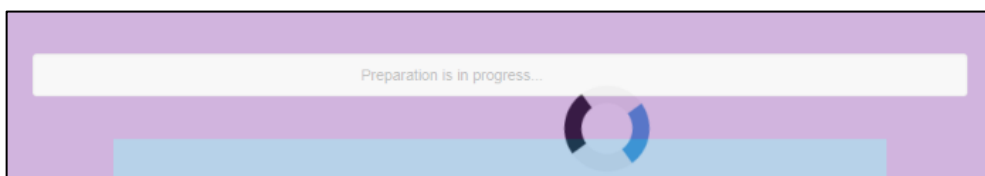
2. On click of the button, a pop-up appears.
3. In the pop-up, the user has to upload 2 files. One is the Postman JSON file and other is the environmental variables file which should also be in JSON format. The postman JSON file **must be uploaded** but the environmental variables JSON file **is optional**.



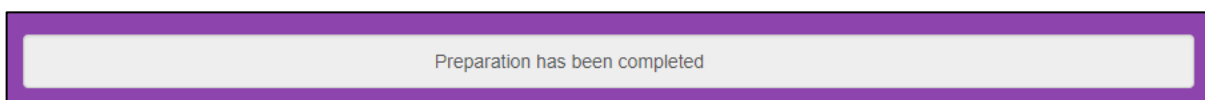
4. Click on the Choose file button.
5. Surf to the folder location where saved the JSON files that was downloaded in the previous step.
6. Now upload the JSON files.

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

8. On 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 **AF\_TestVNF\_Ve-Vnfm\_Login\_001** in the previous step then it will create a folder with same.
  - 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.



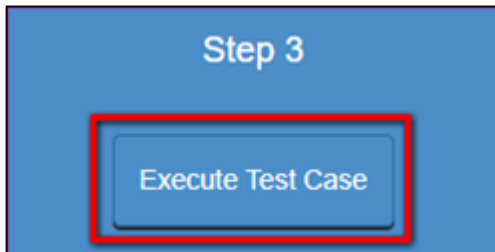
10. Once preparation is finished, a message will be shown as “**Preparation has been completed**” in the status bar.



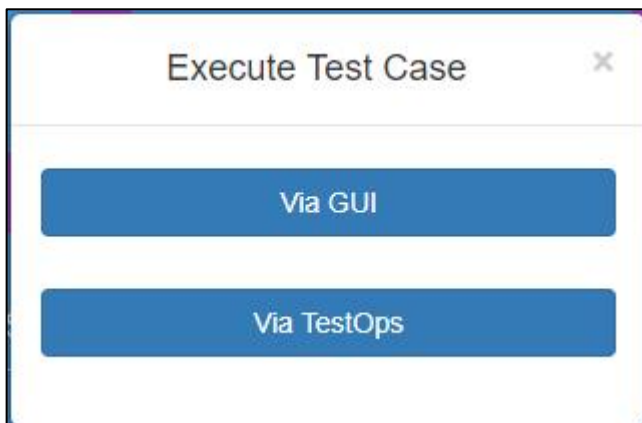
## 5.3 EXECUTE TEST CASE

The Execute Test Case button is used for the execution process using RIDE, Postman and Jenkins.

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

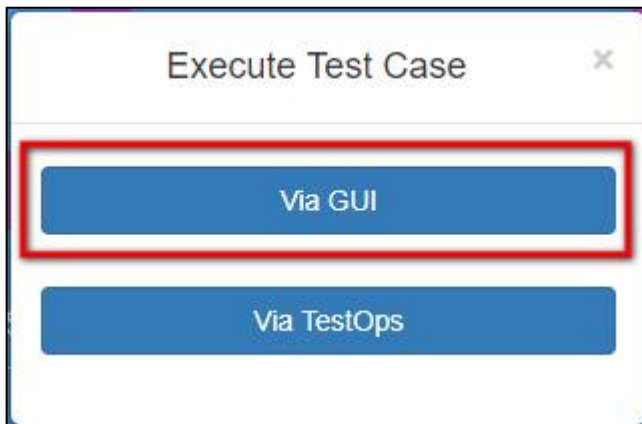


2. On clicking on the button, a pop up appears.
3. From the pop-up, select the button either as **Via GUI**, **Via TESTOPS**.

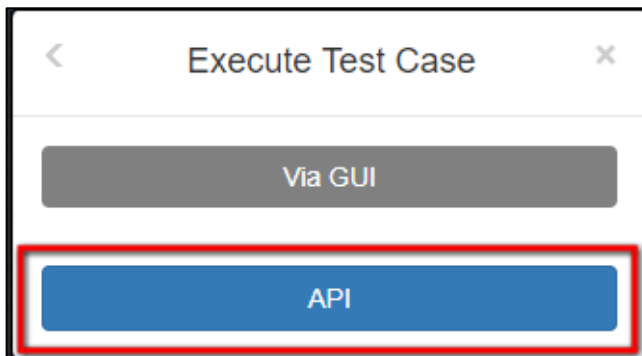


### 5.3.1 EXECUTE – VIA GUI

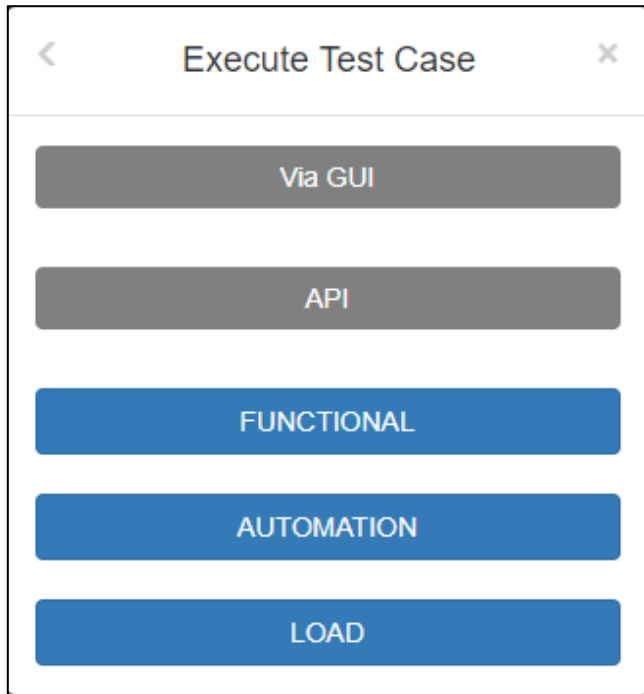
1. Click the **Via GUI** button.



