
RobusTest 2.0 Documentation Documentation

Izinga Software Pvt Ltd

Jul 31, 2023

Contents

1	Introduction	3
2	Getting Started	5
3	Project	9
4	Project Dashboard	11
5	Manual Testing	15
5.1	Device Configuration Menu	15
5.2	Session Configuration Menu	16
5.3	Multi-device testing	18
5.4	Test Embedded Video	20
6	RobusTest Automator	21
6.1	Recording user actions	21
6.2	Test Step Management	23
6.3	Test Session - Device Management	25
6.4	Test Session - Test Case Management	31
6.5	Test Session - Session Management	43
6.6	Creating and Managing Test Cases	44
7	Executing Test Runs	47
7.1	Understanding Test Suites	47
7.2	Creating and Executing a Test Run	48
8	Run Settings	51
9	Live View	53
10	Performance Testing	55
11	Automation Reports	57
11.1	Debugging Test Case Failures	57
12	Advanced Automation Concepts	59
12.1	Verification	59
12.1.1	Using the ‘Verify Element’ option	59

12.1.2	Using the ‘Verification’ button option	67
12.1.3	Verify that an element exists	73
12.1.4	Invert Verification	79
12.2	Functions	80
12.3	Parameterisation of Data	83
12.3.1	Mapping to Step	83
12.3.2	Mapping to Data Set	83
13	Scheduling your tests	87
14	RobusTest Hub	89
14.1	Appium Hub	89
14.1.1	Run Appium Tests	89
14.1.2	Organize Appium Sessions Into Test Cases	90
14.1.3	Appium Test Data	91
14.1.4	Appium Hub to Find Locators	92
14.1.5	Running tests for your Unity based app on the RobusTest Hub	93
14.2	Run Espresso Tests	93
14.3	Run XCUITest Tests	95
14.4	Run Selenium Tests	96
15	User Profile	97
16	Health page	101
17	Admin Console	103
18	Continuous Integration	109
19	Integrating Bug Tracker	111
20	Other Useful Information	113
20.1	Adding new devices to RobusTest - Android	113
20.2	Adding new devices to RobusTest - iOS	115
20.3	Best Practices in Automating Tests using RobusTest	116
21	RobusTest Connect - Run local, test global	119
22	Troubleshooting	123
22.1	Unable to access RobusTest Server	123

Contents:

CHAPTER 1

Introduction

RobusTest is the app testing platform for all things mobile. Internally, we like to call it the Swiss Knife for Mobile App Testing.

RobusTest is designed to be a one stop platform for everyone involved in creating a mobile application - right from business stakeholders who need to see what their application looks like, to developers who need to run some quick checks to testers who need to test the app extensively using manual and automated methods. Some important features of RobusTest are

- Testing on real devices
- Performance-first approach to manual and automated testing
- Simulate real life scenarios for meaningful testing
- Automate your tests and make them data-driven without writing any code
- Run your automated tests on multiple devices in parallel
- In-depth test reports including both functional and performance test results
- Run your own automated tests on the devices connected to the platform

For any questions on the documentation, feel free to write to us at support@robustest.com

CHAPTER 2

Getting Started

Accessing RobusTest

Users can start using RobusTest by signing up at <http://robustest.com>

Enterprise users can also have an on-premise instance of RobusTest installed. For more information on our enterprise offering, do write in to us at hello@robustest.com

If you are using the public instance at <http://robustest.com>, you can easily create a trial account for yourself from the Sign Up page. Please note that you can sign up on the public platform only with a business email address. A trial account has 60 minutes of device usage credits. Your account is associated with your organization based on your email address domain name. Note that in case of trial accounts, there is an overall cap of 180 minutes on the organization. This cap is not applicable in case of paid accounts.

Sign Up

Julia Roberts

julia@roberts.com

.....

.....|

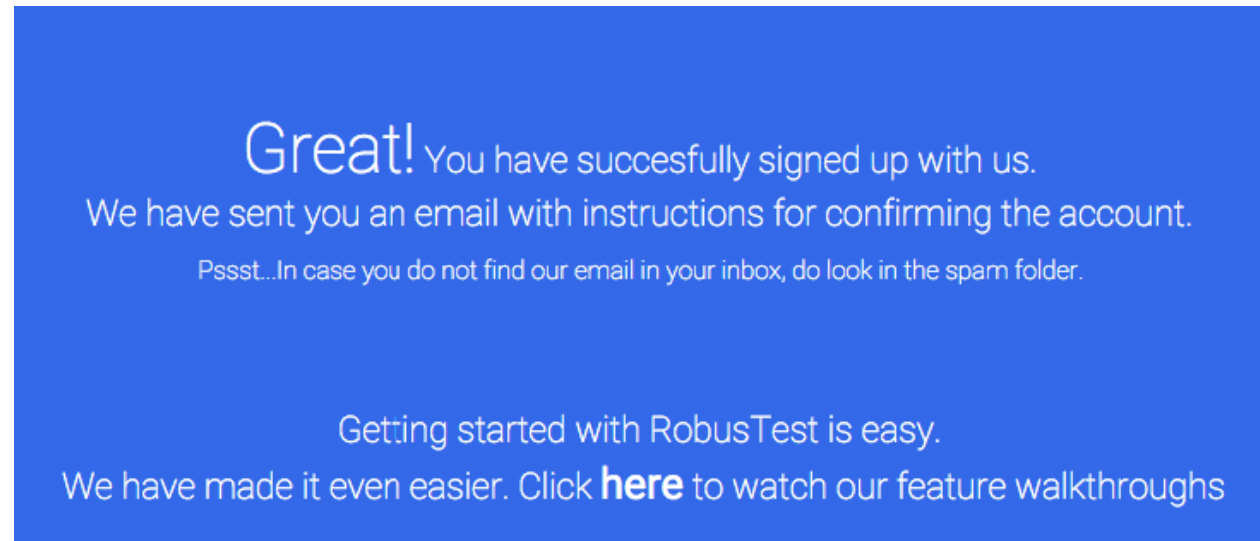
Great! we are ready to go.

SIGN UP

Already have an account ? [Sign In](#)

Sign Up is allowed only with a business email address. Your email address will be safe with us. Upon sign up, 60 minutes of device usage are automatically credited in your account. Your credits will be subject to the overall organization credit limit which is 150 minutes by default.

On Sign Up, you will need to confirm your account.



In an on-premise enterprise instance, the process of signing up may differ e.g. in case of SSO integration, you can just login using your existing credentials. You can get more details from the administrator for your enterprise installation.

On RobusTest, all activity related to the testing of a mobile app will be part of a *Project*.

A Project is a logical place for all builds, test cases, test suites, test runs and test reports associated with the testing of the app. It also facilitates collaboration within the team.

Creating a project

1. Once you have logged into RobusTest, you will find a 'Create Project' button on the landing page. To create a project, click on this button.
2. On the 'Create New Project' page that opens up, you need to enter the following information:
 - *Project Name* - in case of app testing, this is usually the name of the app under test (AUT). This field is mandatory
 - *Project Description* - provide a brief description of the purpose of creating this project
 - *Project Type* - you have the option to create one out of 4 types of projects:
 - a. *Android App Project*
 - b. *iOS App Project*.
 - c. *Mobile Webapp Project*
 - d. *Device Only Project*

You can find more information here: [project-types](#)

Create New Project

Project Name

First letter should be letter

Project Description

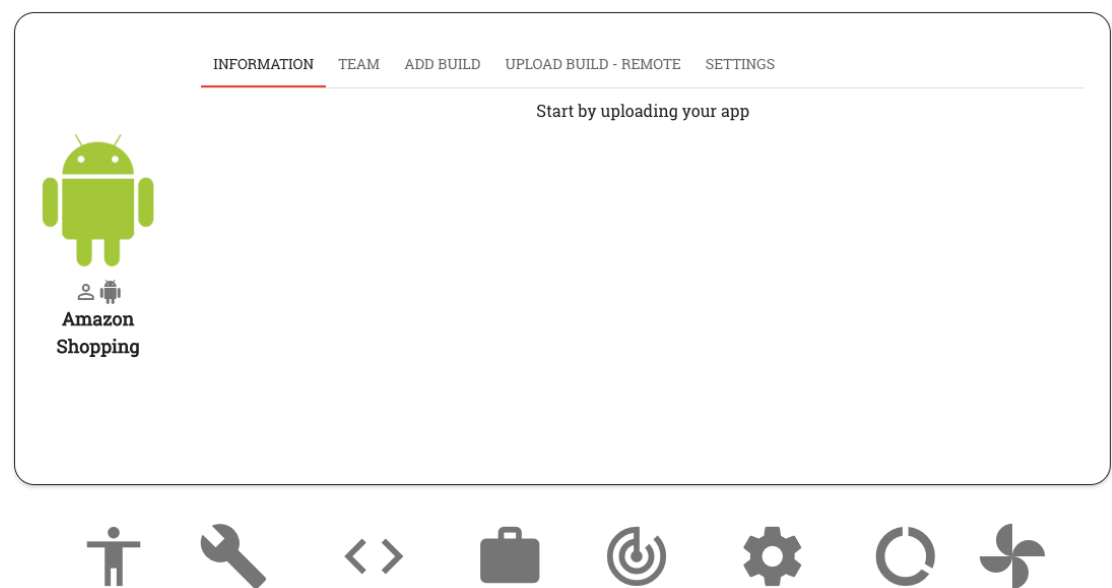
Project Type

GO BACK

CREATE PROJECT

3. Click on the 'Create Project' button. The project has now been created and you will find yourself on the Project Dashboard.

E.g., after creating an Android project the project dashboard looks like below image.



