EVERYDESKTOP – 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.
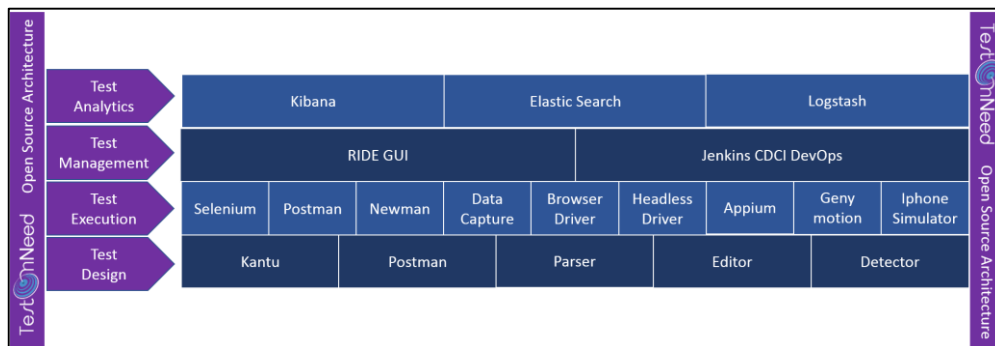
- This Open Source Ecosystem consists of **Test Design, Test Execution, Test Management and Test Analytics**.
- Each stage of Open Source Ecosystem uses set of open sources that are useful in their own way to achieve product quality.
- **Test Design**
  - o In this step, the development or creating of the test cases are done.
  - o Kantu, Postman, Parser applications, Editor and Detector are the open sources that are being used.
- **Test Execution**
  - o In the step, the execution of the developed or created test cases take place.
  - o 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**
  - o In this step, the test cases are given and modified the arguments and parameters that it produces desired result of the test cases.
  - o RIDE GUI and Jenkins are the open sources that are being in this step.
- **Test Analytics**
  - o Used in the process of analysis of data.
  - o 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.

- **UISpy**

    Developers should use other tools such as Inspect.exe that are available in the Windows Software Development Kit (SDK).

- **Winium**

    Winium is a new open source framework, that's based on Selenium and consists of 3 parts: Supports test automation on Windows applications. Supports test automation on Windows Phone operating system context (home button, notifications bar, toggles etc.)

- **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.

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**.

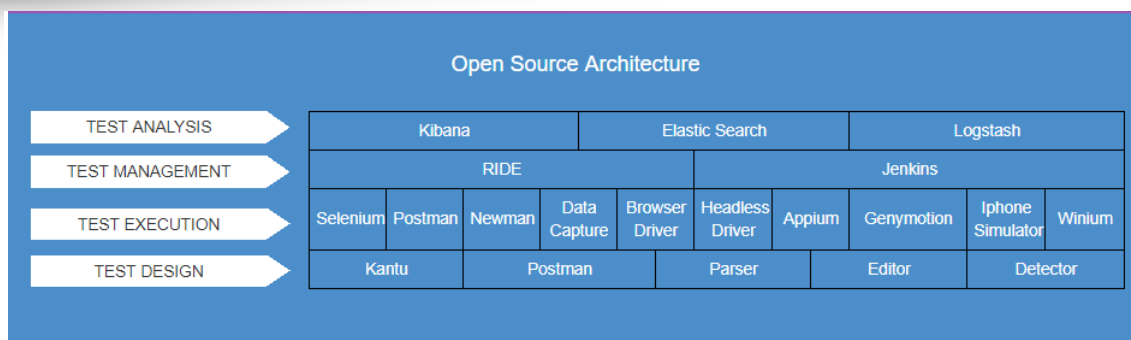- The outline of the Website looks like below:
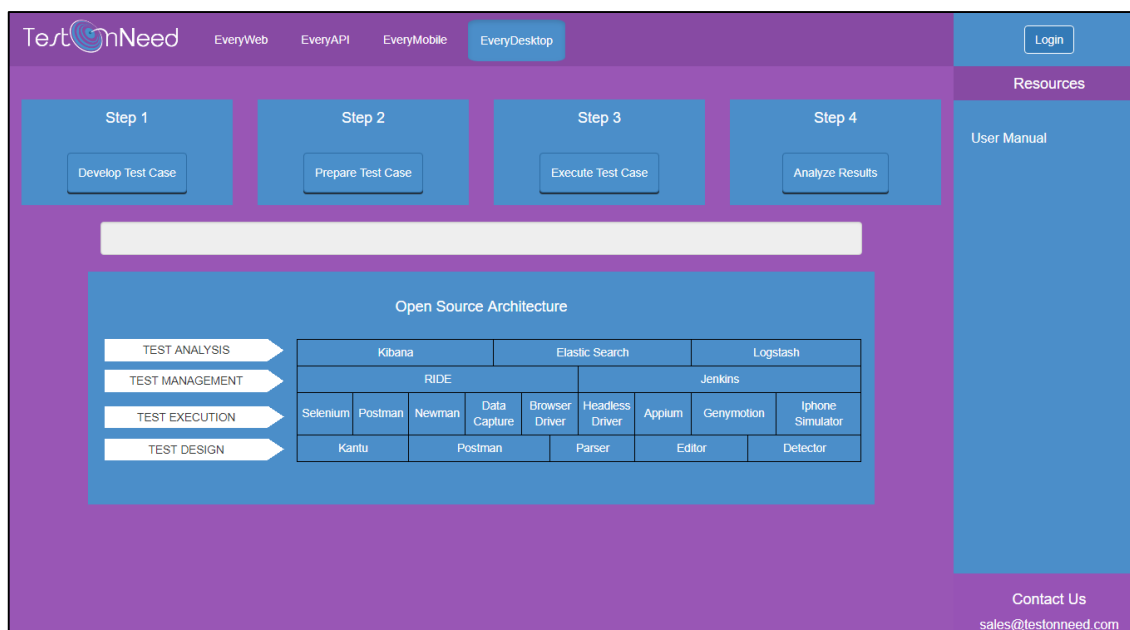
- The application has four tabs namely – EveryWeb, EveryAPI EveryMobile and EveryDesktop.
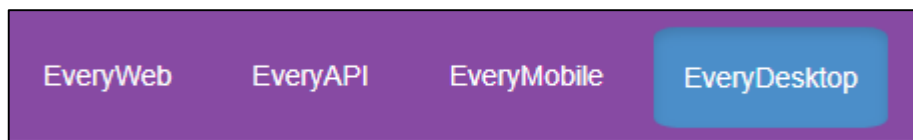

- Things that are common to all the tabs are the stack which contains all the open sources, a status bar and the user manual.

- 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.

Open Source Architecture

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

- 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.
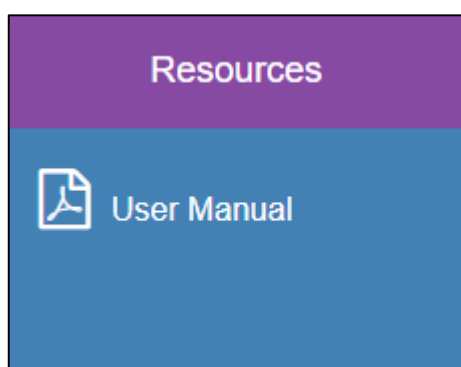
5. EVERYDESKTOP PAGE

EveryDesktop is a desktop application testing ecosystem using open sources. It is used to develop and execute the test case using detector and writing the code based on the detected elements using winium scripting.

The **Prerequisites of EveryDesktop** are the following:
- o The user should know winium scripting.
- o The user should know how to use the detector i.e. UISpy.