2. On clicking the button, **API** button appears.

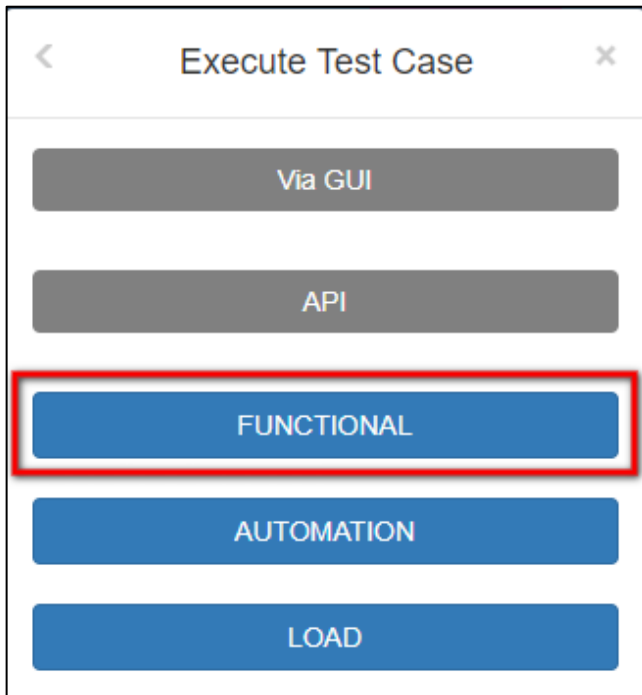


3. Under **API** button, 3 buttons will appear namely – **Function, Automation and Load**.

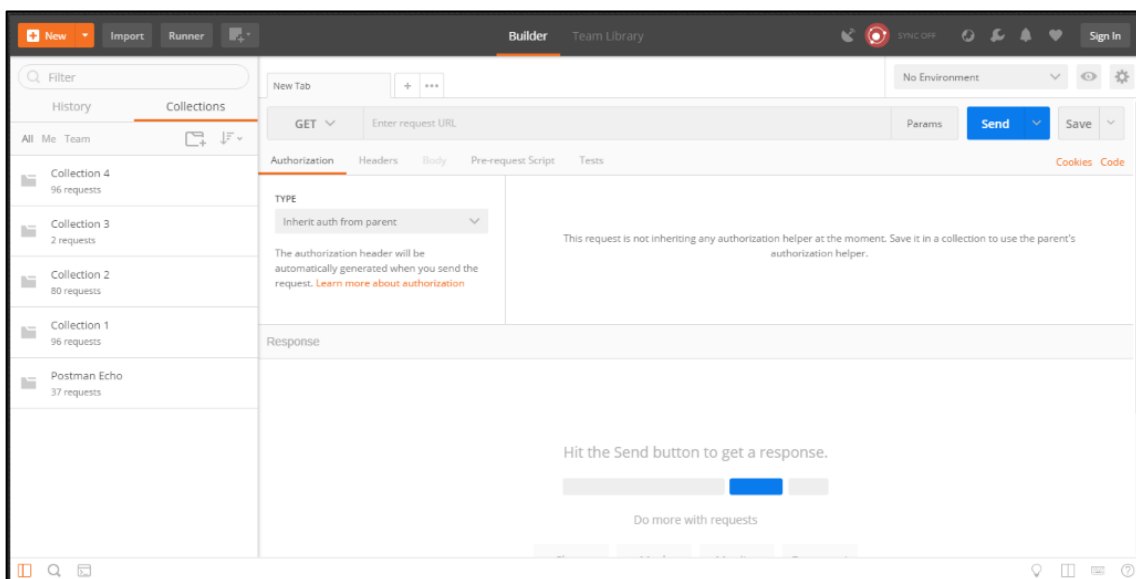


### 5.3.1.1 VIA GUI – API – FUNCTIONAL TESTING

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

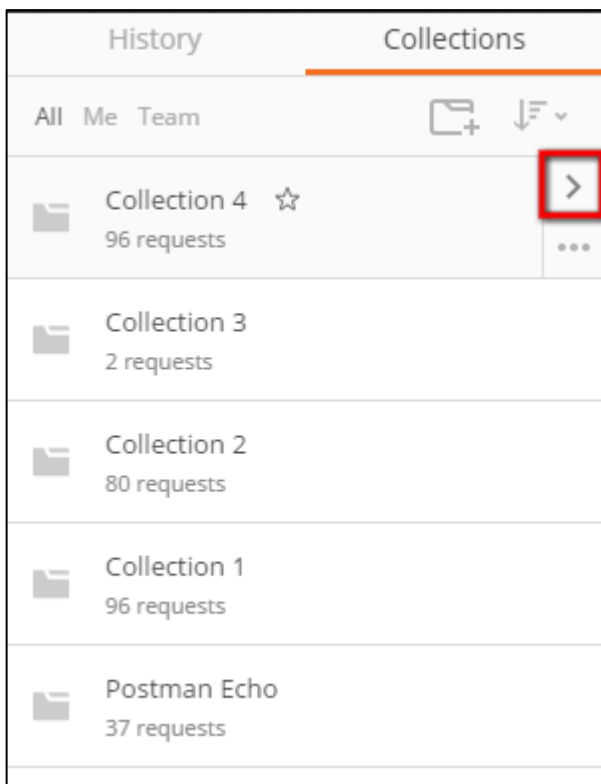


2. To perform functional testing, press **Functional** button.
3. On click of the button, **Postman** will open.

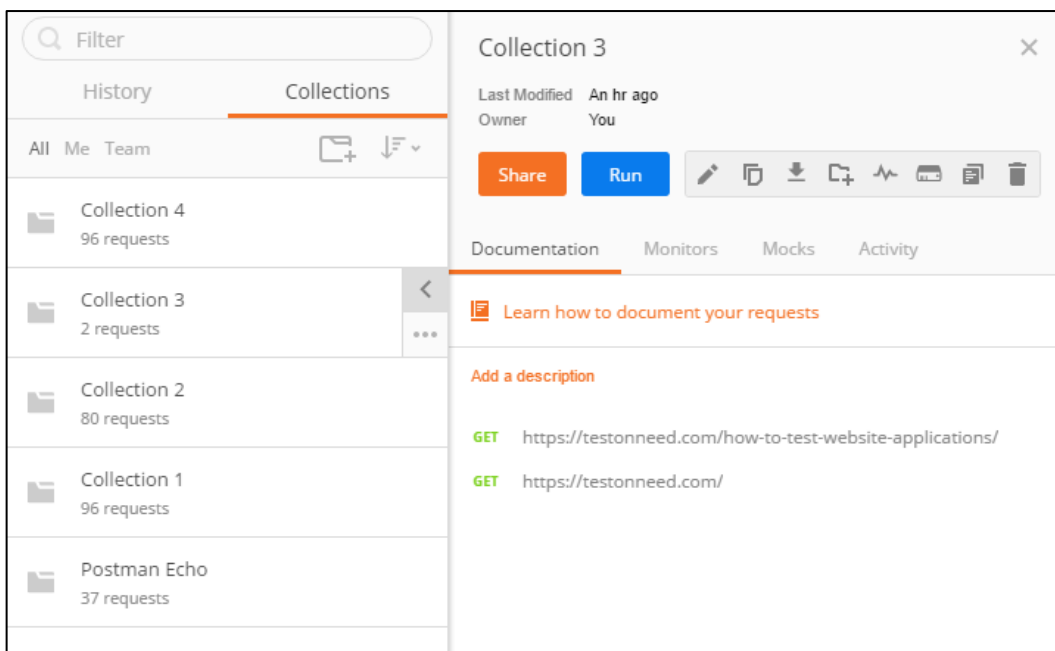


4. On the left, there will be two tabs History and Collections. In the collections tab look for the **Collection** in which the JSON was saved during the **Record Test Case** button.

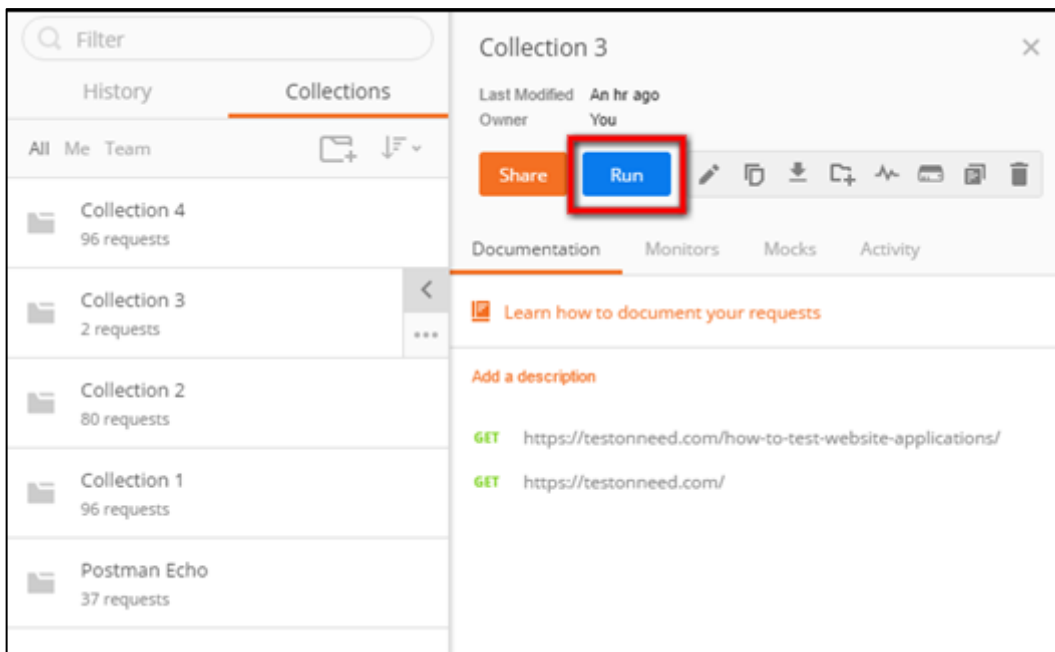
5. On mouse hover of the **Collection name**, click on the right arrow.



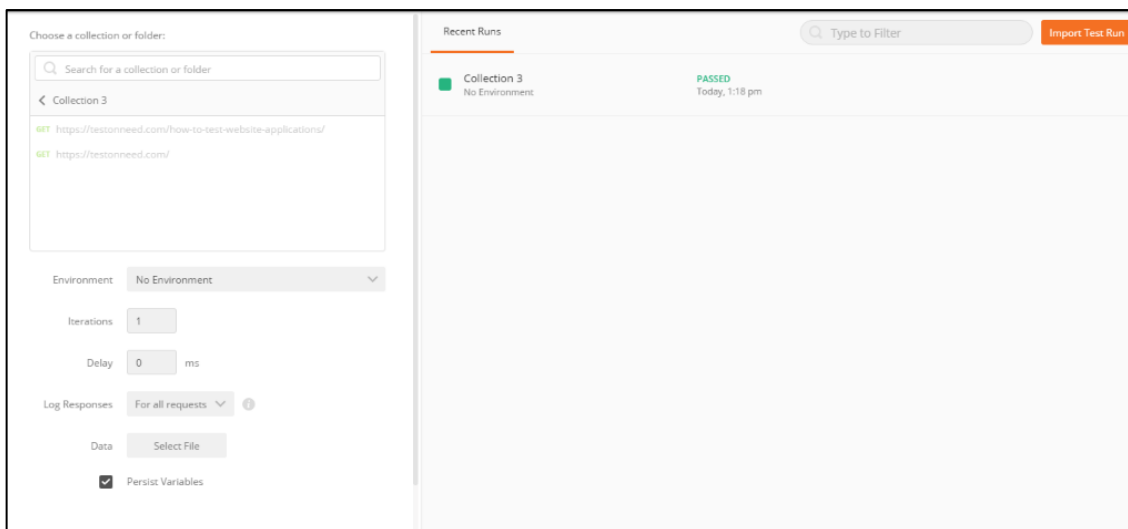
6. On clicking of the arrow button, a small window comes on the right next to the collection selected.



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

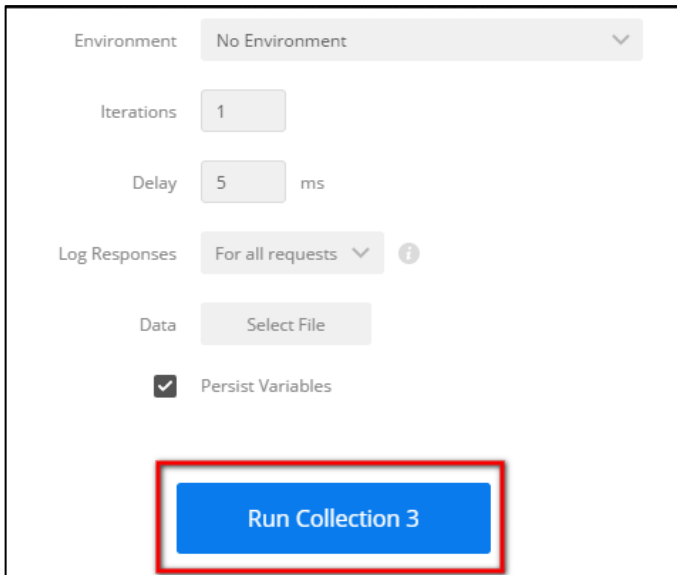


8. On click of **Run** button, a new window will open called **Collection Runner**.





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



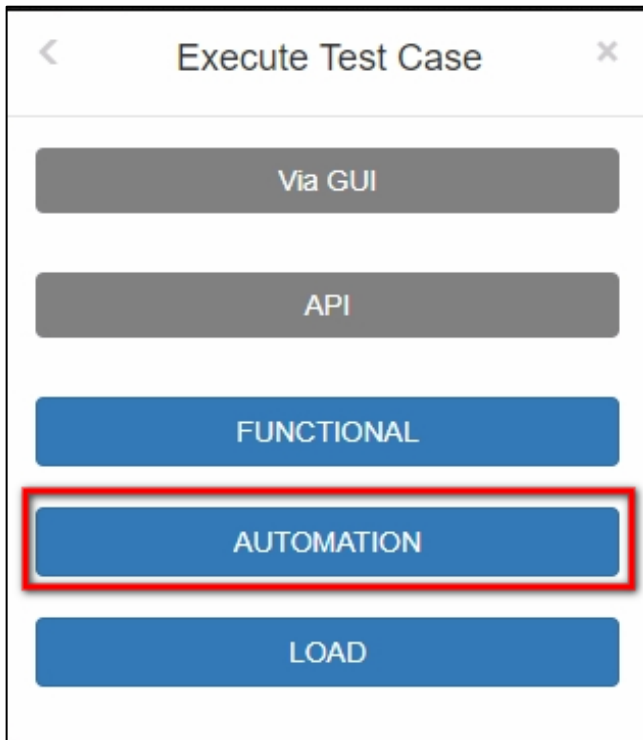
The screenshot displays a configuration panel for running a collection. The parameters are as follows:

- Environment:** A dropdown menu set to "No Environment".
- Iterations:** A text input field containing the value "1".
- Delay:** A text input field containing "5" followed by a unit selector set to "ms".
- Log Responses:** A dropdown menu set to "For all requests" with an information icon to its right.
- Data:** A button labeled "Select File".
- Persist Variables:** A checkbox that is checked.