4. Now that you have created a project, the next step is to upload the app build to be tested * For an Android project, this will be an apk file * For an iOS project, this will be an ipa file

To upload a build, click on the 'Upload new build' button (i.e., the '+' icon) on the Project Dashboard

This step is not applicable to Mobile WebApp and Device Only projects

You are now all set up to begin testing

Before we do so, let's get to understand the Project Dashboard a little more

CHAPTER 4

Project Dashboard

The Project Dashboard provides you information about the project you have created - i.e., the app under test, the build details, project members, CI settings, etc.

The screenshot shows the 'BUILDS' tab selected in the Project Dashboard. The dashboard displays information for a build named 'Vodafone Callertunes v 3.1.7.5 # 215'. The build size is 14 MB. The build was added by Suraj Nair on 10 Jul 2020, 02:28:07. The build name is 'Vodafone Callertunes', the version name is '3.1.7.5', and the version code is '215'. The package name is 'br.com.vodain.ringbacktones'. The launch action is 'com.onmobile.rbt.baseline.activities.SplashActivity'. The dashboard also includes a navigation bar at the bottom with icons for Manual, Automation, Test Scripts, Test Suites, Test Runs, Run Settings, Device queries, and Hub.

BUILDS		TEAM	UPLOAD BUILD - REMOTE	SETTINGS
Build	Vodafone Callertunes v 3.1.7.5 # 215			File size 14 MB
Description				
Added by Suraj Nair		Added On 10 Jul 2020, 02:28:07		
Build Name Vodafone Callertunes		Version Name 3.1.7.5		Version Code 215
Package Name br.com.vodain.ringbacktones		Launch Action com.onmobile.rbt.baseline.activities.SplashActivity		

Manual Automation Test Scripts Test Suites Test Runs Run Settings Device queries Hub

Let's have a look at the Dashboard in detail.

The Project Dashboard constitutes of the following 5 sections:

1. Builds
2. Team
3. Upload Build - Remote
4. Settings
5. Group

1. Builds

This is the section of the Dashboard on which you land by default on clicking on a project.

The Build section provides you information about:

- *Build details* (i.e., build name, version name, version code, package name and launch action)
- *Description about the build*
- *Build file size*
- *Name of the team member who uploaded the build to the project*
- *Date of upload of the build to the project*

You can upload multiple builds to a project. You can choose the build to be tested upon from the 'Build' drop down field.

In addition to providing you the details about the build, this section of the Dashboard also allows you to perform the following tasks:

- **Upload New Build** - Clicking on this button enables you to manually upload an app build from a chosen location to the project
- **Get selected Build URL** - Clicking on this button copies to the clipboard, the URL to download the currently selected build in the project. On pasting this URL on a browser tab and hitting enter, the user is able to download the currently selected build onto their computer.
- **Copy the build ID** - Clicking on this button copies to the clipboard, the build ID of the currently selected build in the project.
- **Delete Selected Build** - Clicking on this button deletes the currently selected build from the project. Only a project member with admin privileges can delete a build.

2. Team

All members who are added to the project are listed on the Team section. Apps that are part of a particular project can be accessed by everyone who is part of that project.

Project members have the ability to test, automate and view reports for any build that is part of their project.

Each team has 2 kinds of members - Admin and non-admin

Admin members have a few additional privileges when compared to non-admin members. They can:-

- Delete a build from the project
- Add/remove members from the project
- Grant to/Revoke from a team member admin privileges
- Execute a CURL command to upload an app build remotely

By default, a member who creates a project has admin privileges on that project.

Note: An Admin member of a project is different from a RobusTest Admin. The former's privileges are restricted to the project the member is a part of while the latter has special privileges on the entire RobusTest platform.

3. Upload Build - Remote

This section is visible only to admin members of the project.

Here, a CURL command is provided which enables the user to upload an app build to the current project remotely.

The general format of this command is as follows:

```
curl -X PUT '<RobusTest URL>/v3/project/<PROJECT IDENTIFIER>/build?accesskey=<USER_
↪ACCESS KEY>' -H 'content-Type: multipart/form-data' -F 'build=@<BUILD NAME WITH_
↪PATH>' -F 'buildInfo={"desc":"<build description>", "label":"<label>",
↪"activityClass":"<launch activity>"}
```

Let's break down the different components of this command for better understanding:

- **RobusTest URL:** This refers to the URL that you use to access the RobusTest platform. It would be of the form *http://<RobusTest IP>* or *http://<RobusTest domain>* E.g. <http://robustest.this.instance.com:8085>
- **Project Identifier:** The 'Project Identifier' is how we identify the project to which we want to upload the new app build. This can be obtained as follows:
 - On RobusTest, go to the Project Dashboard of the project to which the build is to be uploaded
 - The URL displayed there is of the form *<RobusTest IP>/#/project/<Project ID>/dashboard*
 - Copy the *Project ID* value from above URL
 - E.g. if the URL says *http://robustest.this.intance.com:8085/#/project/5d176ffef0238be8f3b7afa5/dashboard*, then, the value *5d176ffef0238be8f3b7afa5* above, is your project ID
- **User Access Key:** Each user has a unique User Access Key associated with their profile on RobusTest.
By default, the user access key will be populated in the 'Remote build upload' command provided on the Project Dashboard. All builds will be uploaded using the RobusTest ID of the team member whose access key is being used.
Later, if you would like to upload the build using a different team member's RobusTest ID, you need to change the value of the 'User Access Key' in the 'remote build upload' command.
Please have a look at the [User Profile](#) page for more details on how to obtain the *User Access Key*.
- **Build Name With Path:** The path to the location from where the build can be picked is specified here. The app build present in the location or path mentioned in the command is uploaded to the project that is identified by the Project Identifier.
- **buildINFO section:** This section is used to provide additional details about the build such as description, label or launch activity. It is not mandatory.

A sample remote build upload command would look as follows:

```
curl -X PUT 'http://robustest.this.instance.com:8085/v3/project/
↪5d176ffef0238be8f3b7afa5/build?accesskey=aY33cDmkt7B2nAjxB16Tp2FWv4' -H 'content-
↪Type: multipart/form-data' -F 'build=@/username/build/new/latestbuild.apk' -F ↪
↪buildInfo={"desc":"description from api"}'
```

You can now run the above command directly on the Command Line OR choose to invoke this build-upload API through a programming script in a language of your choice.

E.g. you can add the above line to your Jenkins shell script that creates a new build. As a result, whenever a new build gets created, it also gets uploaded to the project. Using RobusTest, you can now build a process, say, to test this new build by running a sanity or smoke test each time a new build is uploaded to the project.

4. Settings

This section provides you the following options:

- Enable notifications - On enabling this checkbox, each member of the team is notified whenever a new build is uploaded to the project
- Choose Bug Tracker configuration - RobusTest supports Continuous Integration with your existing CI tools through APIs.

Once you have integrated your Bug Tracker tool with RobusTest, this configuration will be available for selection in the 'Bug Tracker' drop down. Once the required configuration is selected, all bugs encountered during your testing can be logged directly, from RobusTest, into the tool of your choice.

You can configure your project with the tool of your choice through the 'Integration' section of the RobusTest Admin Console.

5.1 Device Configuration Menu

This is the horizontal menu that you find in your test session page. It provides you options to perform various configurations related to the device. Let's have a look at each of them.

1. Location Simulation

This feature allows you to test as if the device is present at a different location than where it actually is. This is done by simulating the location on the device.

Pre-requisites:

On the device:

1. in Developer options: * enable 'Mock Locations' * set the Nizedha app as your mock location app
2. in 'Locations', set the 'Location Mode' to 'Device only'

Once the pre-requisites have been met, you can simulate any location as follows:

1. Click on the 'Simulate Location' button
2. Type the name of the location in the 'Search location here' field and select from the drop down. Alternatively, you can manually pin the location of your choice on the map
3. Once your location has been pinned, click on the 'Set Location' button

Your device will now behave as if it is situated at the location chosen by you

2. Run ADB commands on device

Sometimes, as part of your testing, you may want to run a few ADB commands on the device under testing. You can do so by clicking on the 'Run ADB commands on device' button.

This brings up the Command Line Interface from where you can execute adb commands directly on the device.

3. Enable/Disable Navigation menu on device

This button enables/disables the Android navigation menu bar at the bottom of the device screen.

4. Device screen size configuration

These buttons enable you to increase the screen size, decrease the screen size and to reset the device screen size to its default setting. This is useful in case you want to look at the app in more detail and want to verify the rendering of various objects on the screen.

5. Device screen ratio configuration

These buttons enable you to increase or decrease the screen ratio.

This feature is helpful in scenarios, where the network bandwidth is low and the user would still like to test.

Increasing the screen ratio decreases the resolution of the device screen and enables the user to continue testing at a lower bandwidth.

Decreasing the screen ratio enables the user to test on the device screen at higher resolution.

6. Device screen rotation

You can test the device in Landscape and Portrait modes by rotating the device screen using these buttons

5.2 Session Configuration Menu

This is the vertical menu that you see in the Manual test session. It provides the user information that would aid in the testing process. Let's go through each of them.

1. Session Information

Clicking on this button provides you all details regarding the manual test session in progress. This includes information about:-

- a. the build - App name, build version, last uploaded, etc
- b. the device - Device name, ADB ID, OS version, etc,
- c. the project - Project name, Last build uploaded, etc

2. Focused View

Clicking on this button provides you a distraction-free view of the device screen so that you can focus solely on your testing

3. Create & View Contacts

The 'Contacts' feature is useful in use cases where you need to make calls or send SMSs to specific contacts or phone numbers.

You can add names to the contact list and specify their contact numbers. Now you can call or send an SMS to these contacts from the device on which you are testing.

4. Create & View Notes

The 'Notes' feature enables you to save tidbits of information that may be of use to you at a later point in time or perhaps information which you may want to share with your project members. The notes you create are available to any project member even after the end of the test session in which the note was created.