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

Step 1	Step 2	Step 3	Step 4
Develop Test Case	Prepare Test Case	Execute Test Case	Analyze Results

- **Develop Test Case**
  - o The user can write the test cases using winium scripting and UISpy.
- **Prepare Test Case**
  - o The developed Python file will be uploaded and in the background, the test case will get converted to its XML file. It will get saved with the uploaded Python file along with the XML file in a folder name with the same name as test case name.
- **Execute Test Case**
  - o The user can execute the automated test cases that was developed in the first step.
- **Analyze Results**
  - o The user can analyze the results visually with the help of graphs.
- Each step is explained below in detail

5.1 DEVELOP TEST CASE

The Develop Test Case is used to develop a test cases for the desktop.

1. Click the **Develop Test Case**.
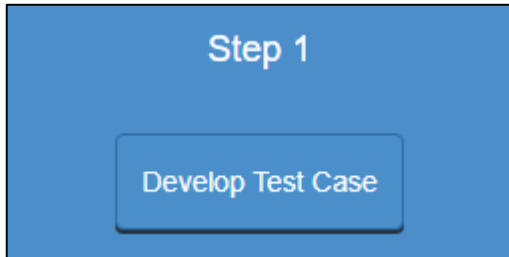
2. On clicking the **Develop Test Case** button, it will open the Notepad++ and UISpy.
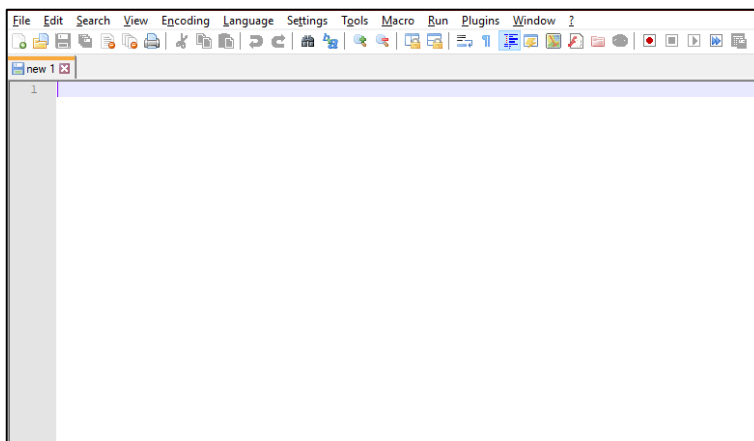

NOTE: The Notepad ++ should be saved in the C: drive then only it will open. For the installation process, please refer the installation manual.
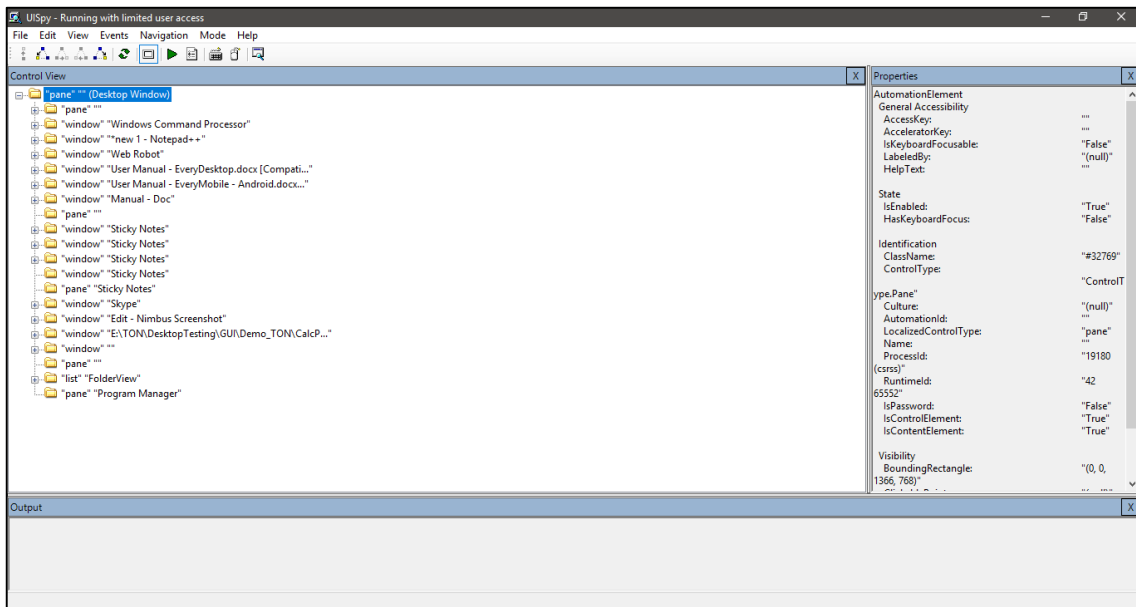
3. Write the test case code for executing the test case. For writing the code, please <u>refer</u> in the Reference section.

4. After writing the code, click Save button.
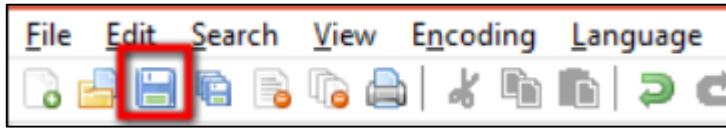
5. Surf to the location where file has to be stored. Example shows folder location as Documents.

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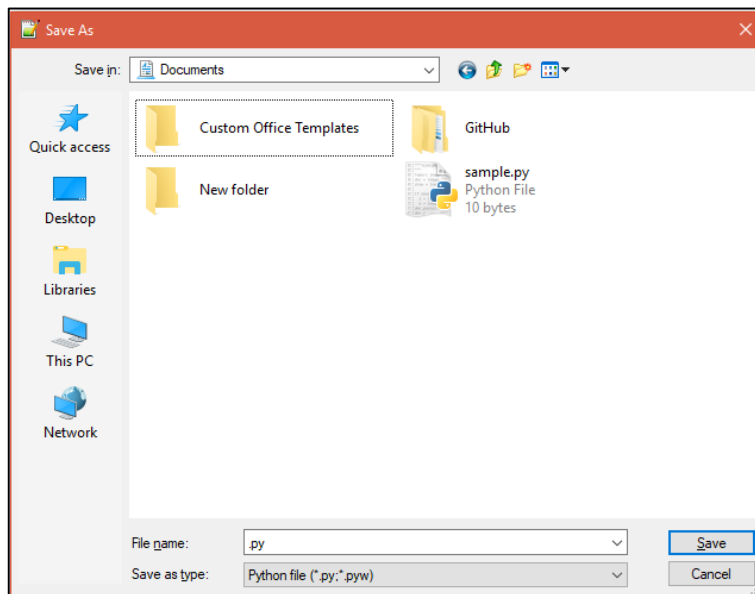
7. Click Save to save the file.