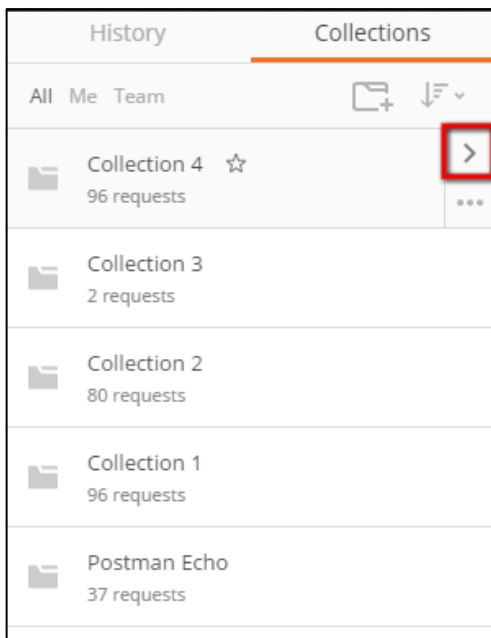
At the bottom of the panel, a blue button labeled "Run Collection 3" is highlighted with a red rectangular border.

### 5.3.1.2 VIA GUI – API – AUTOMATION TESTING

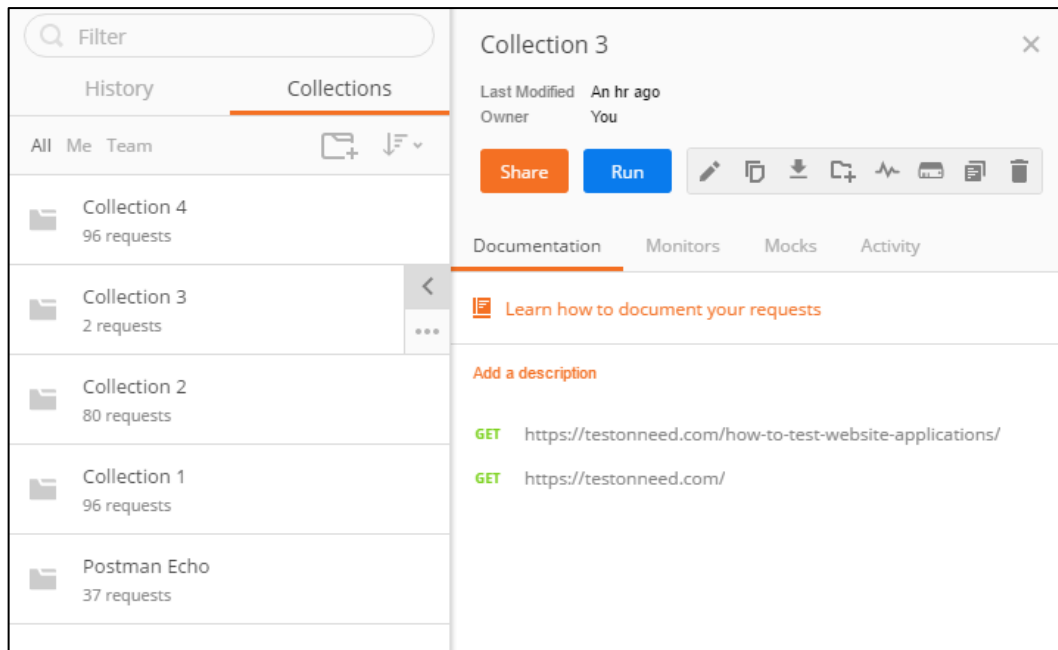
1. To perform automation testing, press **Automation** button.



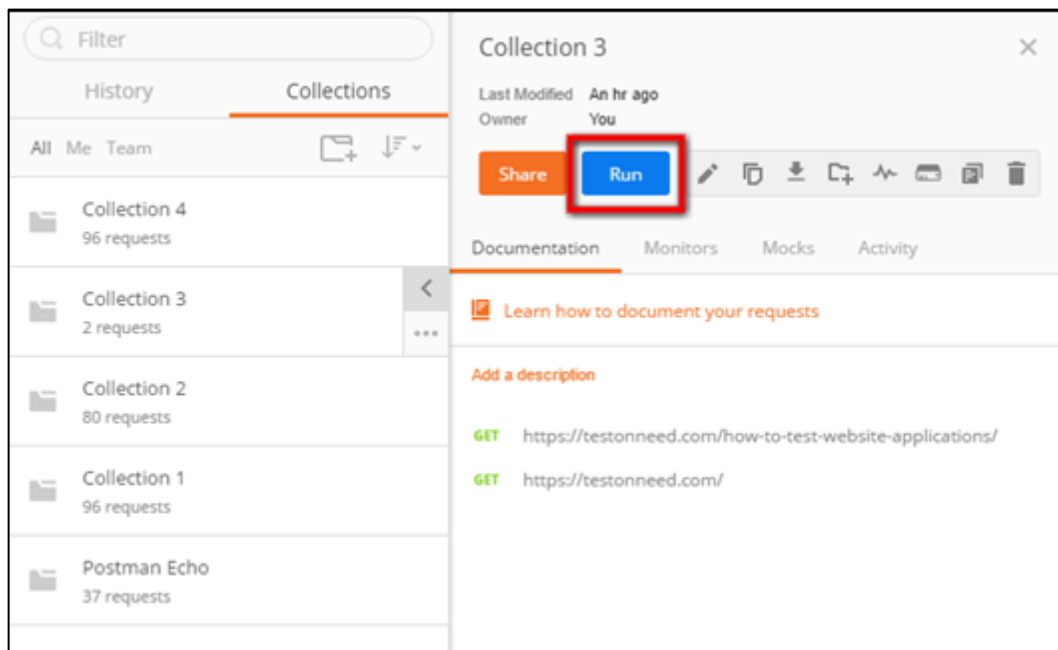
2. On click of the button, **Postman** will open.
3. On the left, there will be two tabs History and Collections. In the collections tab look for the **Collection** in which the JSON was saved during the **Record Test Case** button.
4. On mouse hover of the **Collection name**, click on the right arrow.



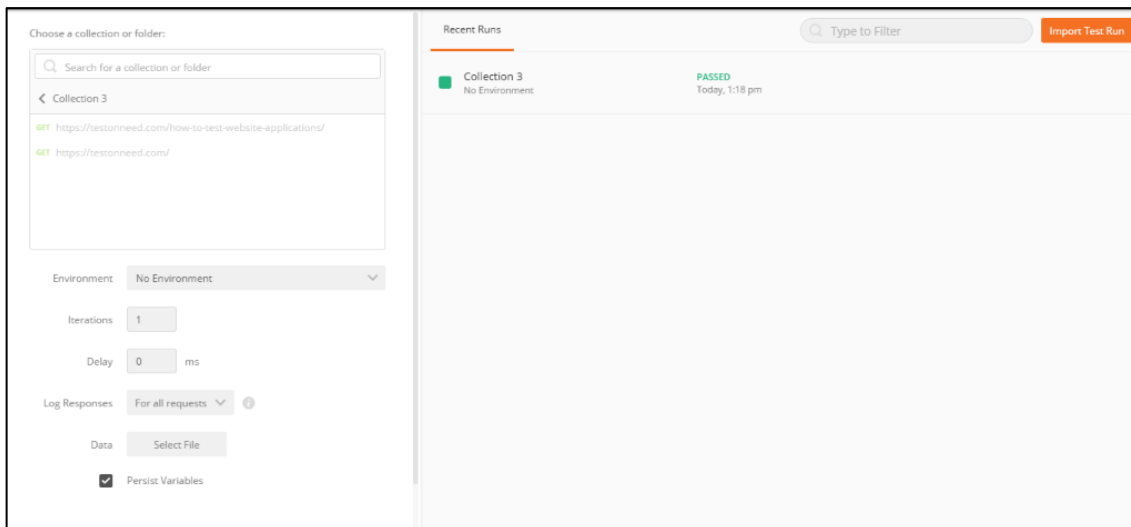
5. On clicking of the arrow button, a small window comes on the right next to the collection selected.



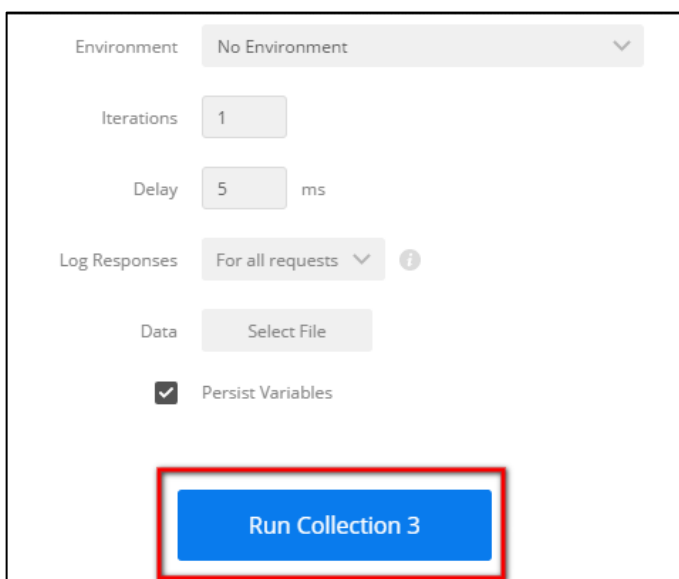
6. From the pop-up, click **Run** button.