5. Execute Deeplink

This feature enables you to test deeplinks being used in your app. Provide the deeplink and, if required, the package name. Then click on 'Execute'. You can now see the deeplink being executed on the device

6. Copy from device clipboard

Sometimes the need arises where you may want to copy data from the device screen, say, - maybe a text in an app page or a URL opened on a browser on the device - to your own computer or laptop.

In such cases, do the following:

1. Copy the text you want to retrieve onto the device's clipboard by selecting and long pressing it
2. Now click on the 'Copy from device clipboard' button in your Manual test session. This data is now available on the clipboard of your computer.
3. Paste (Ctrl+V) the text anywhere on your computer

7. Device Screenshot

This feature enables you to take capture the device screen at any point during testing. You can use this to highlight an issue and share it with your colleagues

8. Burst Mode Screenshot

Burst Mode is an advanced mode of taking screenshots. On clicking on this button, for a period of 30 seconds, a screenshot is automatically taken everytime there is a change on the device screen. At the end of this period, you can individually edit and download each screenshot.

This feature proves very helpful in cases where you would like to capture an entire flow in your testing scenario.

9. Device Log

You can view the logcat report in a tabular format with options to filter by the log level. You also have the option to download the log in CSV format.

10. ANR Log

The ANR log or the 'Application Not Responding' report is generated in the event that your app crashes. This log gives you significant information in determining the casue of the crash and is also available for download.

11. Share Test Session

If you encounter a situation where you find a specific scenario in your app flows that you would like to cross-check with or show to your colleagues but they aren't nearby, don't worry, we got your back.

The 'Share Test Session' feature enables you to collaborate with colleagues by sharing your device screen with them.

You can share your device screen in two ways:-

- a. by sending them a link to your device link by clicking on the 'Copy Share-Link to Clipboard' button
- b. by scanning the QR code displayed with a mobile phone

Once your colleagues click on the link you shared or finish scanning the QR code, not only can they see you perform various actions on the device screen, they can also interact with the same device screen through their computer or mobile device.

12. Connect to ADB remotely

This feature was developed with the intention to help your dev team.

Let's say you find an issue while testing and then reach out to a developer for support.

Now, if this developer would like to access the ADB on the same device for further investigation, she or he can do so remotely by running the command displayed on your screen.

They can now work on the device as if it is connected directly to their own computer.

This feature is also of use to developers who may want to test their code while developing on Android Studio

13. Switch to multiplexing

You can toggle between multi-device testing and testing in a single device using this button.

14. Log bug

Use this feature to directly logs bugs that you encounter while testing into your bug tracking system, without moving away from your test session.

RobusTest supports API integration with a host of third-party bug tracking software such as JIRA, Bugzilla, etc.

Once integration with the bug tracking software is enabled from the Admin Console, you can start logging bugs.

While logging the bug, you can choose the Assignee, the reporter, the type of issue, a summary of the issue and a detailed description. You also have the option to attach the device logs, the ANR log and screenshots as well.

At the time of logging of the bug, in addition to the above details, RobusTest will add more information to the ticket pertaining to the app, app version, OS version, device details, project details, etc

15. Change Wifi

Sometimes you may want your test device to connect to a different Wifi network. In such cases, you can use this feature to select the Wifi network of your choice by providing the SSID and Password.

16. Install Build

This option enables you to select and install a build of your choice from the options provided in the drop down. Only builds previously uploaded to your project will be available for selection

17. Network Shaping

Network Shaping enables you to select a specific kind of network to test your app on. E.g. 2G, 3G, 4G, etc. You are enabled to create an ATC Network profile which simulates characteristics of the kind of network you choose. You can know more about creating ATC profiles in the Admin Console section.

5.3 Multi-device testing

The ability to perform manual testing on multiple devices in parallel is one of the defining features of the RobusTest platform.

We refer to this ability as **Multiplexing**

Advantages of Multiplexing

Multiplexing provides you an advantage in the following scenarios:

1. You can perform manual testing on multiple devices of different screen sizes and running different OS versions and view all device screens simultaneously.
2. It is useful in scenarios where testing involves interaction between two devices.
 - E.g. 1: Making a phone call from one device to another
 - E.g. 2: Sending an SMS from one device to another
 - E.g. 3: Testing the chat feature in an app using two or more devices

3. The 'Replication' feature (described below) enables you to execute the same action on multiple devices by performing the action on one device

How to start a Multiplexing session

There are two ways to begin a Manual multiplexing test session:

Method 1: *Starting a multi-device test session directly*

1. Click on the 'Manual' icon on the Project Dashboard.
2. Select the devices you wish to test on by clicking on the 'Select' button under those devices.
3. Once you have selected the devices, click on the 'Play' button on the top right corner. The multiplexing session is now in progress.

Method 2: *Switching to a multi-device test session from a single device manual test session*

1. Click on the 'Manual' icon on the Project Dashboard.
2. Select one device you wish to test on by clicking on the 'Select' button under those devices.
3. Once you have selected the device, click on the 'Play' button on the top right corner. The single device manual test session is now in progress.
4. Click on the 'Switch to Multiplexing' button on the horizontal menu. You are now in a multiplexing test session and can add more devices to this session for testing.

Understanding the Multiplexing test session

Now that you have started a multiplexing session, let's see what operations and features are available to you to aid you in your testing.

There are two menus you need to familiarize yourself with in a multiplexing test session:

1. The Vertical menu along the left border of the multiplexing test session
2. The Horizontal menu above each device screen

1. Vertical menu on the left of the test session

The following buttons are visible on this menu:

1. Enable Replication - Clicking on this button enables you to replicate an action that you perform on one device in the multiplexing test session on all other devices in the same test session.

This feature works in cases where all devices being used in the multiplexing test session are of comparable screen size and resolution.

It helps you to considerably reduce the time spent in testing on multiple devices.

2. Add more devices - You can add more devices to a multiplexing session by clicking on this button.

Clicking on this button brings up the device selection dialog. Click on the 'Select' button for the device that you wish to add to the multiplexing test session and then click on the 'Play' button. The new device screens will now be visible in the multiplexing test session.

3. Take Screenshot - On clicking on this button, a screenshot of all device screens in the test session is taken and downloaded to your computer.

4. End Test Session - Clicking on this button ends the multiplexing test session in its entirety and all devices that are used in the test session are freed and made available for new test sessions.

2. Horizontal menu above each device screen

This menu is visible above each device screen in the test session and is specific to that device. The following buttons are visible on this menu:

1. Switch to Manual mode - Clicking on this button, opens that specific device in a single-device manual test session. You can always go back to your multiplexing test session by clicking on the ‘*Switch to Mutlplexing*’ button in the manual test session.
2. Download device logcat - As the name suggests, clicking on this button downloads the logcat report for that device.
3. View device logcat - Clicking on this button displays the logcat for that device on a new browser tab.
4. End Test Session - You can remove a specific device from a multiplexing test session by clicking on ‘End Test Session’ button on the horizontal menu bar on that device.

5.4 Test Embedded Video

Now that you have created a project and selected a build to test, you may want to get right into it and perform some manual testing on your app.

To start a Manual test session:

1. Click on the ‘Manual’ icon on the Project Dashboard

A device selection screen now pops up. You may search for a device on the ‘Device selection screen’ based on device name, platform version, screen size, hardware configuration (e.g. Memory and CPU), node name, node IP, etc.

2. Select the device you wish to test on by clicking on the ‘Select’ button under the device

Note: RobusTest provides you a way to select multiple devices in parallel to perform your manual testing. To know more go to the page on [Multi-device testing](#)

3. Once you have selected the device, click on the ‘Play’ button on the top right corner.

The device screen now comes up and you can see that your app is installed on the device.

Congratulations! You have succesfully begun a Manual test session

Now let’s see how RobusTest helps you to test better

The Manual Test Session, consists of 2 parts:

1. Device screen
2. Test Configuration section

1. Device screen

The device screen is where you perform your testing. You can perform various gestures like tap, swipe, scroll and entering text with the help of mouse/trackpad and keyboard. The buttons at the bottom of the screen are available for use as Android navigation buttons.

2. Test Configuration section

The ‘Test Configuration’ section enables you to test better and easier by providing various add-on features

These features are available across 2 menus:

- a. *Device Configuration Menu*
- b. *Session Configuration Menu*

Once your testing is complete, you can click on ‘End Session’ button to exit the Manual test session.

6.1 Recording user actions

When you hover your mouse pointer over the app that you are testing, you will notice a rectangular placeholder overlaid on top of various objects on the screen. You can also find the name of the highlighted object on the header.

When a particular object is highlighted using the placeholder, it is selected and you can record various actions related to that object.

Left-clicking on a highlighted element opens up a context menu which lets you do the following:-

1. Record user actions

Depending on the element highlighted, you can select one of the following actions to be recorded:

Type - this records a 'Type' action on the object and is used to enter text into the selected element

RobusTest provides you a number of options for typing-text

Tap - this records a 'Tap' action on the object that you have selected

Double Tap. - this records a 'Double Tap' action on the object that you have selected

Press and Hold - this records a 'Press and Hold' or a 'Long Press' action on the object that you have selected

Zoom. - this records a 'Zoom' action on the object that you have selected

Pinch. - this records a 'Pinch' action on the object that you have selected

Swipe - this records a 'Swipe' action on the object that you have selected. For more details, go to the swipe-on-page section

Swipe Till. - this records a 'Swipe Till' action on the object that you have selected

Precise Tap - this records a 'Precise Tap' action on the object that you have selected

Recording any of the above actions results in a corresponding test step being created

2. Verify element

This option allows user to create a check or an assertion on an object. If the verification condition fails, the test step fails.

You can know more on the [Verification](#) page.

3. Store element

Sometimes you may want to store values corresponding to an element's attributes for later verification. These include attributes such as *'text'*, *'content-desc'*, *'checkable'*, *'checked'*, *'enabled'*, *'clickable'*, *'focused'*, *'selected'*, etc.