8. Now save the file based on the type of testing functionality. There are 2 types of testing functionality– **Function and Automation.**

9. In the case of **Function**, the test case should be saved as **DF_<TestCase_Name>**. For example, the test case should be saved as **DF_Calc_001.**
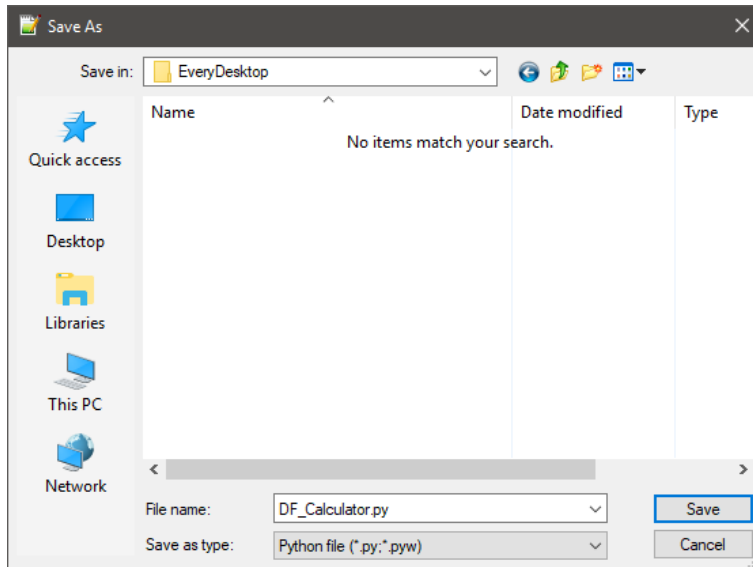
10. In the case of **Automation**, the test case should be saved as **DA_<TestCase_Name>**. For example, the test case should be saved as **DA_Calc_001**.
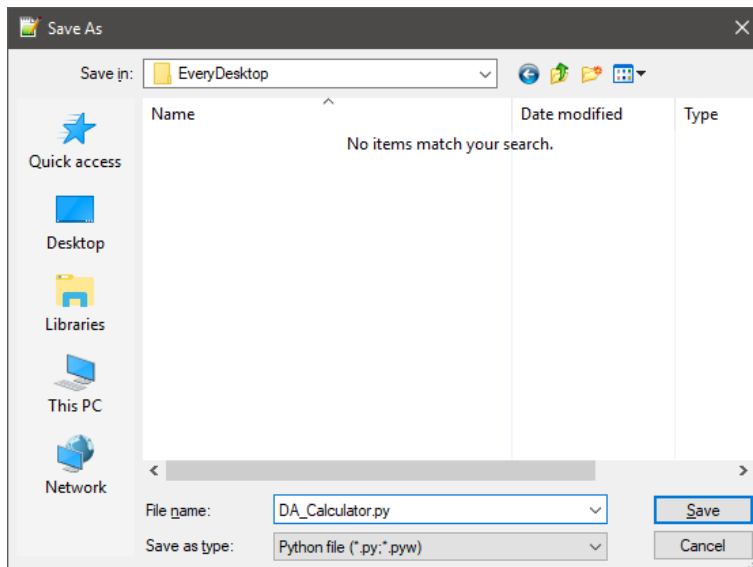
5.2 PREPARE TEST CASE

The user will save the Python file after writing the test case using Notepad ++ and UISpy. The user will upload the saved Python file in this step. It will convert the uploaded Python into its corresponding Python run file and its XML file.

1. Click the **Prepare Test Case** button.
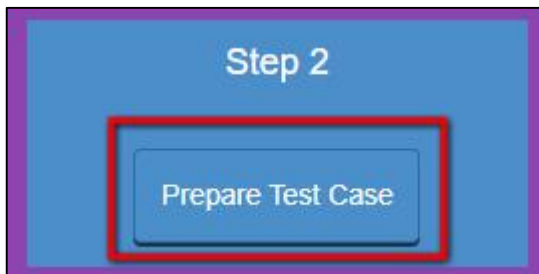
2. On clicking on the button, a pop up appears to choose the file.
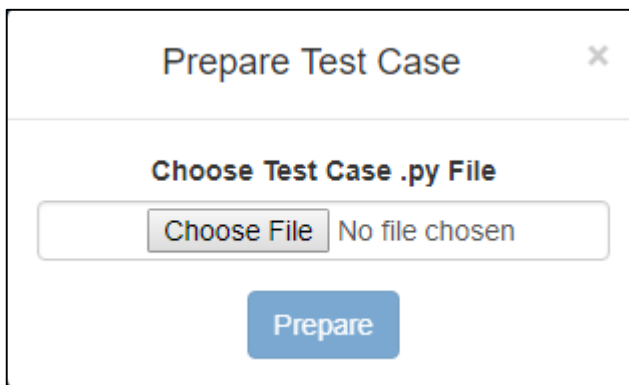
3. From the pop up, click on the Choose File button.
4. Surfing the folder location where saved the Python file that was developed and saved in the previous step.
5. Now upload the Python file.
6. The user can upload **only one file** at a time.

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

Prepare Test Case   ✕

Choose Test Case .py File

[Choose File] CalcPythton.py
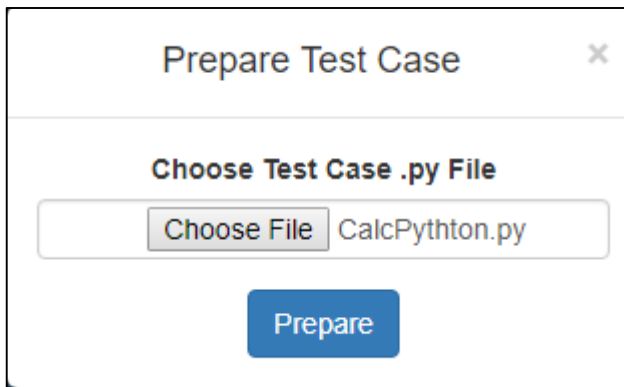
[Prepare]

8. On uploading Python 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 > DesktopTesting > GUI > Demo_TON > <TestCase_Name_Folder>**
- **> TON > DesktopTesting > TestOps > Demo_TON > <TestCase_Name_Folder>**
- In case if the test case name was saved as **DF_Calculator** in the previous step then it will create a folder with same name **DF_Calculator.**
- In this folder, the python file that was uploaded and its corresponding Python run file and the corresponding XML 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.

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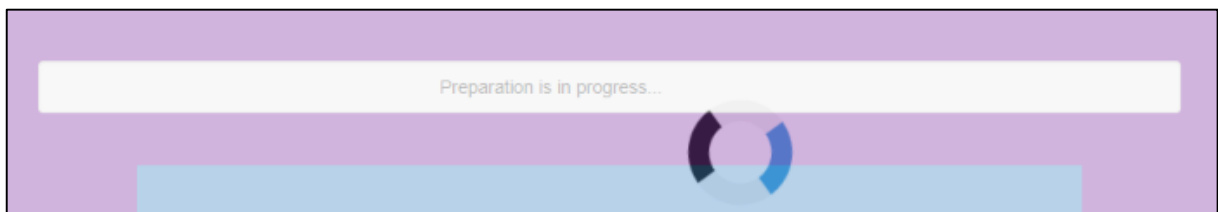
10. Once preparation is finished, a message will be shown as **"Preparation has been completed"** in the status bar.

Preparation has been completed
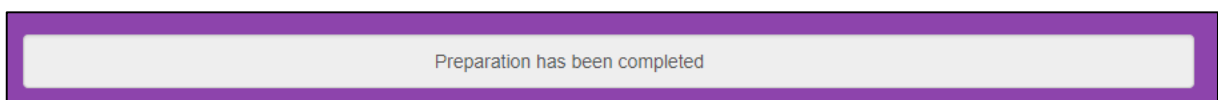
5.3 EXECUTE TEST CASE

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 two types of testing that can be performed under GUI and TestOps – Functional, and Automation 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.