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



8. In the window, on the left panel, set the parameters and click **Run <Collection\_Name>**.

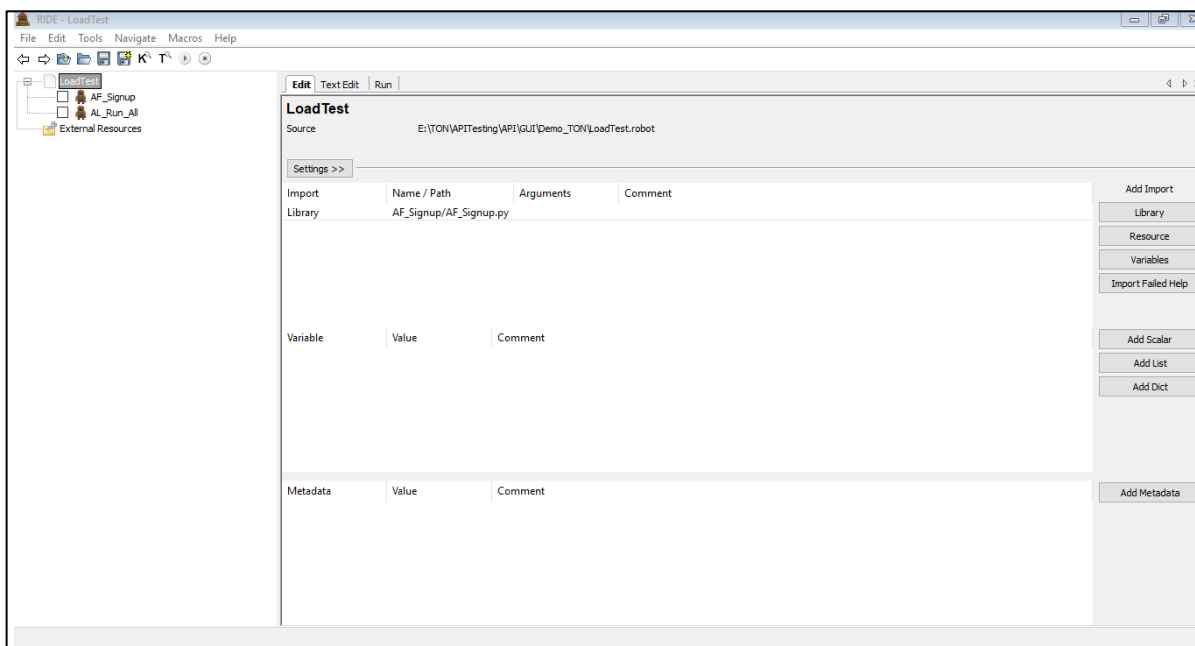


### 5.3.1.3 VIA GUI – API – LOAD TESTING

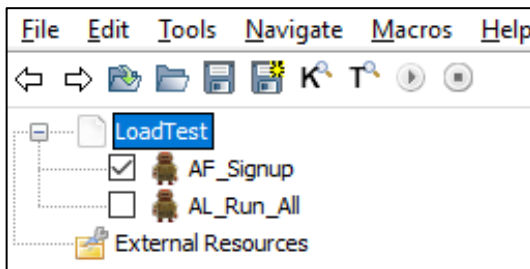
1. To perform load testing, press **Load** button.



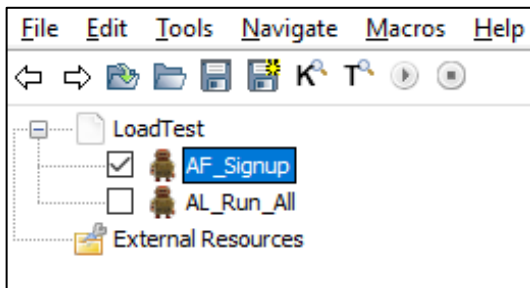
2. On click of the **Load** button, **RIDE** will open.



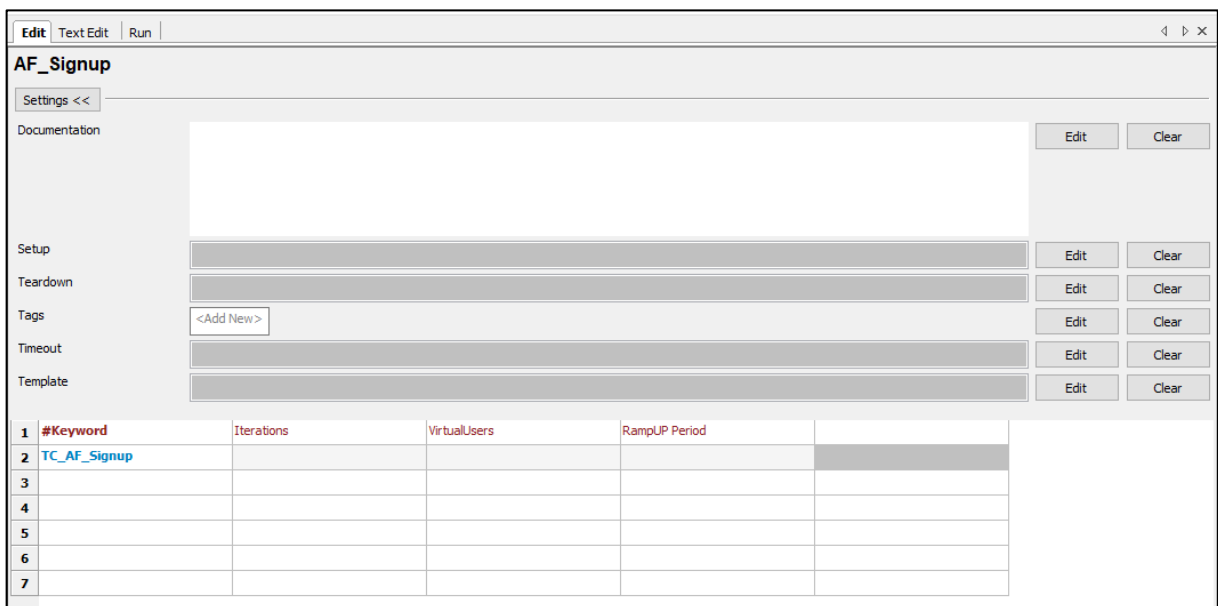
3. On the left panel, select one test case to execute.



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



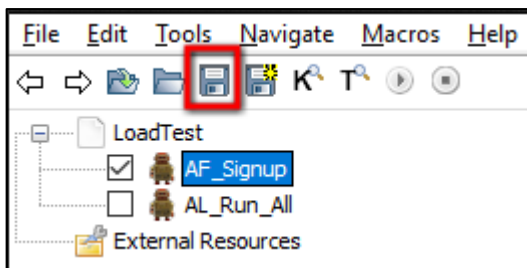
5. On click of the test case name, edit tab changes so that parameters can be configured.



6. To configure the parameter, click on the test case name.
7. On click of the test case name, edit tab changes so that parameters can be configured.
8. The value of the parameters is:  
Iterations, VirtualUsers and RampUP Period.
9. Configure the parameters – **Iterations, VirtualUsers and RampUP Period.**

1	#Keyword	Iterations	VirtualUsers	RampUP Period	
2	TC_AF_Signup	1	1	1	
3					
4					
5					
6					
7					

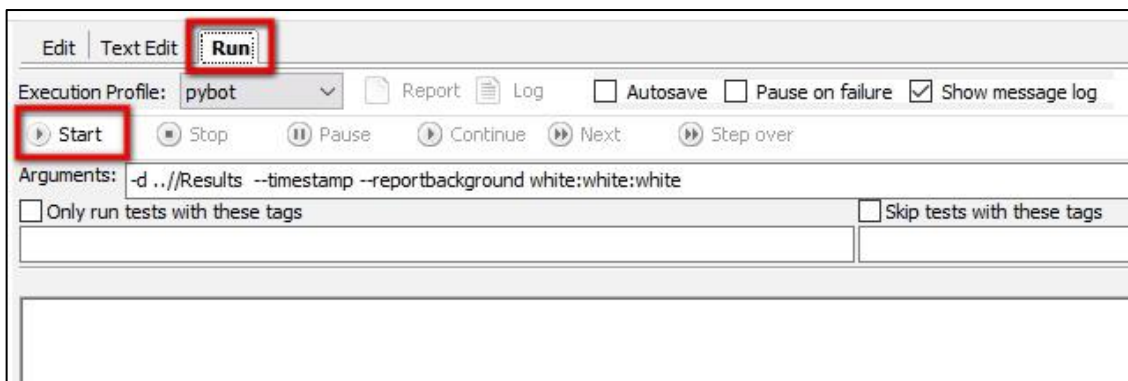
10. Click **Save** button to save the test case.



11. Configure the Arguments textbox by entering:

**-d ../Demo\_TON<test\_case\_name>\Results --timestamp --reportbackground white:white:white**

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



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



14. 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	
All Tests	1	1	0	00:00:55	

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

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

**Test Details**

Totals Tags Suites Search

Type:
☐ Critical Tests  
☐ All Tests

15. 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	
All Tests	1	1	0	00:00:55	

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

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

**Test Execution Log**

SUITE TestCases

Full Name: TestCases  
Source: E:\TONWebTesting\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

00:00:55.548

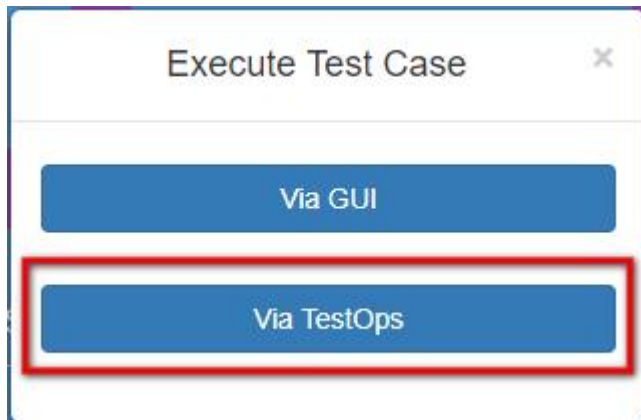
TEST BF\_Signup

00:00:55.131

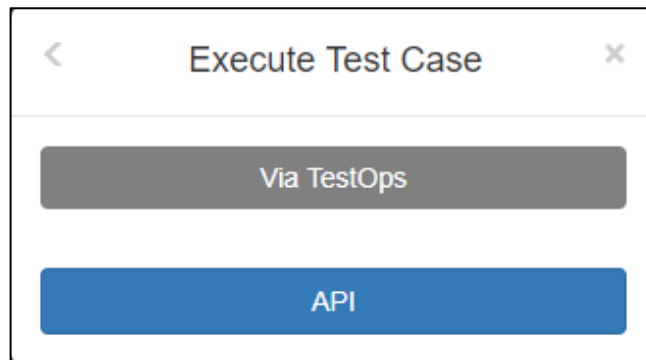


### 5.3.2 EXECUTE - VIA TESTOPS

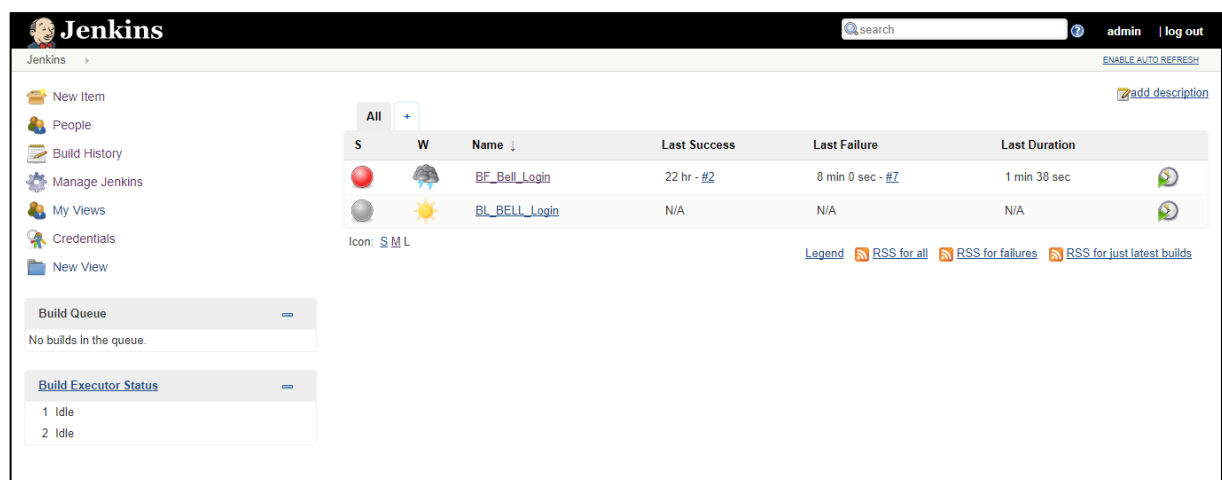
1. Click the Via TestOps button.