You can use the 'Store Element' option to do so.

- a. Left-click on the element whose information you need to store and click on 'Store Element' on the Context Menu.
- b. You can see that a 'Store Element' test step has been recorded
- c. Expand this test step and go to the 'Return Data' tab. Here you can see all attribute information corresponding to the element selected. This information can be used for later verification (as explained in the point above).

4. Find concurrent element

This option enables the user to identify and select the right element on which an action is to be performed.

The elements on which an action is to be performed by the user are identified from the pagesource that is fetched for that app page.

However, sometimes, it is found that an element is overlaid by one or more other elements in the pagesource. This results in the user not being able to highlight the correct element to record a user action.

This is where the 'Find Concurrent Element' option comes into the picture and enables the user to identify and select the right element. It does so as follows:

- a. Hover the mouse pointer over the area on the device screen where the element on which you wish to perform an action is present
- b. Left-click and select the 'Find Concurrent Element' option.

A 'Concurrent Elements' pop-up window now opens up. The 'Concurrent Elements' window lists all elements that are present on the area of the device screen on which you executed a left-click.
- c. On moving the mouse pointer over each element listed on the 'Concurrent Elements' window, the corresponding element is highlighted on the device screen.
- d. On the 'Concurrent Elements' window, click on the name of the element on which a user action is to be recorded and then click on the 'Save' button. This element is now selected and is seen highlighted on the device screen.
- e. Any user action that is now recorded by left-clicking on the highlighted element is executed on the highlighted element.

5. Find parent element

This option allows the user to identify the parent element of the highlighted element.

6. Inspect element

This option allows the user to study the object and its attributes. This is useful for advanced users and for debugging.

6.2 Test Step Management

For each action you record on the app, a corresponding test step is created in the 'test step table' to the right of the device screen

<input type="checkbox"/>	#	Test Step							
=	<input type="checkbox"/>	1		Tap on text "LOGIN"					
=	<input type="checkbox"/>	2		Tap on text field "text_input_email_login"					
=	<input type="checkbox"/>	3		Type in text field "user@robustest.com"					

As you can see, RobusTest provides a default test step name for each test step by attempting to identify the user action. You can customise the name as required (more on this later).

Let's have a look at the different operations available to the user to perform on a test step

The icons to the right side of the test step relate to actions you can perform on that specific test step (see screenshot below)



- Copy step - this enables you to create a duplicate test step performing the same action as the original test step. This saves you time taken in re-recording the same test step, when you have to repeatedly perform the same action in the test case
- Update Screenshot - when you first record a test step, RobusTest captures a screenshot of the app page as it was at that moment. If you feel that this screenshot does not adequately represent the test step that was recorded (say, because the page took too long to load), then you can use the 'Update Screenshot' button to capture a new screenshot.

The screenshot that is thus captured is displayed as the 'Original' screenshot in your run report when you execute the test case as part of a test run.

- Play step - this enables you to execute the test step while in the Automation test session
- Delete step - this enables you to delete the test step that you have recorded
- Show step details - clicking on this expands the test step to reveal more options. You can know more by clicking on test-step-details

The icons to the left side of the test step relate to actions you can perform on one or more test steps (see screenshot below)

<input type="checkbox"/>	#	Test Step					
=	<input type="checkbox"/>	1	Tap on text "LOGIN"				
=	<input type="checkbox"/>	2	Tap on text field "text_input_email_login"				
=	<input type="checkbox"/>	3	Type in text field "user@robustest.com"				

- Re-arrange icon

This icon helps in two ways:

- By pressing and holding the mouse pointer over this icon and then dragging it, the user can re-arrange the position of this specific test step within the test case.
- By clicking on this icon, the user can pin the test step. This means that any new test step that is now recorded will be positioned after the pinned test step.

<input type="checkbox"/>	#	Test Step					
=	<input type="checkbox"/>	1	Tap on text "LOGIN"				
	<input type="checkbox"/>	2	Tap on text field "text_input_email_login"				
=	<input type="checkbox"/>	3	Type in text field "user@robustest.com"				

- Checkbox to select test step - when the checkbox on a test step is selected, two more buttons become at the top of the Test Step table - 'Group Play' button and the 'Group Delete' button
 - Group Play button - when this button is clicked, all test steps in the test case, that have been selected by enabling the checkbox on the test step, are executed in the order in which they are present in the test case.
 - Group Delete button - when this button is clicked, all test steps in the test case, that have been selected by enabling the checkbox on the test step, are deleted in the order in which they are present in the test case.

<input type="checkbox"/>	#	Test Step					
=	<input type="checkbox"/>	1	Tap on text "LOGIN"				
=	<input checked="" type="checkbox"/>	2	Tap on text field "text_input_email_login"				
=	<input checked="" type="checkbox"/>	3	Type in text field "user@robustest.com"				

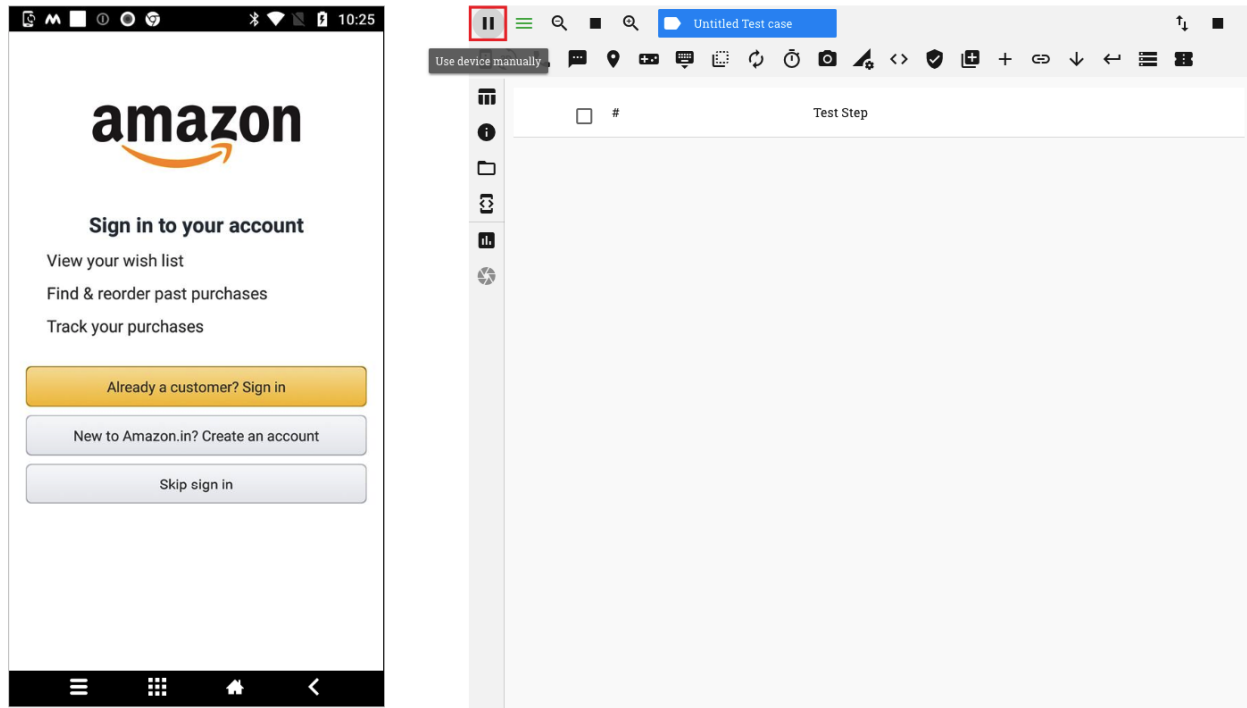
- Test step execution status - this icon indicates the status after execution of the test step.

6.3 Test Session - Device Management

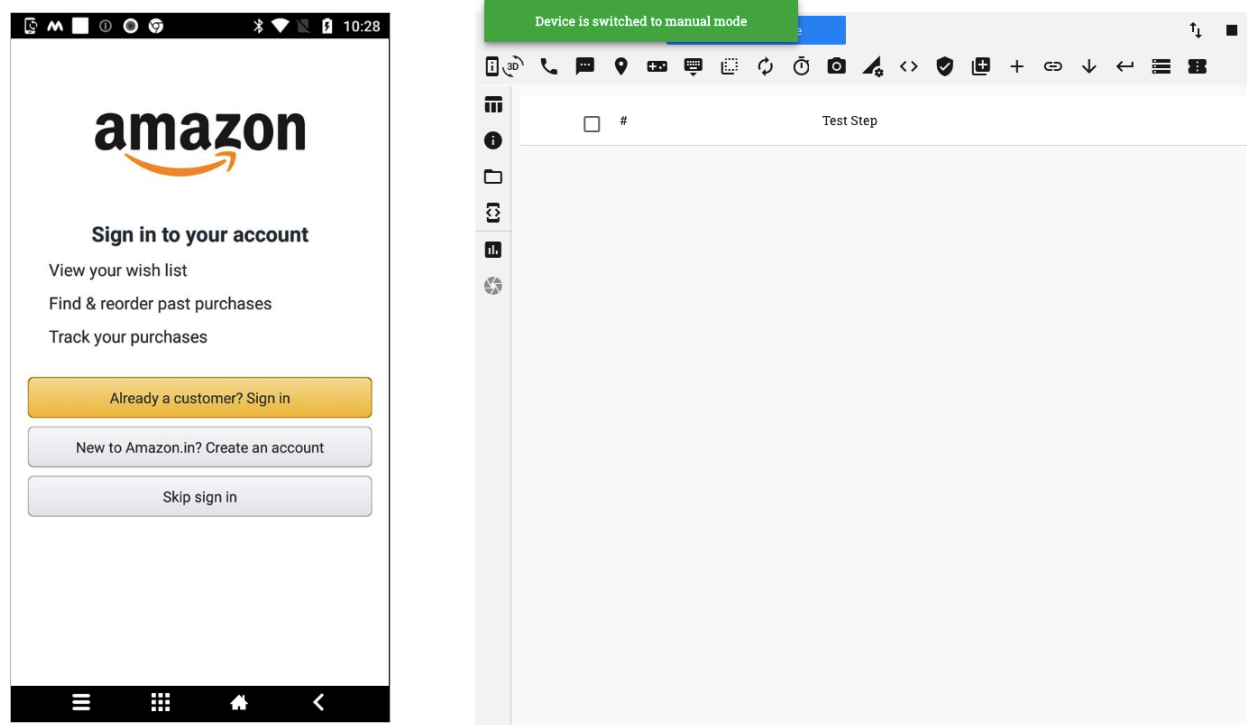
1. Pause/Resume Automation session

While in an automation test session, you may, at times, choose/need to access the device in Manual mode.