1. Click on the **Execute Test Case** button.
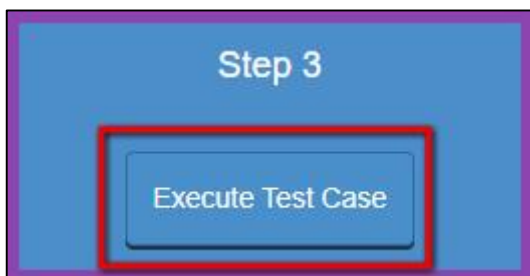
2. On clicking on the button, a pop up appears.
3. From the pop-up, select the button either as **Via GUI, Via TESTOPS.**
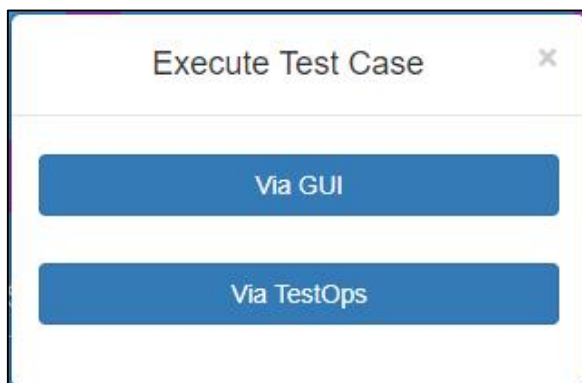
4. Clicking on the execute button, winium will start as a background process.

5.3.1 EXECUTE – VIA GUI

Here the user will get to know how to execute the Python file saved and prepared in the previous steps. Now let's see how to execute the test case using the GUI option.

1. Click the **Via GUI** button.
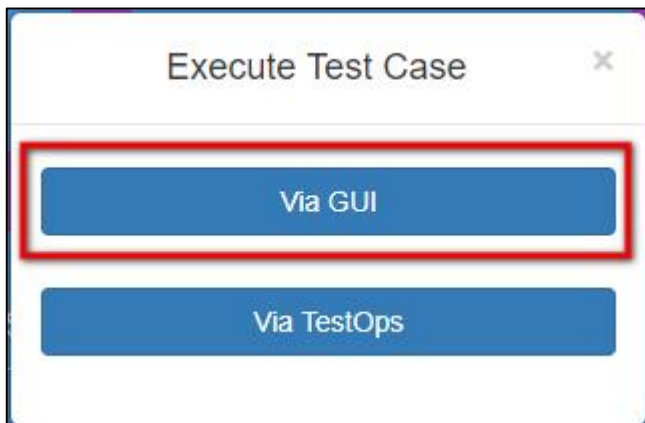
2. On clicking the button, execution through GUI can be done using 2 buttons– **Functional and Automation**.
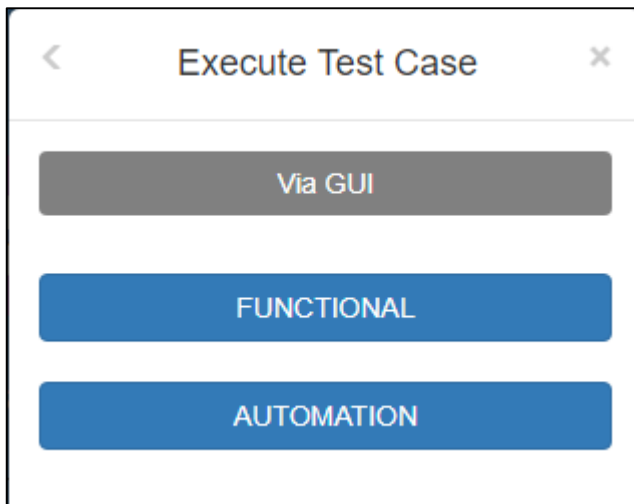
5.3.1.1 VIA GUI – FUNCTIONAL TESTING

Functional testing is a software testing process used within software development in which software is tested to ensure that it conforms with all requirements.

1. To perform functional testing, press **Functional** button.
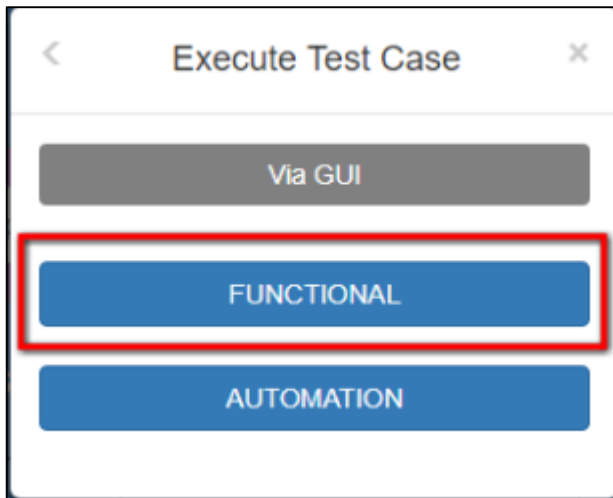
2. On click of the button, **RIDE** will open.
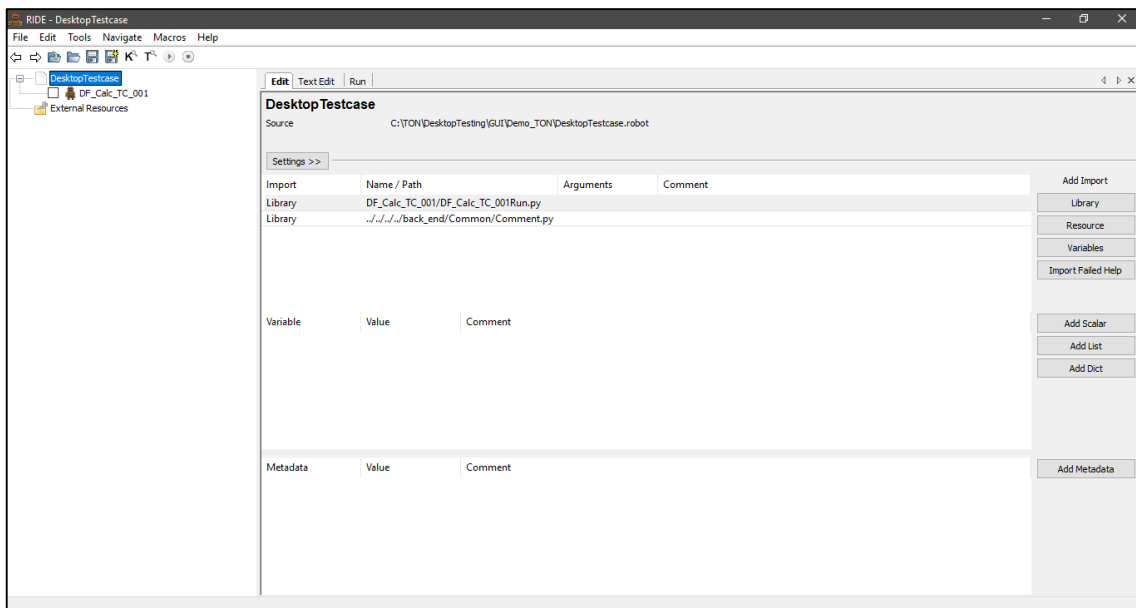
3. On the left panel, select one test case to execute.
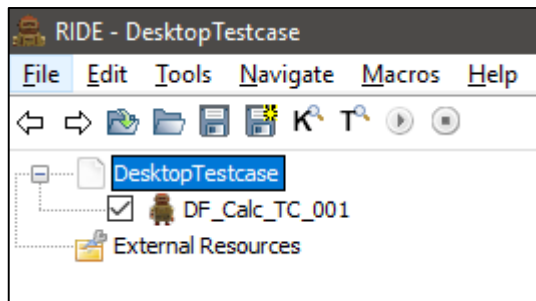
4. Click on the test case name.