2. On clicking the button, **API** will appear.



3. On click of any one button, the **Jenkins** will open.
4. On opening of the Jenkins, the prepared test case will automatically appear as a new job.



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

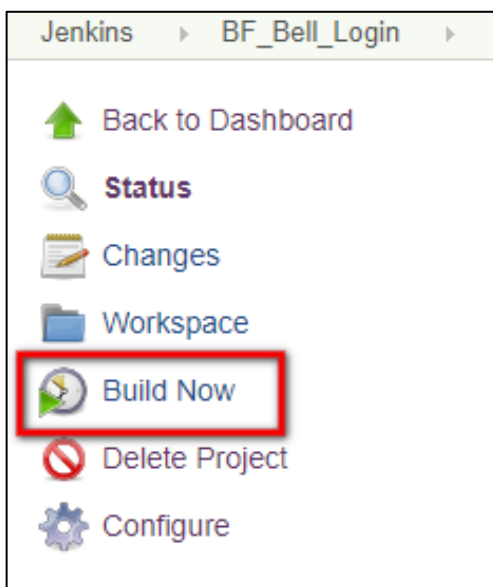
5. Click on the job that has to be executed.



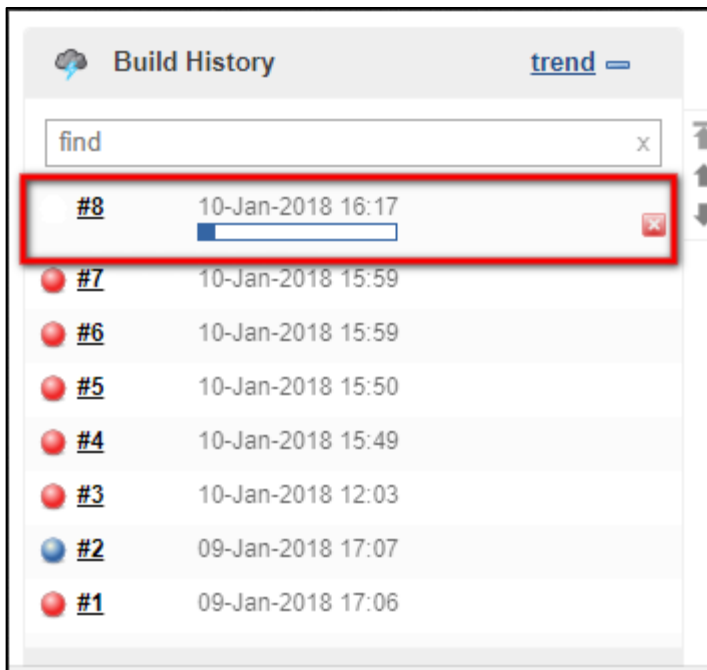
The screenshot shows the Jenkins job list interface. At the top right is a link 'add description'. Below it is a table with columns: S, W, Name, Last Success, Last Failure, Last Duration, and an icon column. The first row, 'BF\_Bell\_Login', is highlighted with a red box. The second row, 'BL\_BELL\_Login', is also visible. Below the table is a link 'Icon: S M L' and a 'Legend' section with three RSS links: 'RSS for all', 'RSS for failures', and 'RSS for just latest builds'.

S	W	Name	Last Success	Last Failure	Last Duration	
		<a href="#">BF_Bell_Login</a>	22 hr - #2	8 min 0 sec - #7	1 min 38 sec	
		<a href="#">BL_BELL_Login</a>	N/A	N/A	N/A	

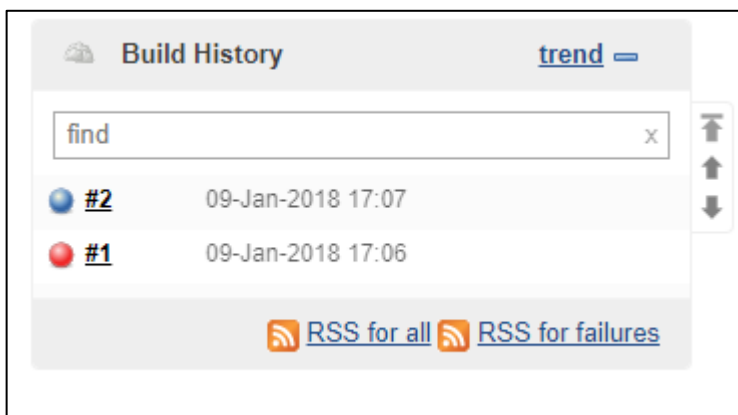
6. On the left side, select Build Now.



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



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



9. All the jobs of the Jenkins will be saved in:
- > TON > WebTesting > Browser > TestOps > Demo\_TON > <Job\_Name\_Folder>**

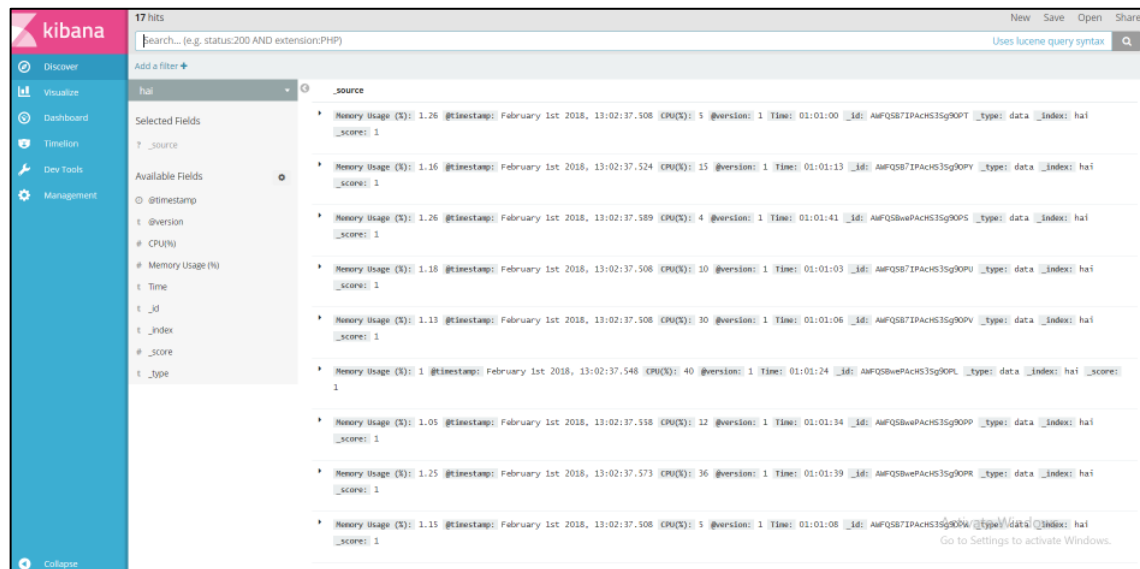
## 5.4 ANALYZE RESULTS

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

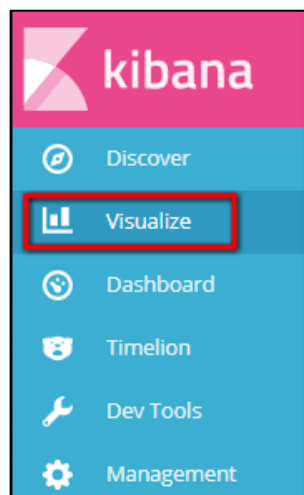
1. Click on the **Analyze Results**.



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



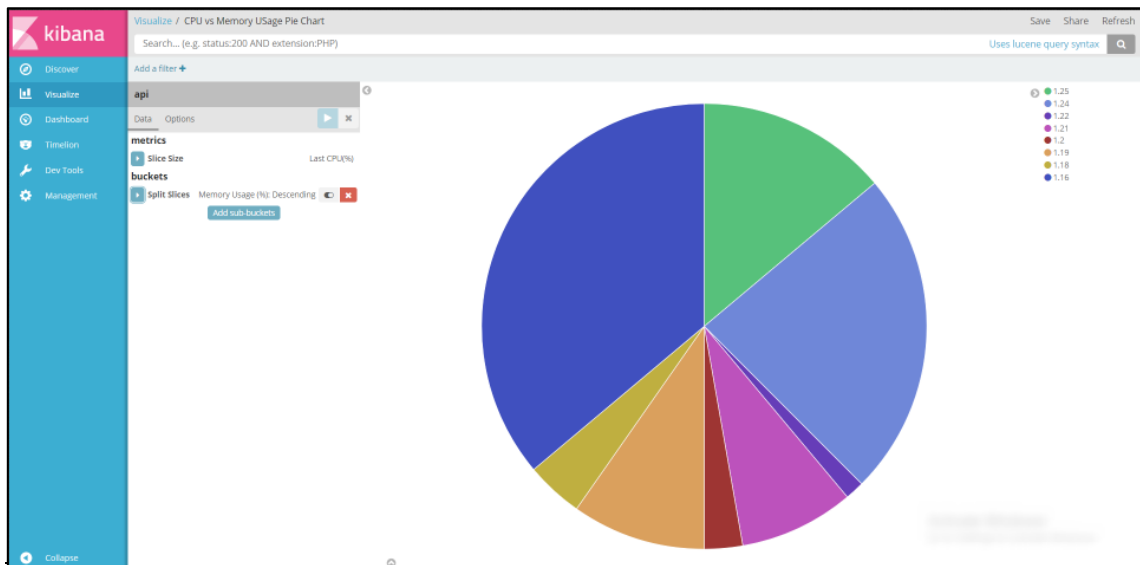
3. On the left panel, click Visualize.