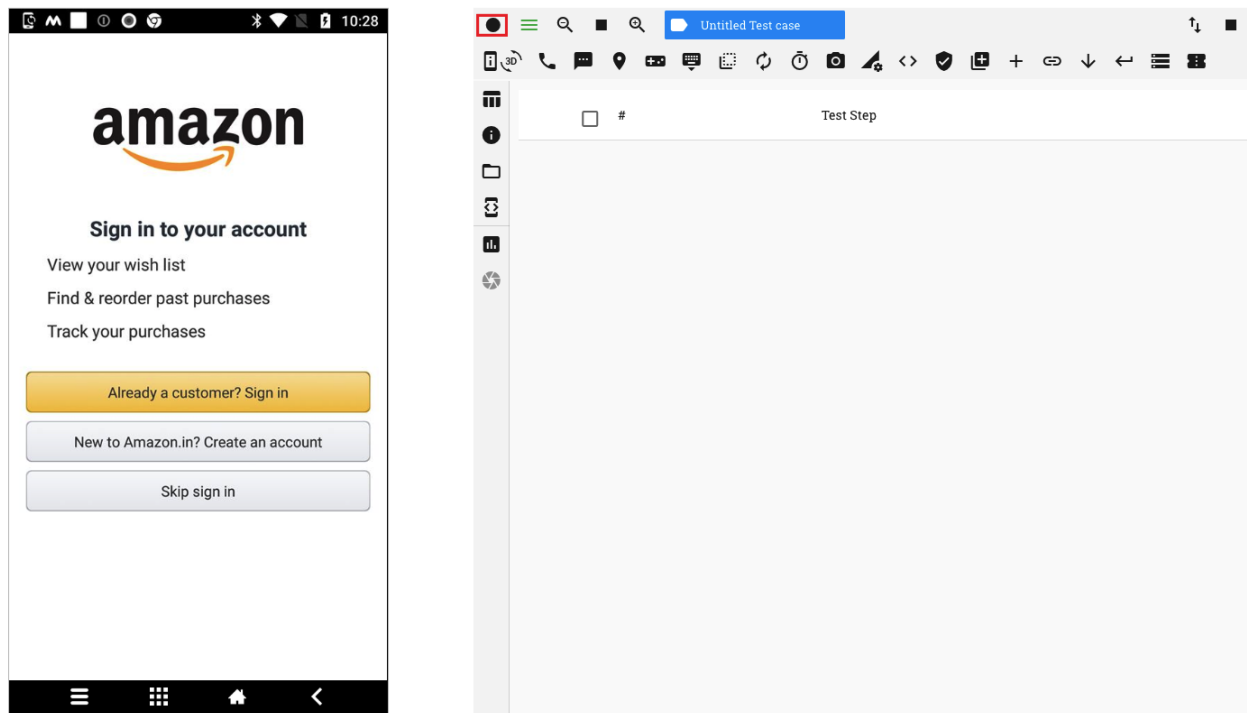
You may toggle between Automation and Manual test modes by clicking on the 'Pause/Resume' button



On clicking the 'Pause' button, the device is switched to Manual mode and a corresponding message is displayed. You can now access the device screen directly as if you were accessing it manually

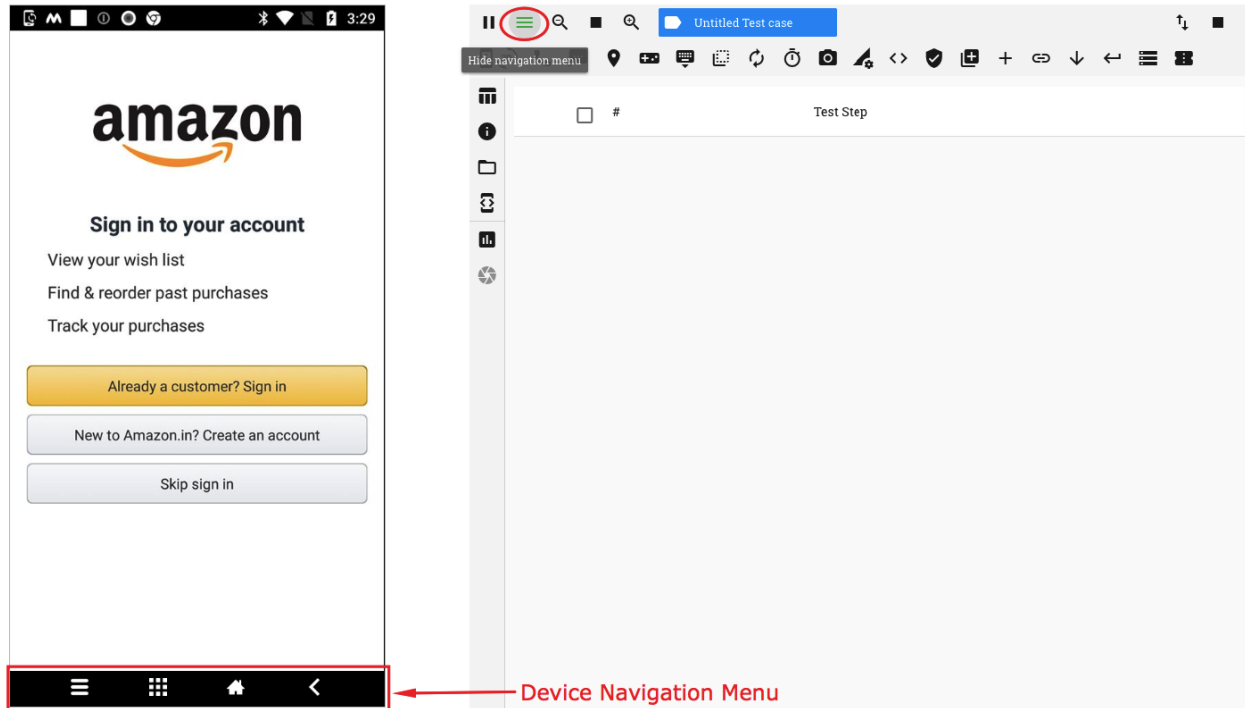


You can see that the icon for the button has now changed. On clicking on the 'Resume' button, the device is switched to automation mode

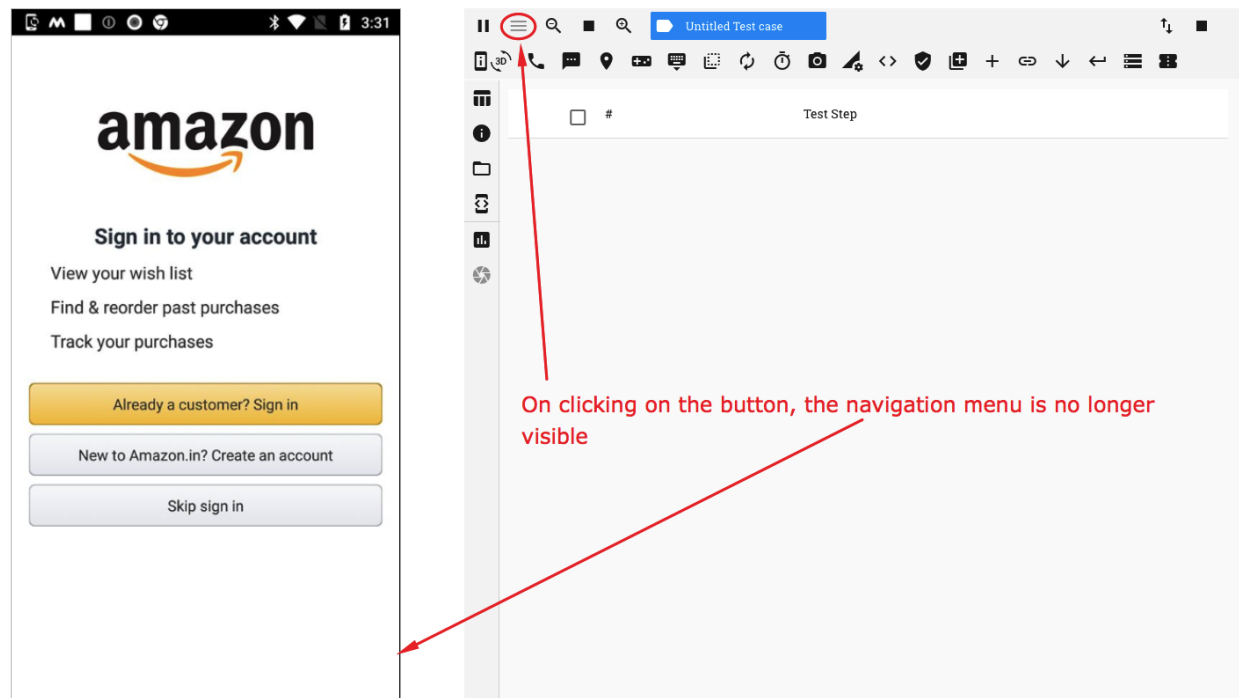


2. Show/Hide Navigation Menu

RobusTest provides you a device navigation menu on the device screen when in Automation mode