5. On click of the test case name, Edit tab opens up.

6. Now click on the Run tab.
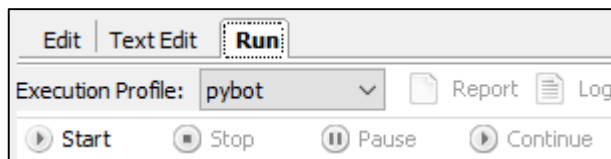
7. Configure the **Arguments** textbox by entering:

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

    white:white:white

8. Under Run tab, click **Start** button to start the execution.
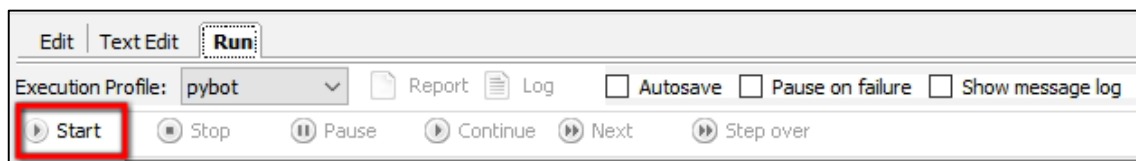
9. Click on the **Log / Report** button to view the result. The button becomes active on executing the test case.

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

11. 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 VIA GUI – AUTOMTION TESTING

Test automation is the use of special software to control the execution of tests and the comparison of actual outcomes with predicted outcomes.

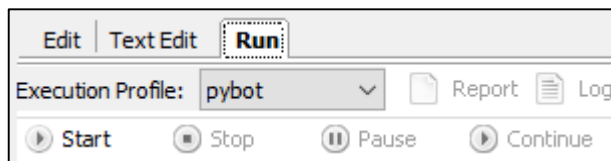1.   To perform automation testing, click **Automation** button.
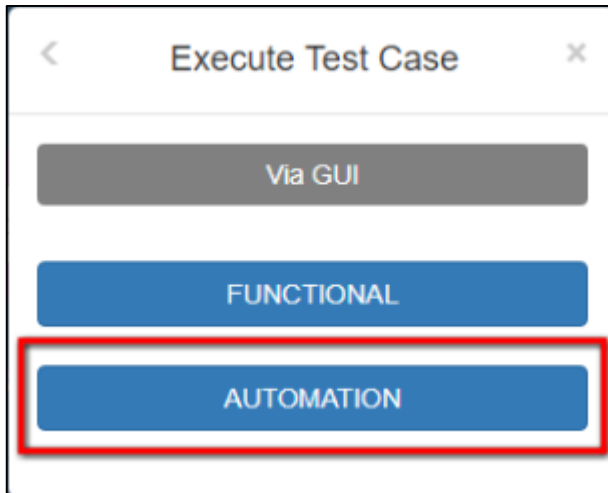

2.   On click of the button, **RIDE** will open.
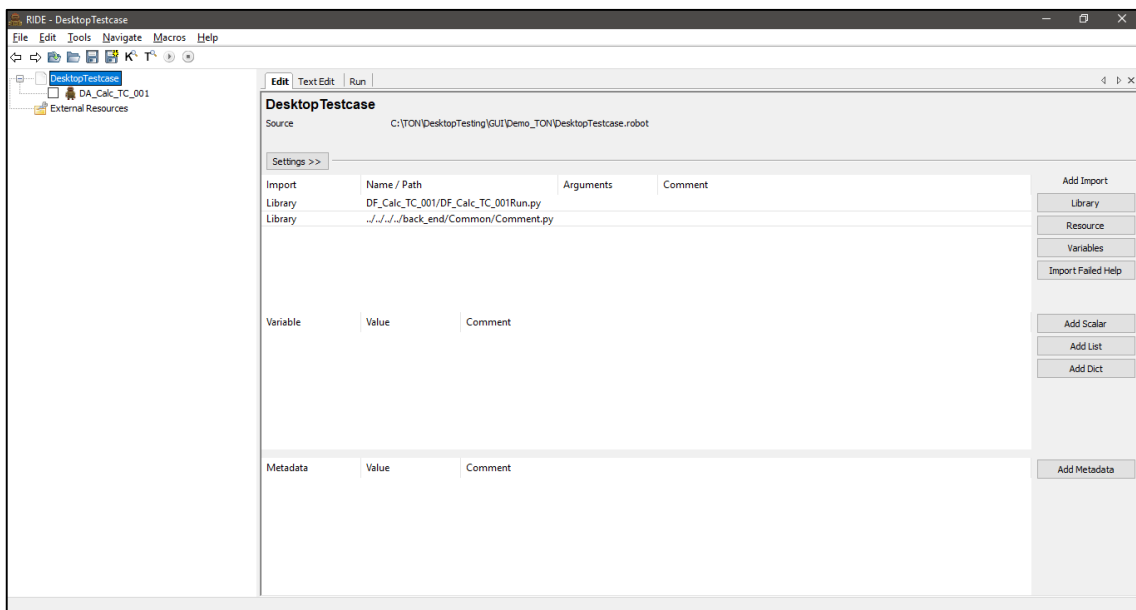
3. On the left panel, select multiple test cases to execute.
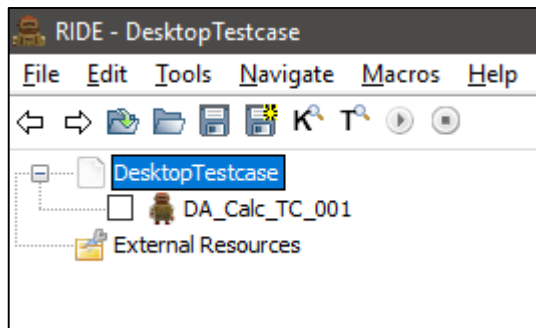
4. Click on the test case name.

5. On click of the test case name, Edit tab opens up.

6. Now click on the Run tab.

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

   -d..\\Demo_TON\<test_case_name>\Results –timestamp –reportbackground

   white:white:white

8. Click on the Start button to start the execution, under the arguments textbox.
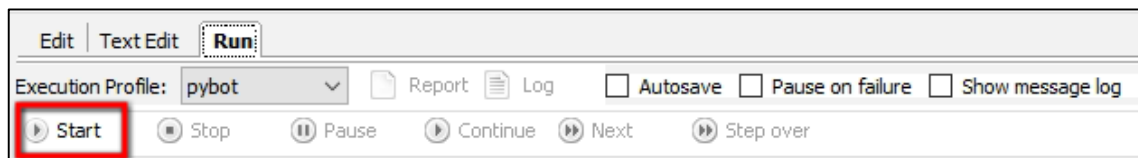
9. Click on the Log / Report button to view the result. The button becomes active on executing the test case.

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

11. In the browser, on the top right corner, the log button is there. Clicking on it, will show the log and to go back to the report click on the Report button.

5.3.2 EXECUTE - VIA TESTOPS

The concept of **TestOps** can be broadly **defined** as a process of using a combination of engineering artifacts, test artifacts and field artifacts and applying the process of software analytics to improve the V&V (Verification and Validation) strategy.