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

Search...		+	1-5 of 5
<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

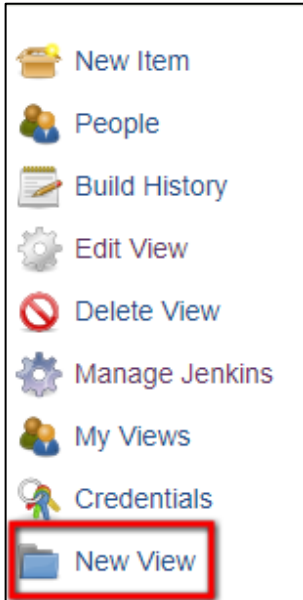
- On click of the chart name, the user can see the graph based on the type of the chart. (The example shows for Pie Chart)



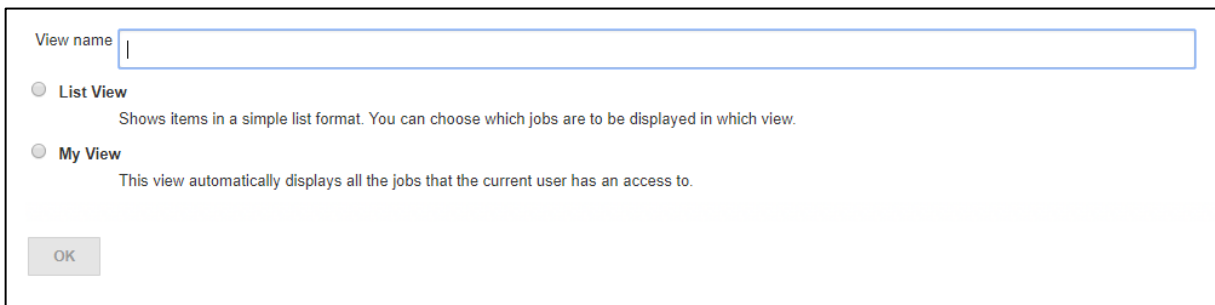
## 6. References

### 6.1 CREATION OF NEW VIEW

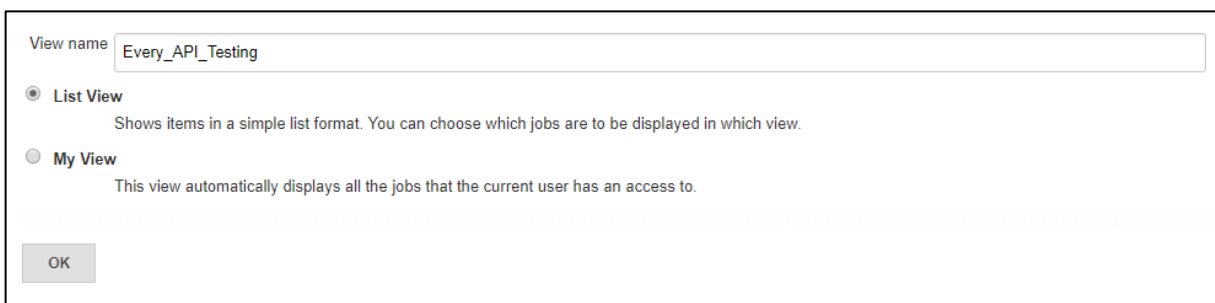
- To create a new view, on the left panel, click **New View**.



- On clicking, give the name of the tab as **Every\_API\_Testing** and select the type of view.

A screenshot of the 'New View' dialog box. It has a text input field for 'View name' which is currently empty. Below the field 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.' and 'This view automatically displays all the jobs that the current user has an access to.' 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' 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 (highlighted in grey).

- 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

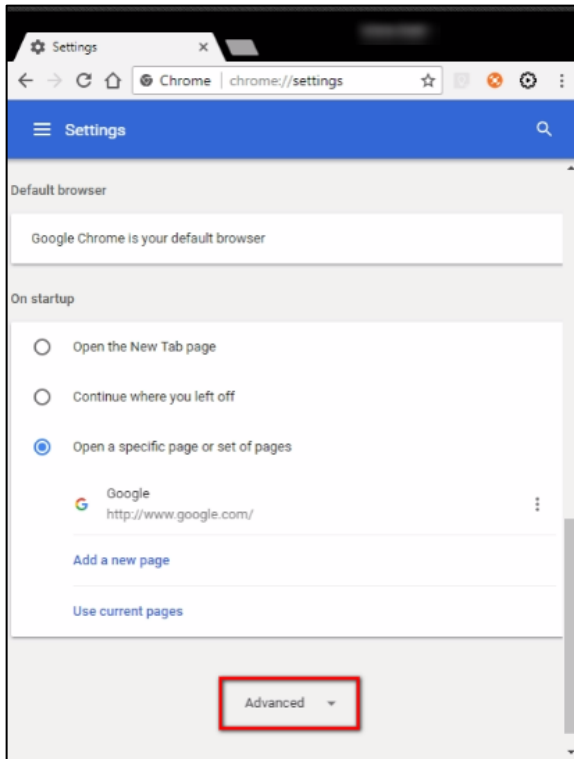
Add Job Filter

Columns

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

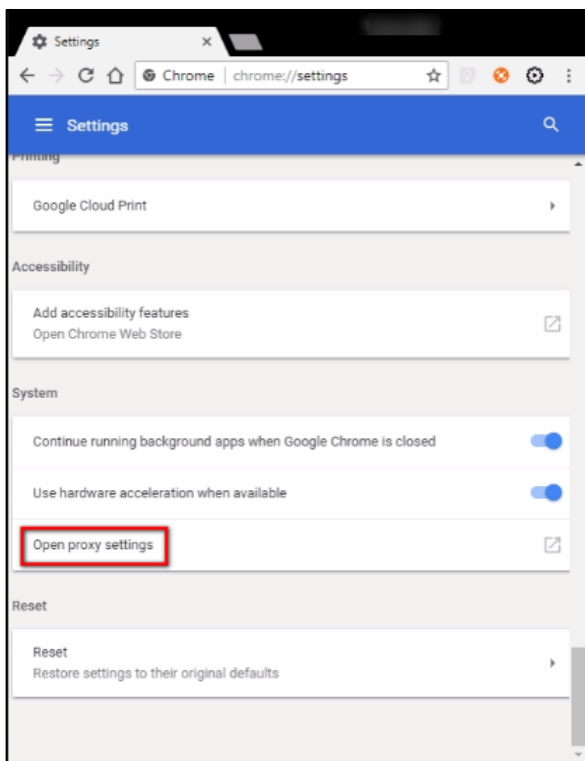
## 6.2 SETTING UP OF PROXY IN CHROME

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

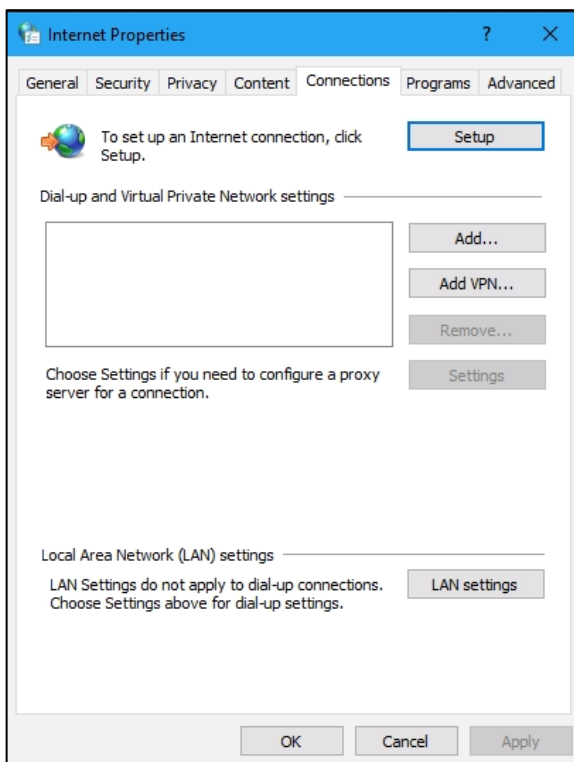


- Again, scroll down and click “Open Proxy Settings”.

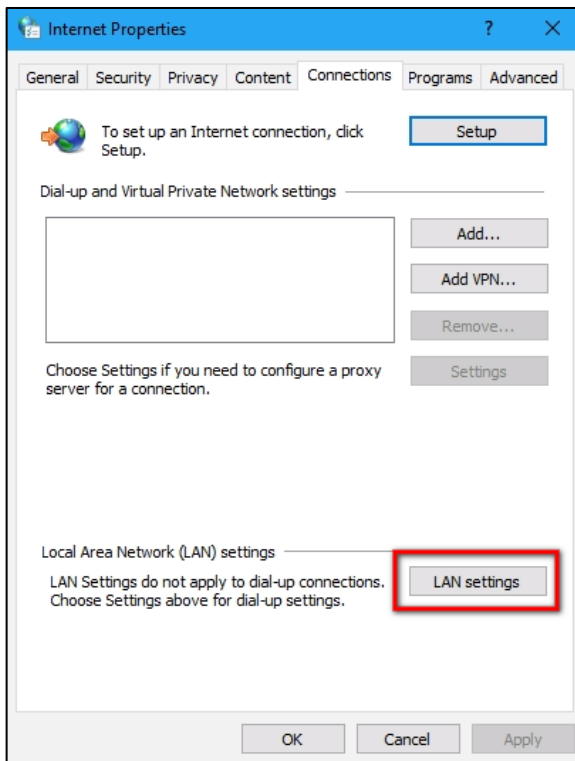




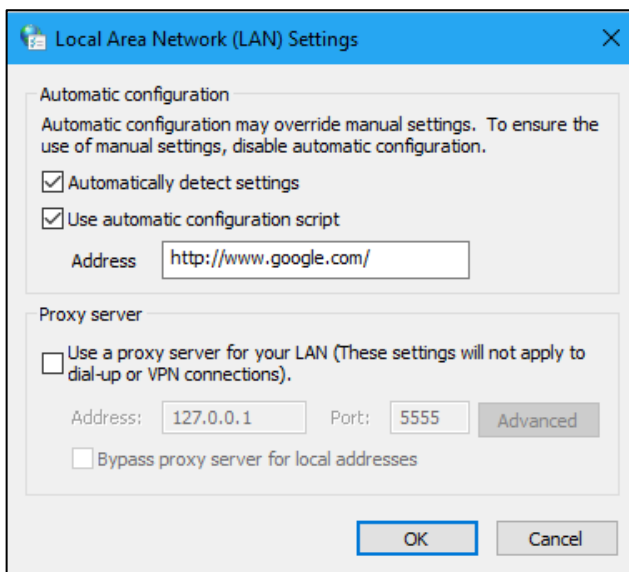
- A dialog box will appear.