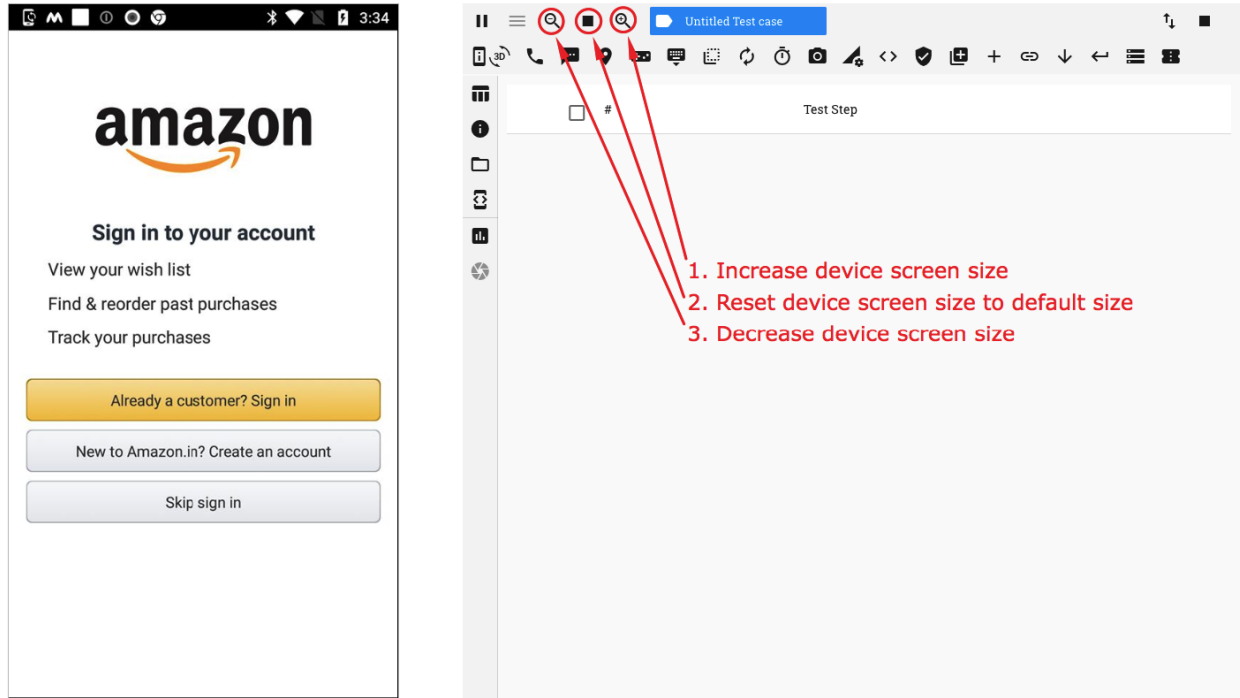
On clicking the 'Hide Navigation Menu' button, the menu is no longer displayed on the device screen. Clicking on this button again, enables the menu.



On starting an automation test session, by default, the navigation menu is enabled

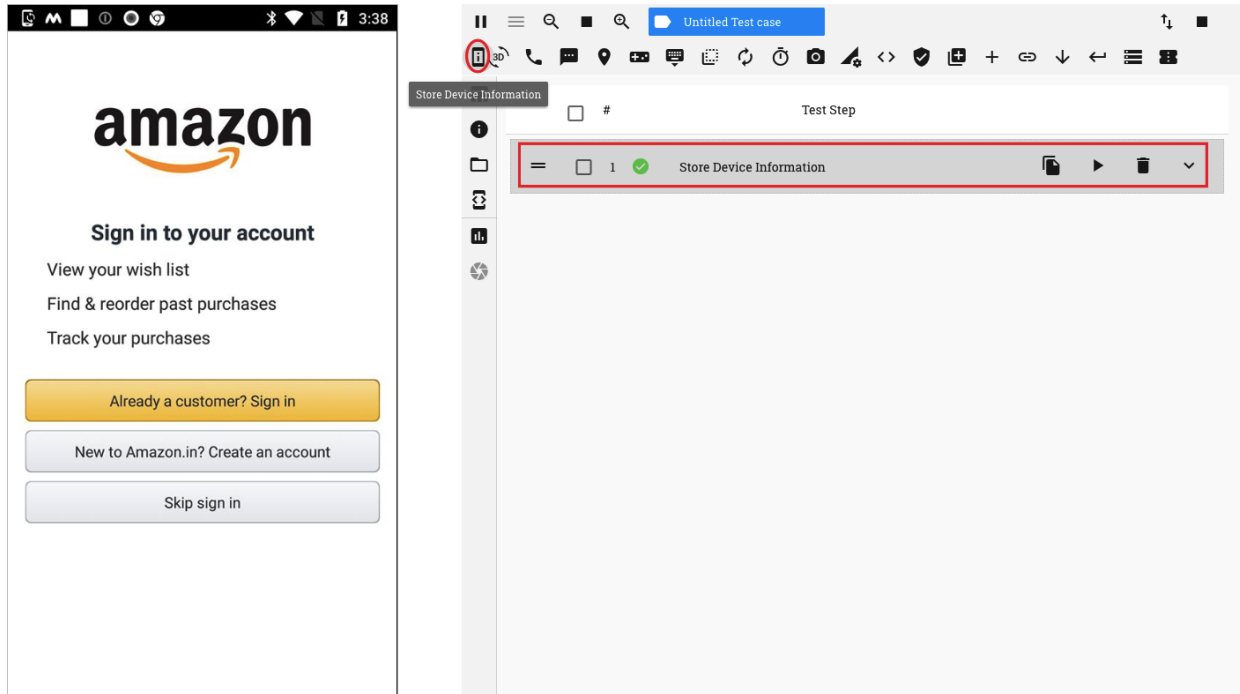
3. Increase/Decrease device screen size

You can manipulate the size of the device screen by using the resizing buttons on the horizontal menu, as shown in the screenshot below

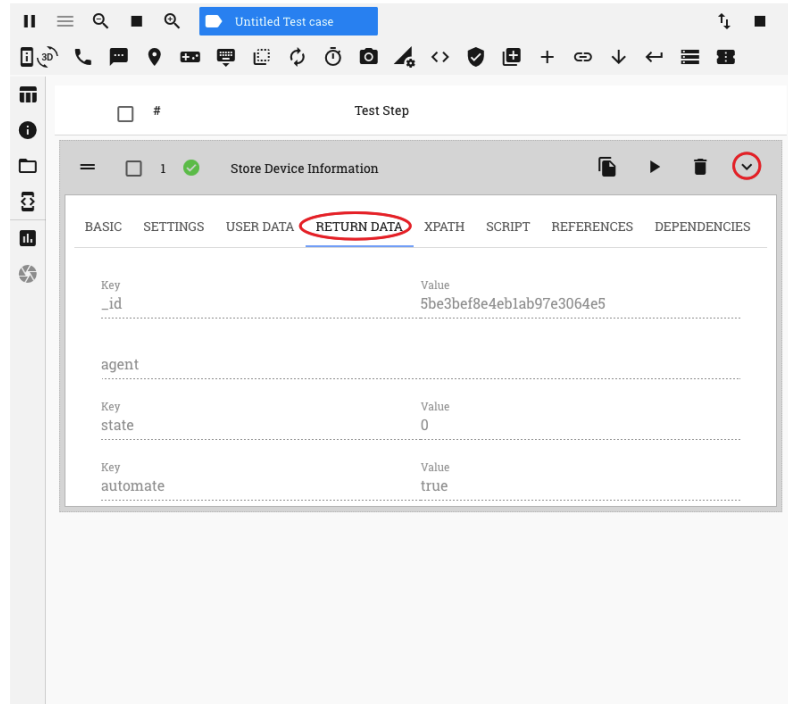
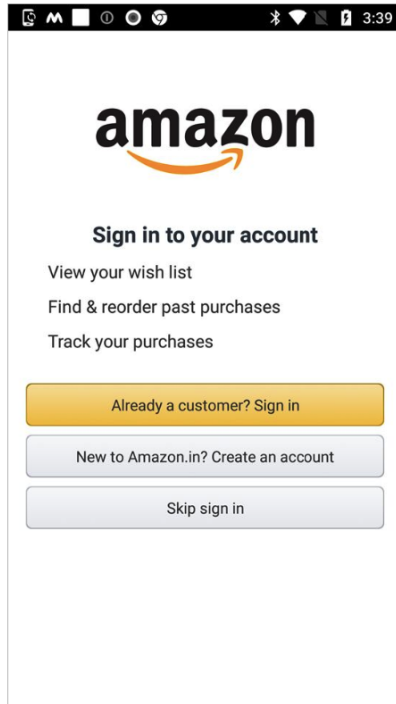


4. Capture device information

Information about the device can be captured by clicking on the 'Store device information' button. On clicking this button, a test step is seen to be recorded



Now, expand the test step and click on the 'Return Data' tab. All information about the device can be viewed on this tab. You can scroll down to see more information