Now let's see how to execute the test case using the TestOps option.
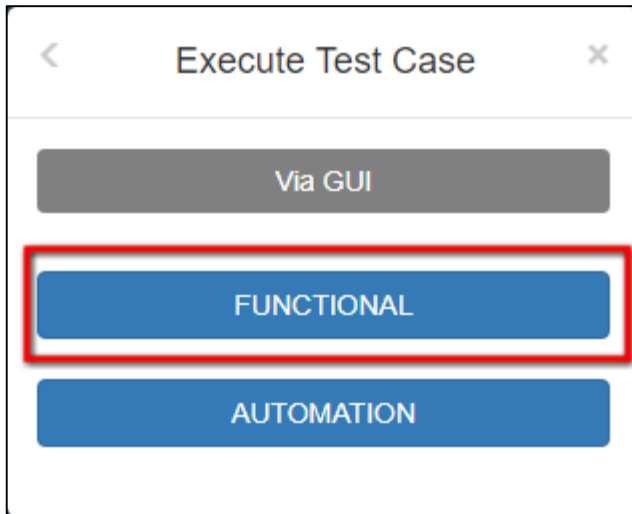
1. Click the Via TestOps button.
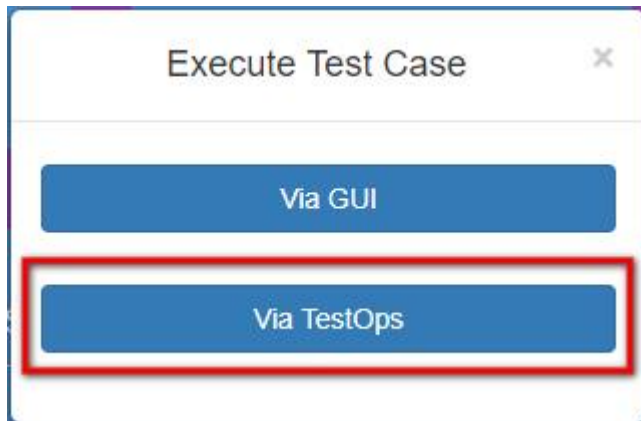
2. On clicking the button, 2 buttons will appear – **Functional and Automation**.
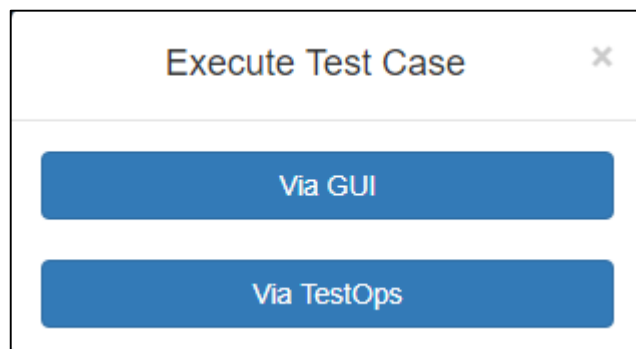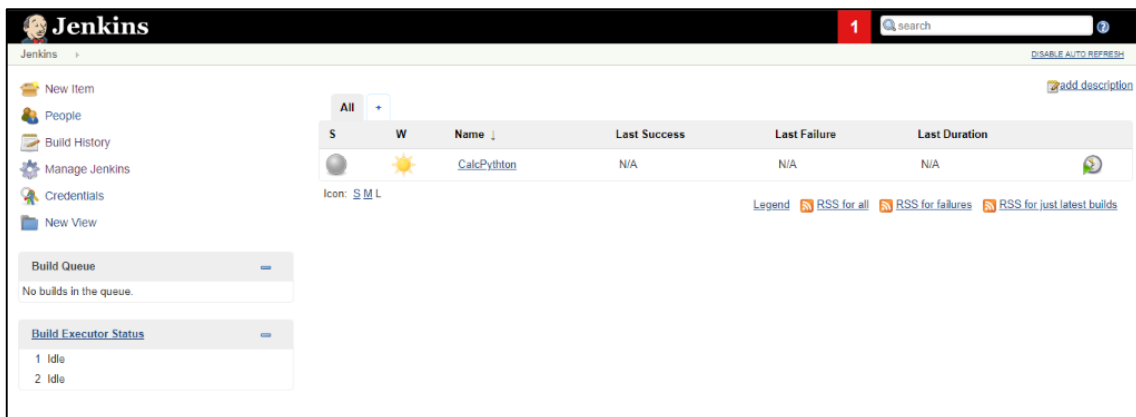
5.3.2.1 VIA TESTOPS – FUNCTIONAL
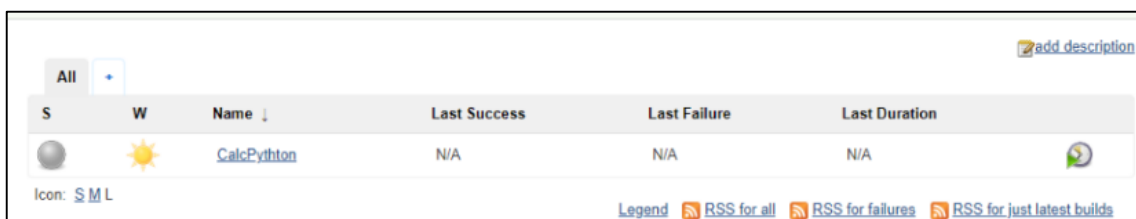
1. Click on the **Functional** button.

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

3. On opening the Jenkins, the prepared test case will automatically be present there as a new job.
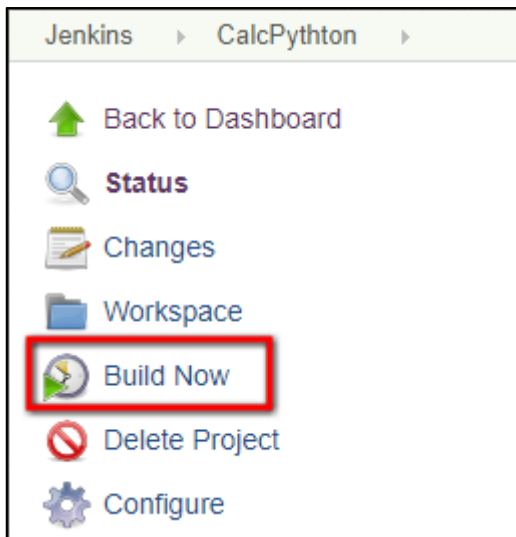

For creation of new <u>View</u>, refer the Reference Section below.

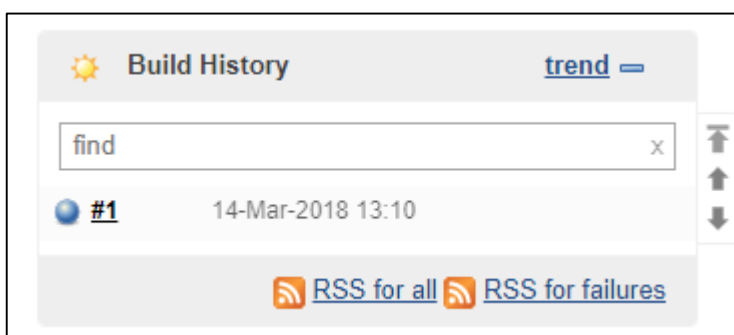4. Click on the job that has to be executed.

5. On the left side, select Build Now.

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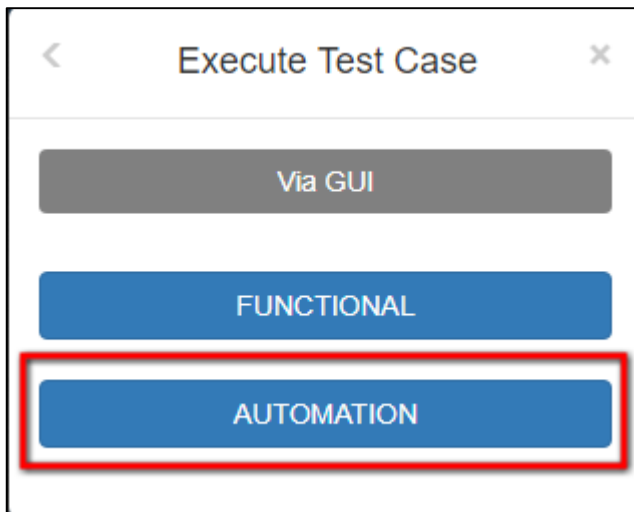