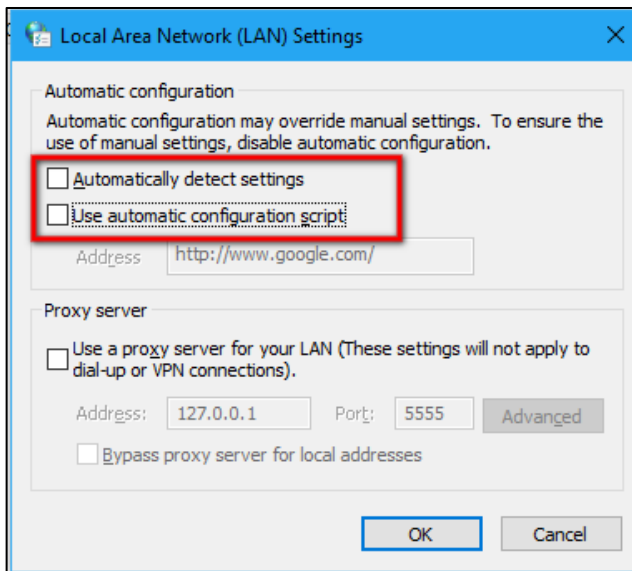
- In the dialog box, click LAN Settings.



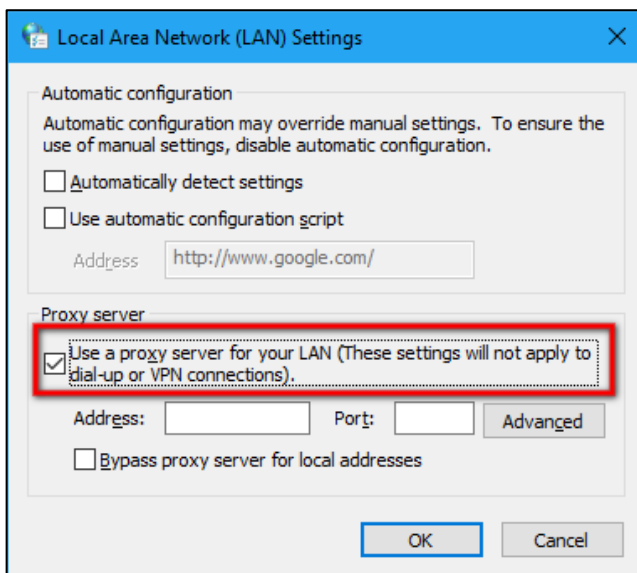
- It will display another dialog box.



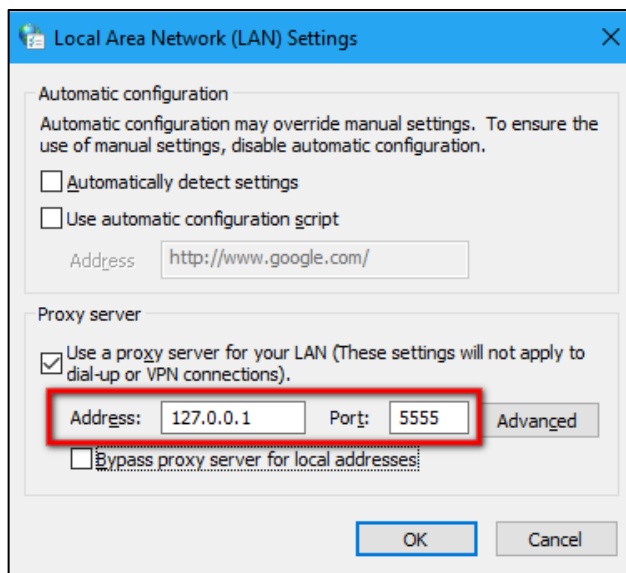
- In the dialog box, uncheck the Automatically detect settings and Use automatic configuration script under Automatic configuration.



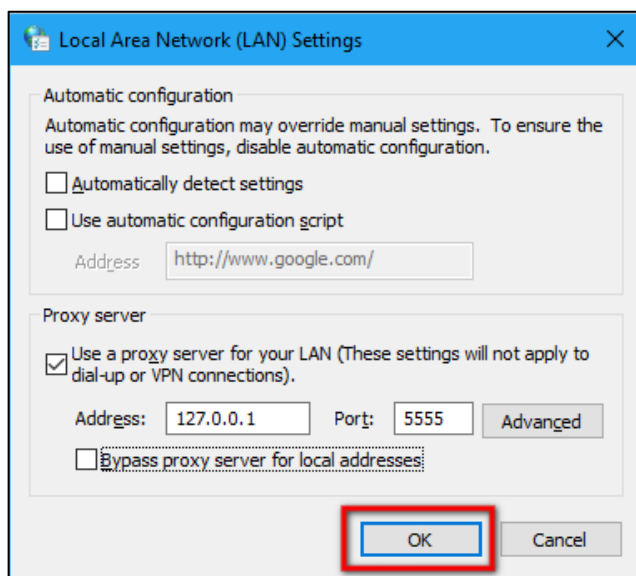
- In the same dialog box, check the box under Proxy server.



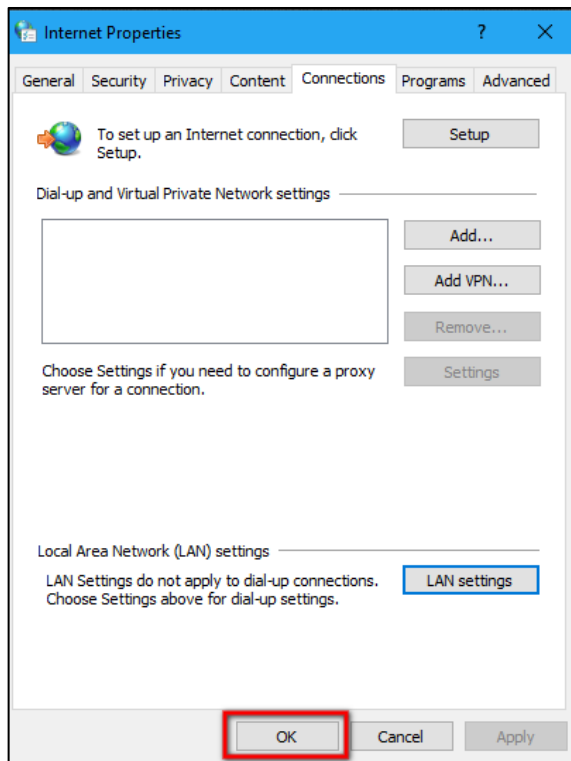
- On checking the box, the Address and Port text boxes becomes active.
- In the address text box, enter the ip address as "127.0.0.1" and in the port text box, enter the port number as "5555".



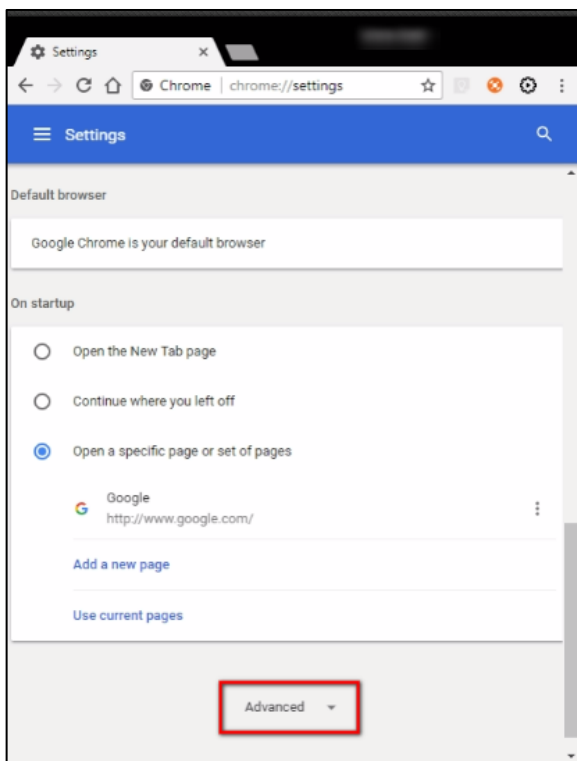
- After entering the address and port number, click Ok.



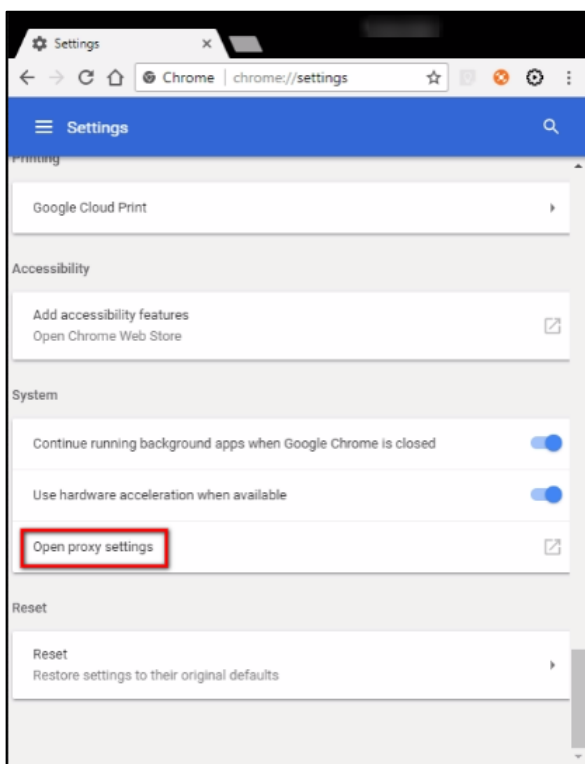
- Again, in the internet properties dialog box, click Ok.



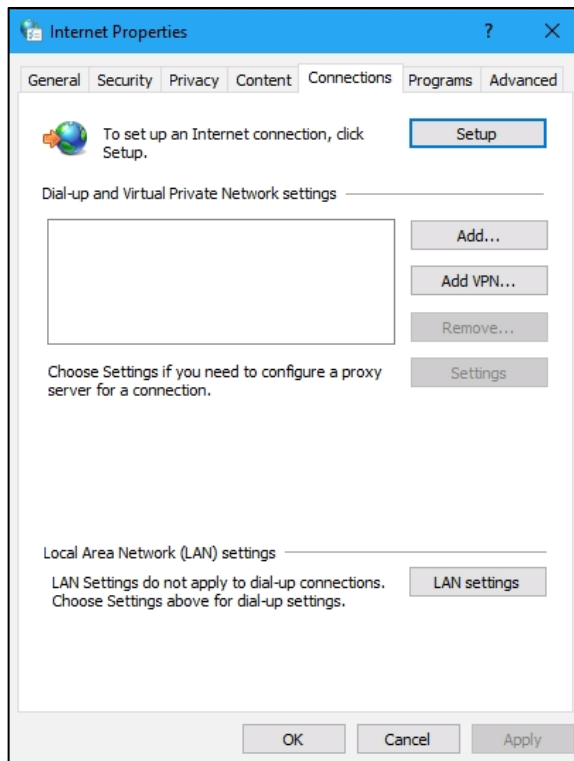
- When the user has finished the recording, the user has to revert back the changes in order to connect to the internet.
- To connect the revert the changes back, open Chrome Settings.
- Scroll down and click Advanced.