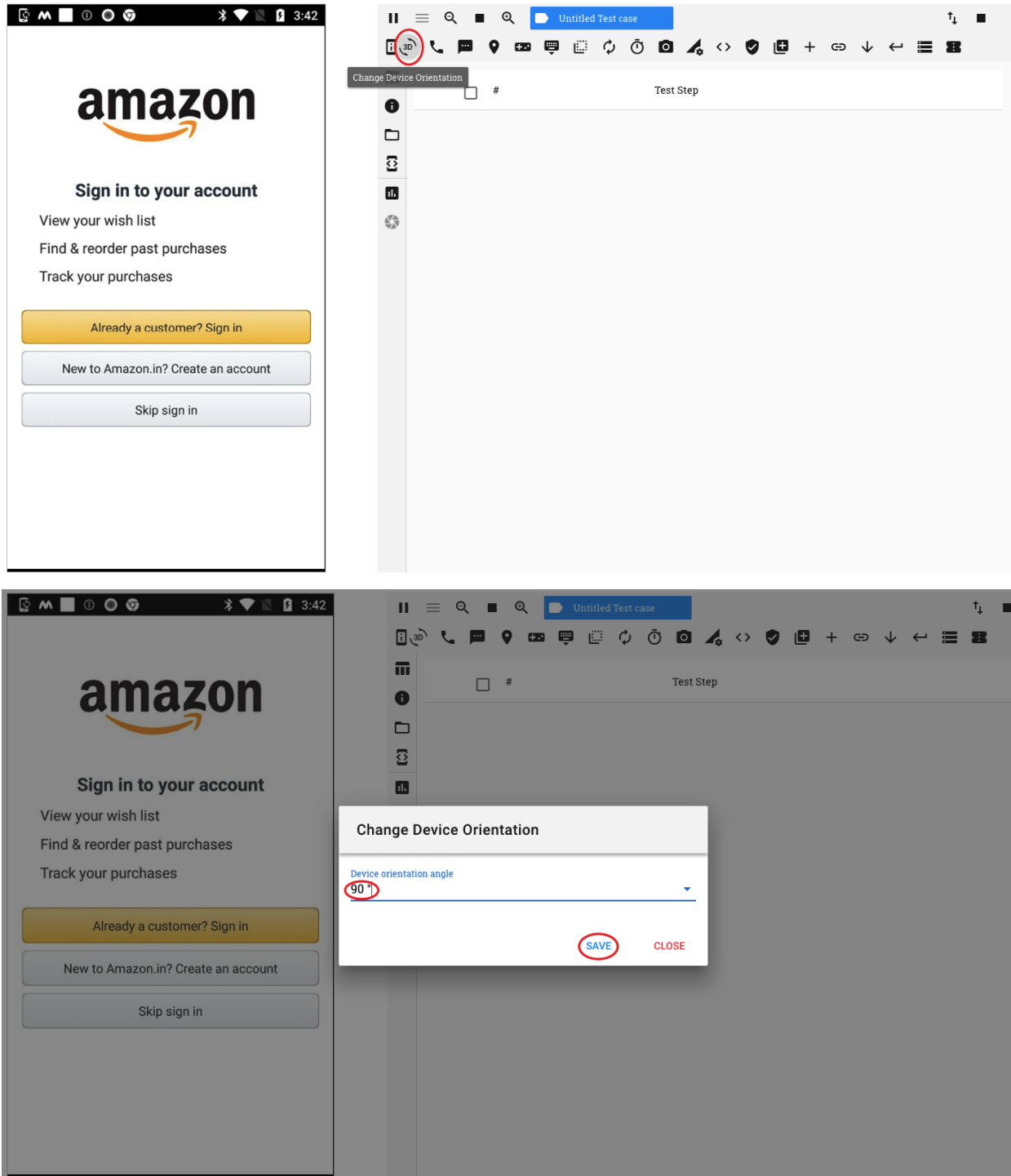


5. Change Device Orientation

You can toggle between the Landscape and Portrait modes by clicking on the 'Change Device Orientation' button

To change to Landscape mode:

1. Click on the 'Change Device Orientation' button
2. Choose 90 degrees on the pop up window
3. Click on the 'Save' button



<Add image of device screen in Landscape mode here>

6. Location Simulation

This feature allows you to test as if the device is present at a different location than where it actually is. This is done by simulating the location on the device.

Pre-requisites:

On the device:

1. in Developer options: * enable 'Mock Locations' * set the Nizedha app as your mock location app
2. in 'Locations', set the 'Location Mode' to 'Device only'

Once the pre-requisites have been met, you can simulate any location as follows:

1. Click on the 'Set Location' button on the header
2. Type the name of the location in the 'Search location here' field and select from the drop down. Alternatively, you can manually pin the location of your choice on the map
3. Once your location has been pinned, click on the 'Set Location' button

Your device will now behave as if it is situated at the location chosen by you.

A corresponding 'Set Location' test step is seen created in the test step table.

You can change the location setting by modifying the latitude and longitude values specified in the 'User Data' section of the recorded test step.

6.4 Test Session - Test Case Management

1. Call a phone number

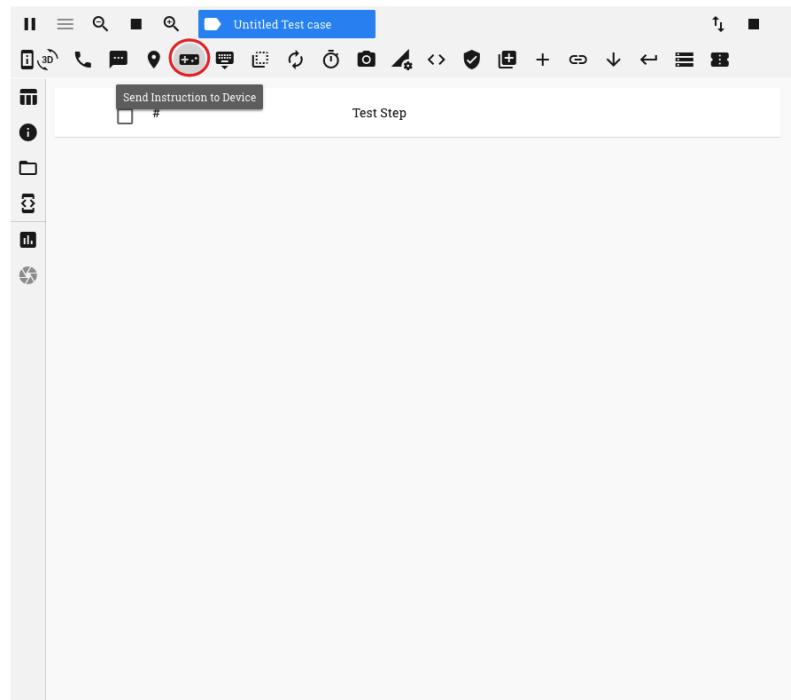
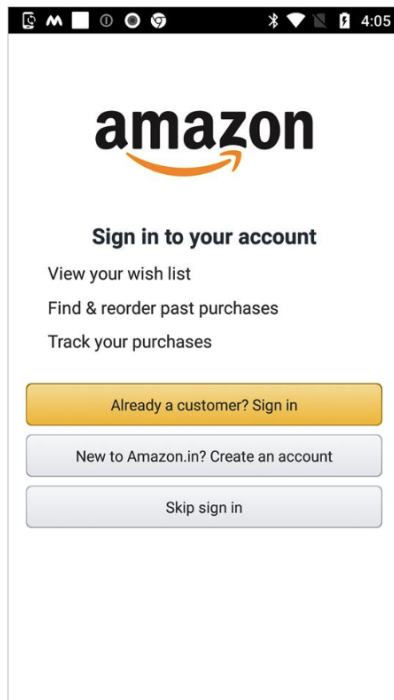
Under Construction