7. If the job is successfully built, it will show the built time, number of times of build and status of the built.

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

> **TON > DesktopTesting > TestOps > Demo_TON > <Job_Name_Folder> > <Test_Case_Name>.xml**

5.3.2.2 VIA TESTOPS – AUTOMATION

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

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

3. On opening the Jenkins, the prepared test case will automatically be present there as a new job.

For creation of new <u>View</u>, 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.
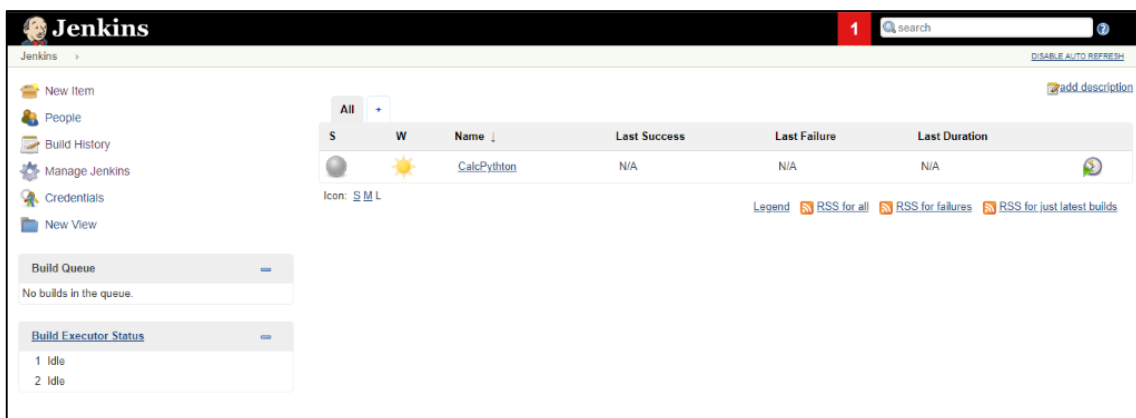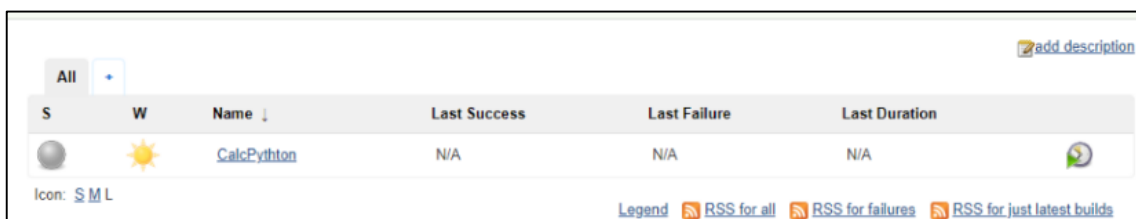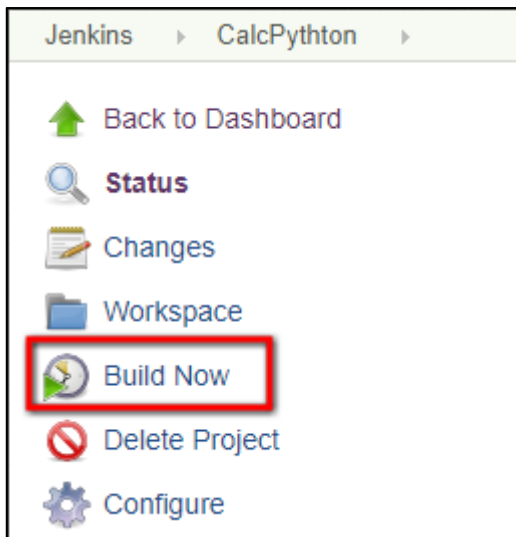

8. All the jobs of the Jenkins will be saved in:

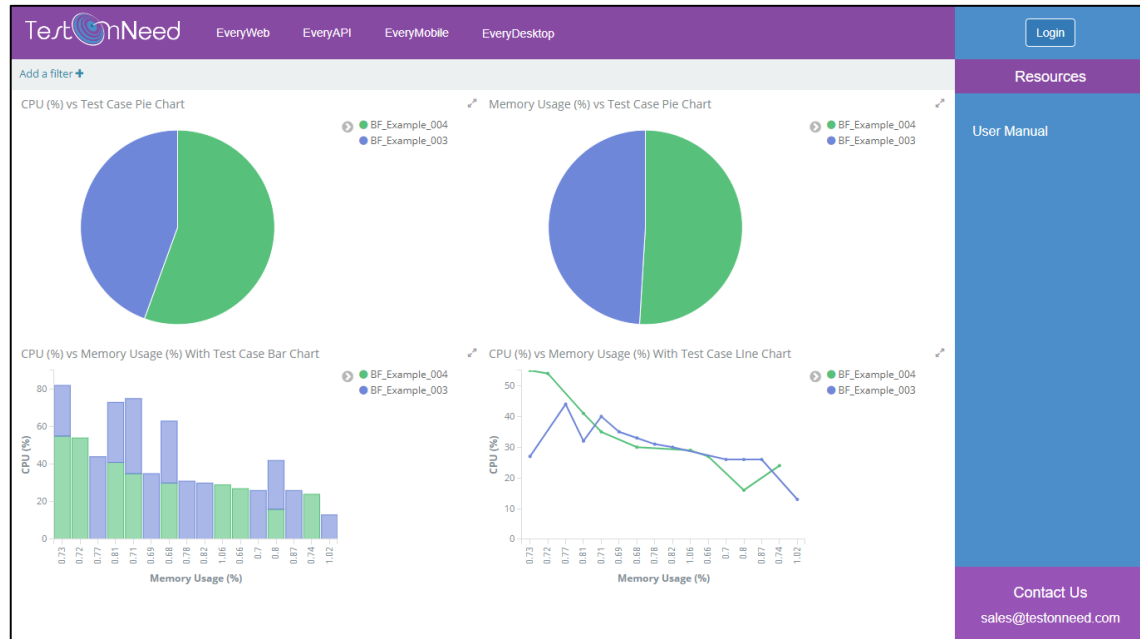> **> TON > WebTesting > API > TestOps > Demo_TON > <Job_Name_Folder>**

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 tab.
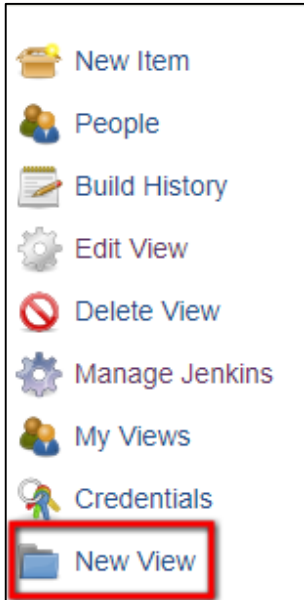