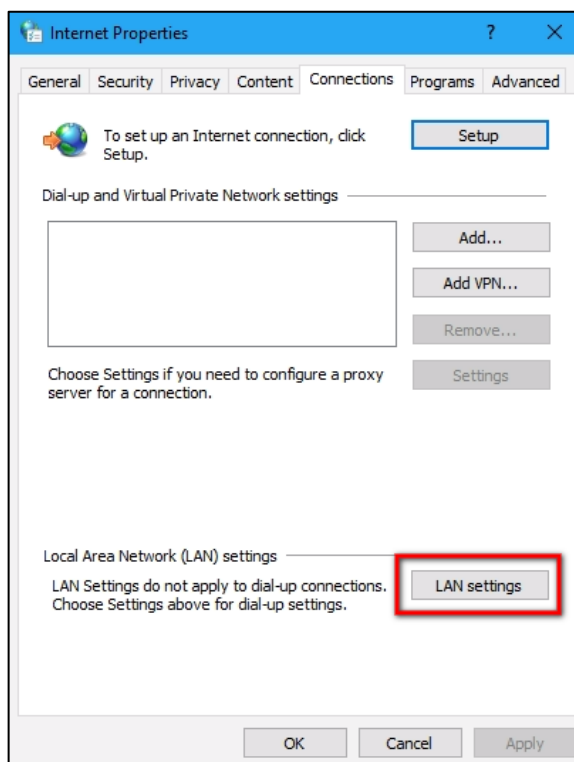
- Again, scroll down and click “Open Proxy Settings”.



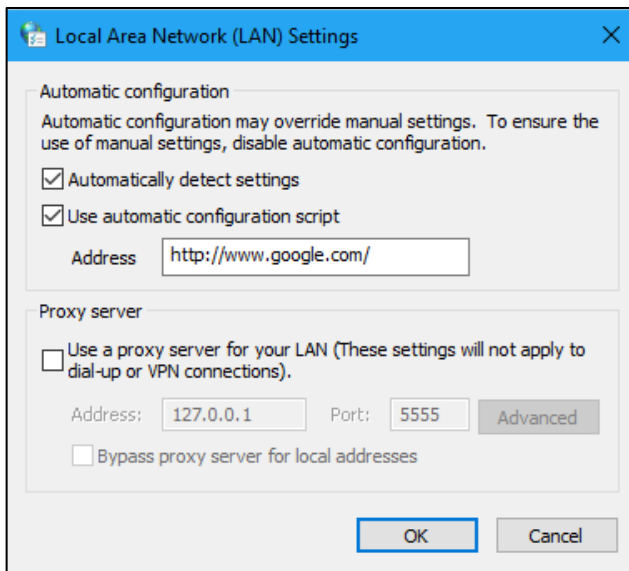
- A dialog box will appear.



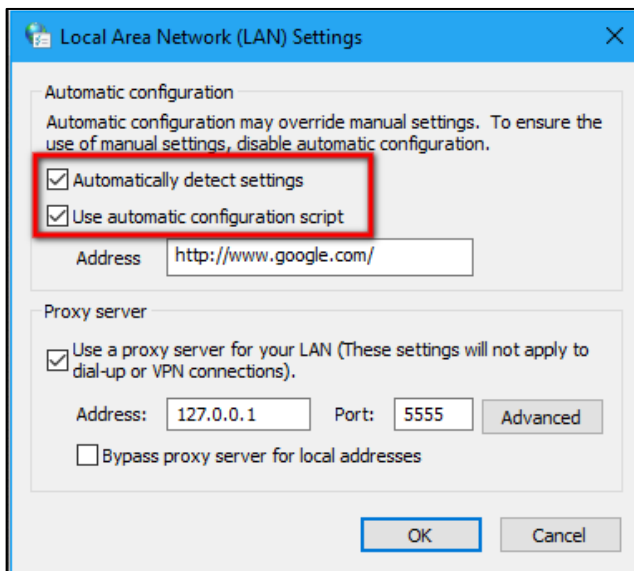
- In the dialog box, click LAN Settings.



- It will display another dialog box.

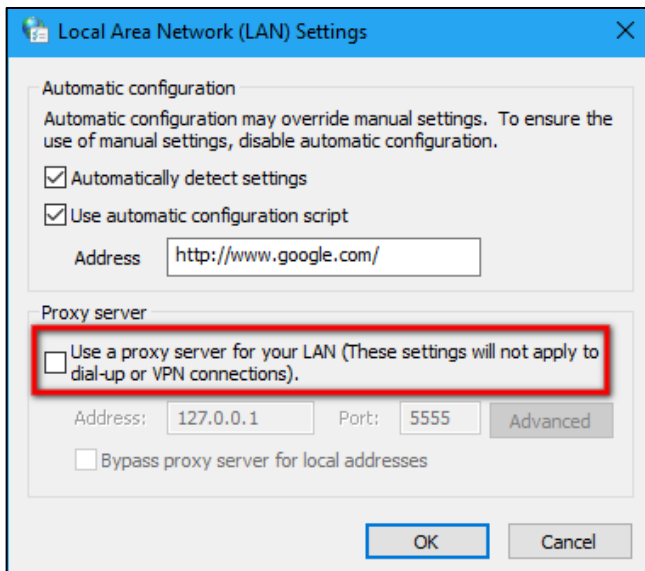


- In the dialog box, check the Automatically detect settings and Use automatic configuration script under Automatic configuration which will be unchecked earlier.

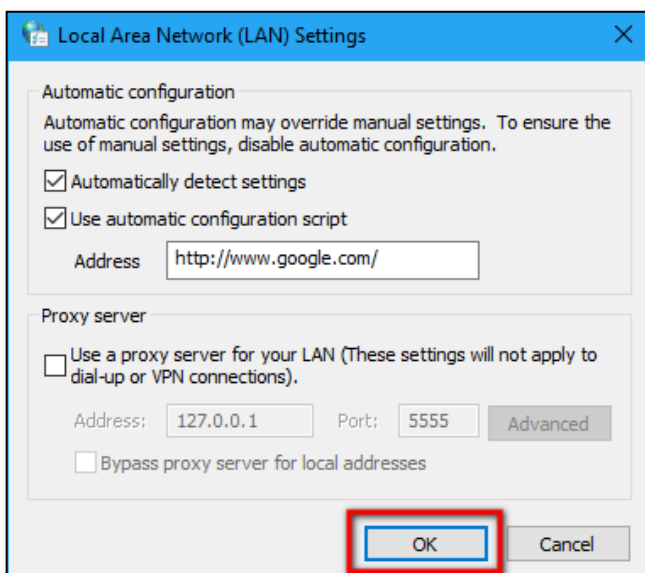


- In the same dialog box, uncheck the box under Proxy server which will be checked earlier.

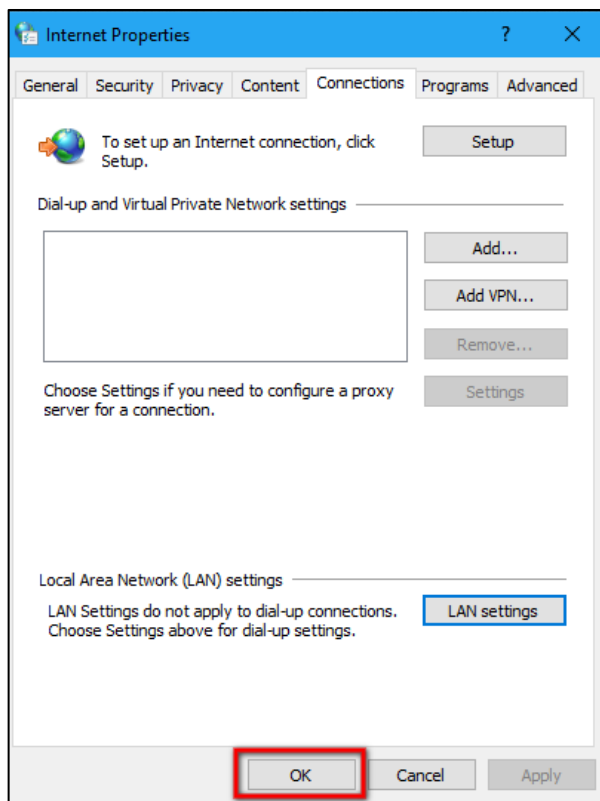




- After checking and unchecking the boxes, click Ok.

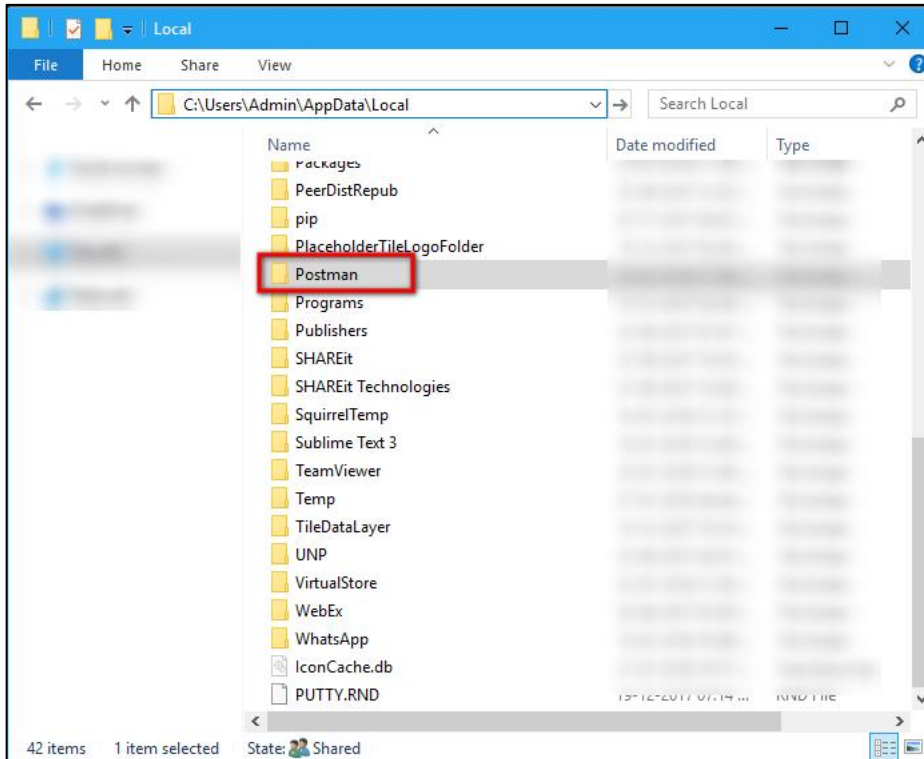


- Again, in the internet properties dialog box, click Ok.



## 6.3 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.