3. It will display 4 charts – 2 pie charts, one bar graph and one line graph.
4. The 2 pie charts will be – CPU vs Test Case and Memory usage vs Test Case.
5. The bar graph and the line graph denote the CPU vs Memory usage with Test Case.
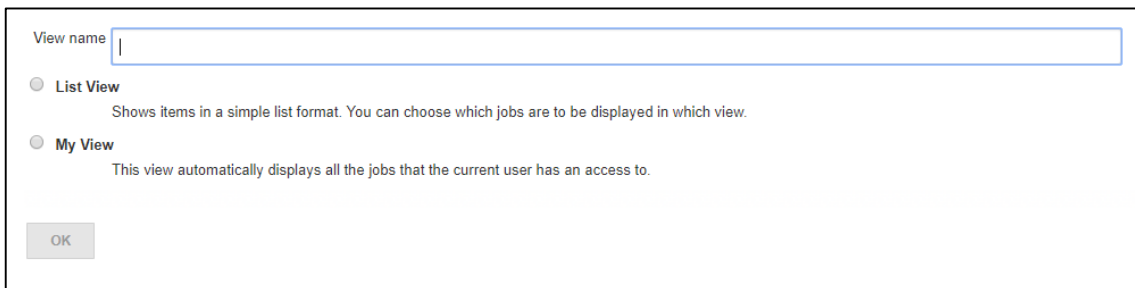
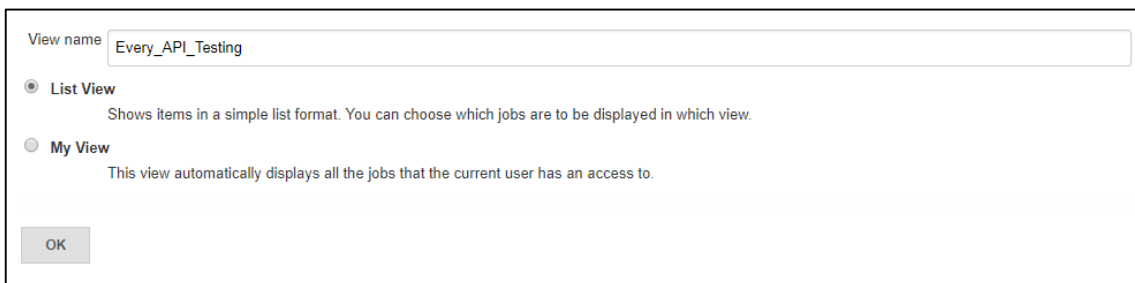# 6. REFERENCES

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.

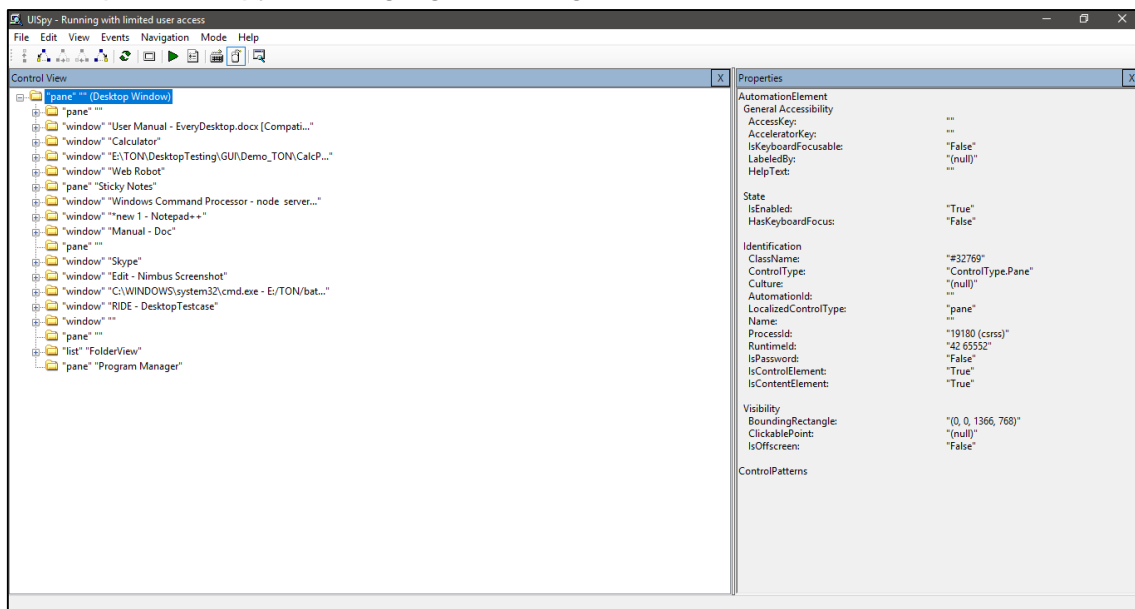## 5.2 WRITING THE CODE

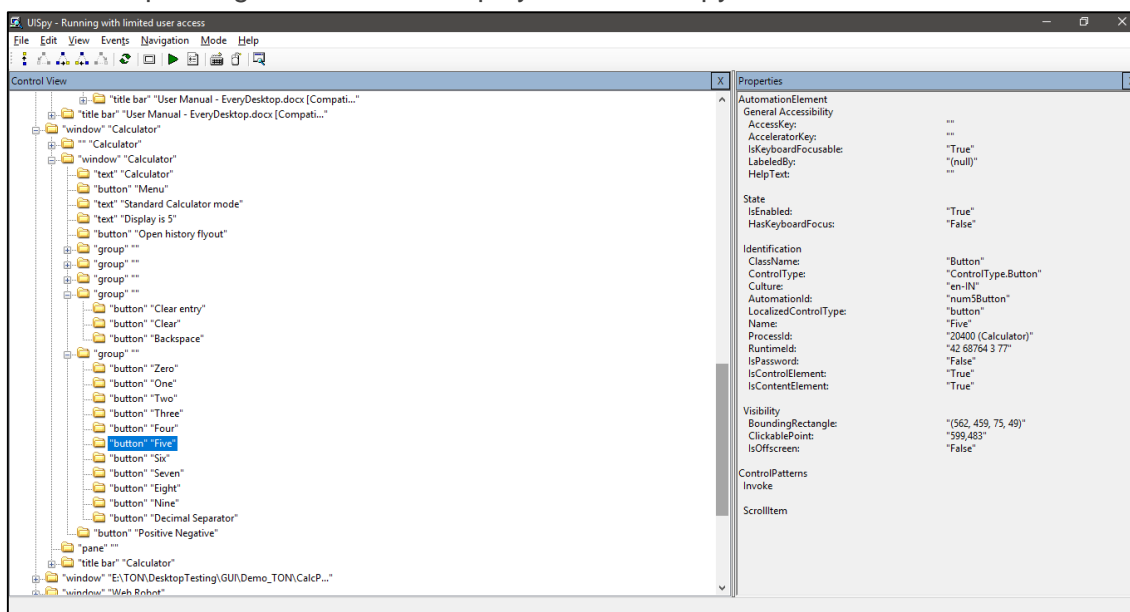1. Open the application which has to be detected. Example shows calculator app.

2. In the opened UISpy, click Highlight Rectangle and Mouse icons.

3. Pressing the Ctrl button down and mouse hover on the item to be detected.

4. The corresponding details will be displayed in the UISpy.

5. On the right side, under Properties, the Automation Id and Name will be found.

```
Properties                                              [X]
AutomationElement
 General Accessibility
  AccessKey:                     ""
  AcceleratorKey:                ""
  IsKeyboardFocusable:           "True"
  LabeledBy:                     "(null)"
  HelpText:                      ""

 State
  IsEnabled:                     "True"
  HasKeyboardFocus:              "False"

 Identification
  ClassName:                     "Button"
  ControlType:                   "ControlType.Button"
  Culture:                       "en-IN"
  AutomationId:                  "num5Button"
  LocalizedControlType:          "button"
  Name:                          "Five"
  ProcessId:                     "20400 (Calculator)"
  RuntimeId:                     "42 68764 3 77"
  IsPassword:                    "False"
  IsControlElement:              "True"
  IsContentElement:              "True"

 Visibility
  BoundingRectangle:             "(562, 459, 75, 49)"
  ClickablePoint:                "599,483"
  IsOffscreen:                   "False"

ControlPatterns
 Invoke

 ScrollItem
```

6. Both the Automation Id and Name is used in the process of writing winium code.