2. Read OTP from phone

Under Construction

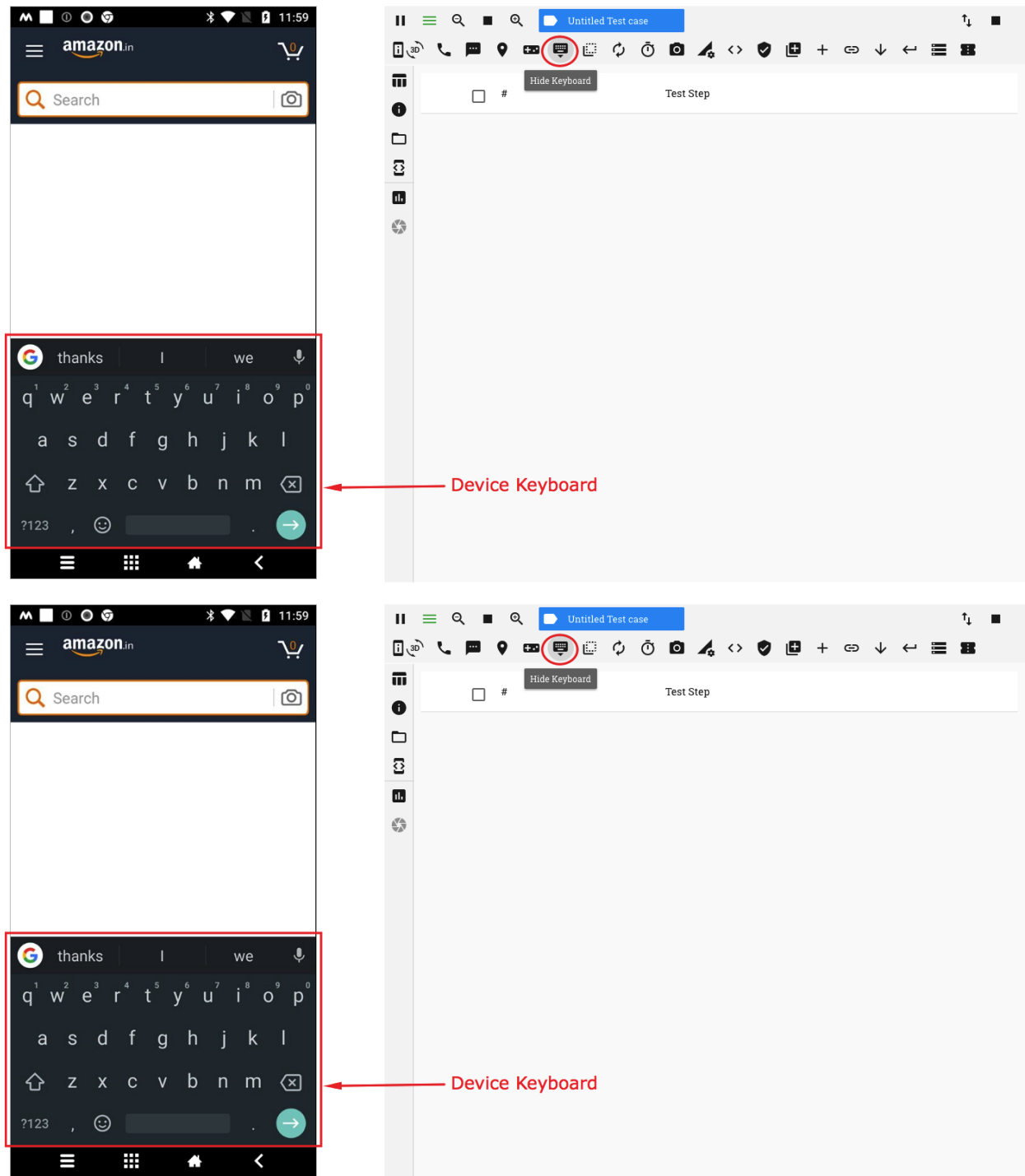
3. Send Instruction to device

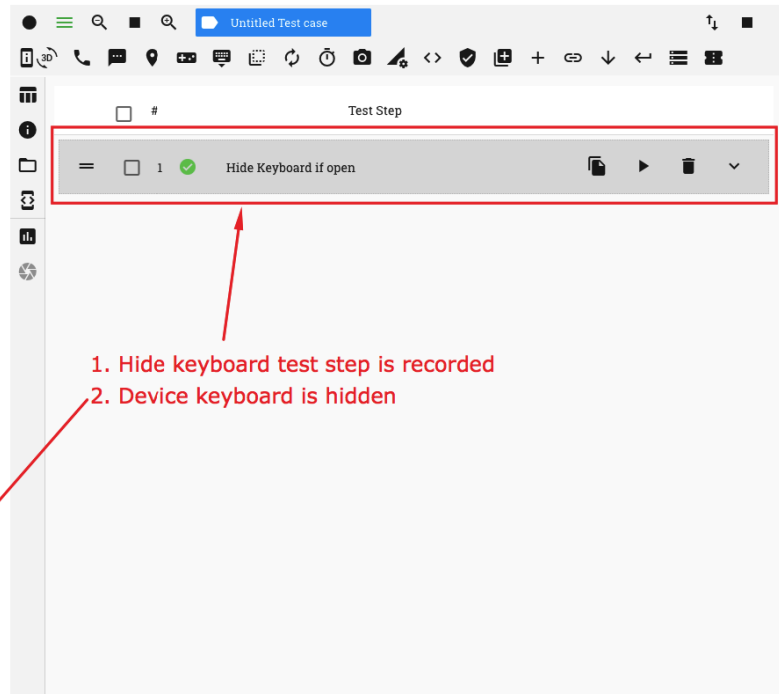
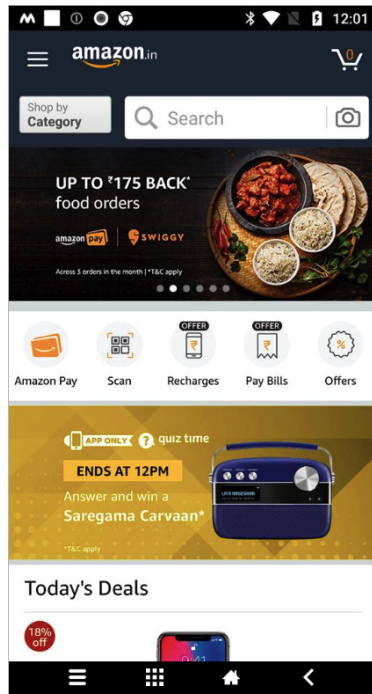
At times, you may want to execute commands through the device keyboard. You can use the 'Send Instruction to device' button to execute actions performed on the device keyboard.



4. Hide keyboard if displayed

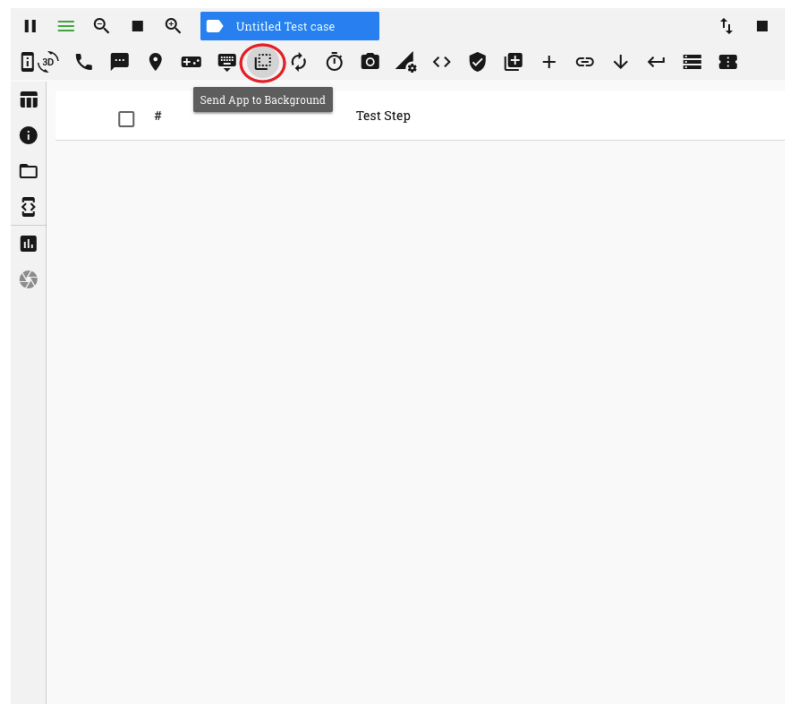
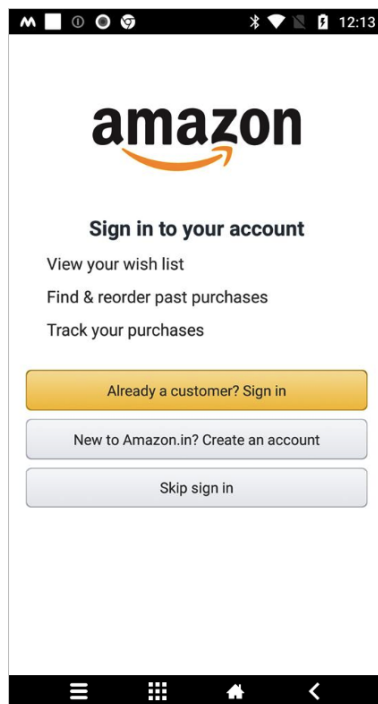
Clicking on the 'Hide Keyboard' button, minimizes the device keyboard that is open on the app



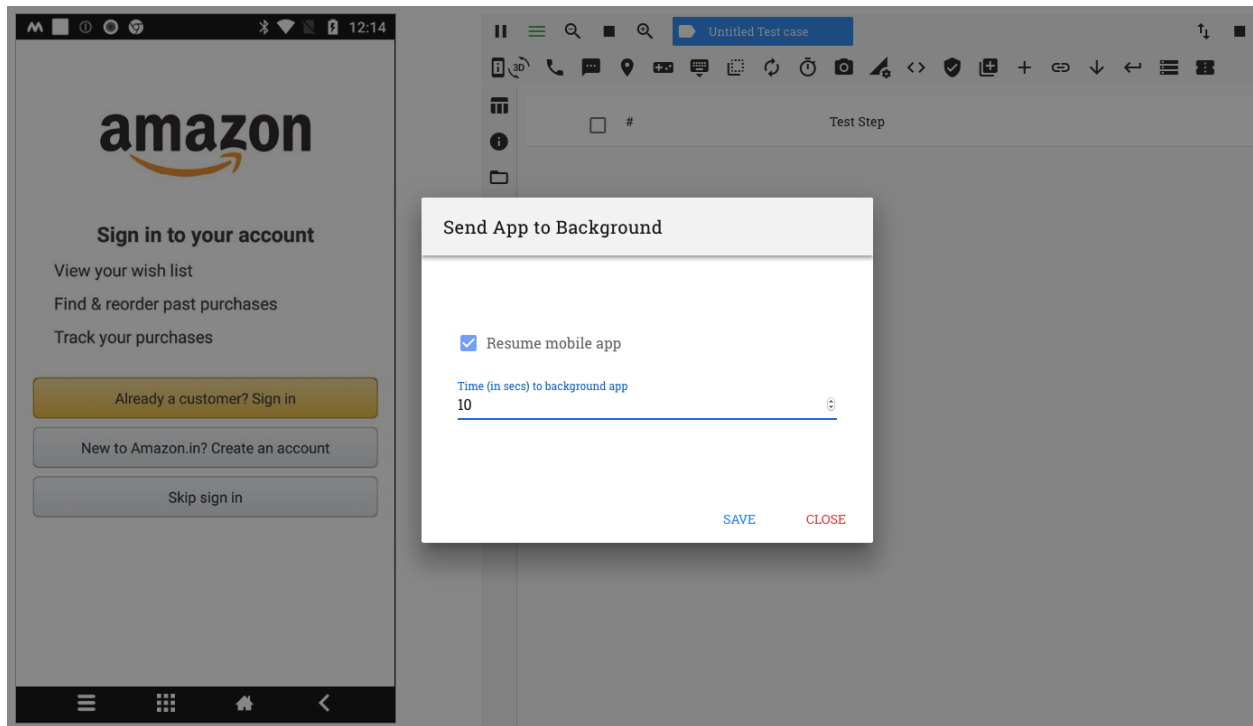


5. Send app to background for 10 secs

The 'Send App to background' button, enables you to send the app into the background for a specified duration.

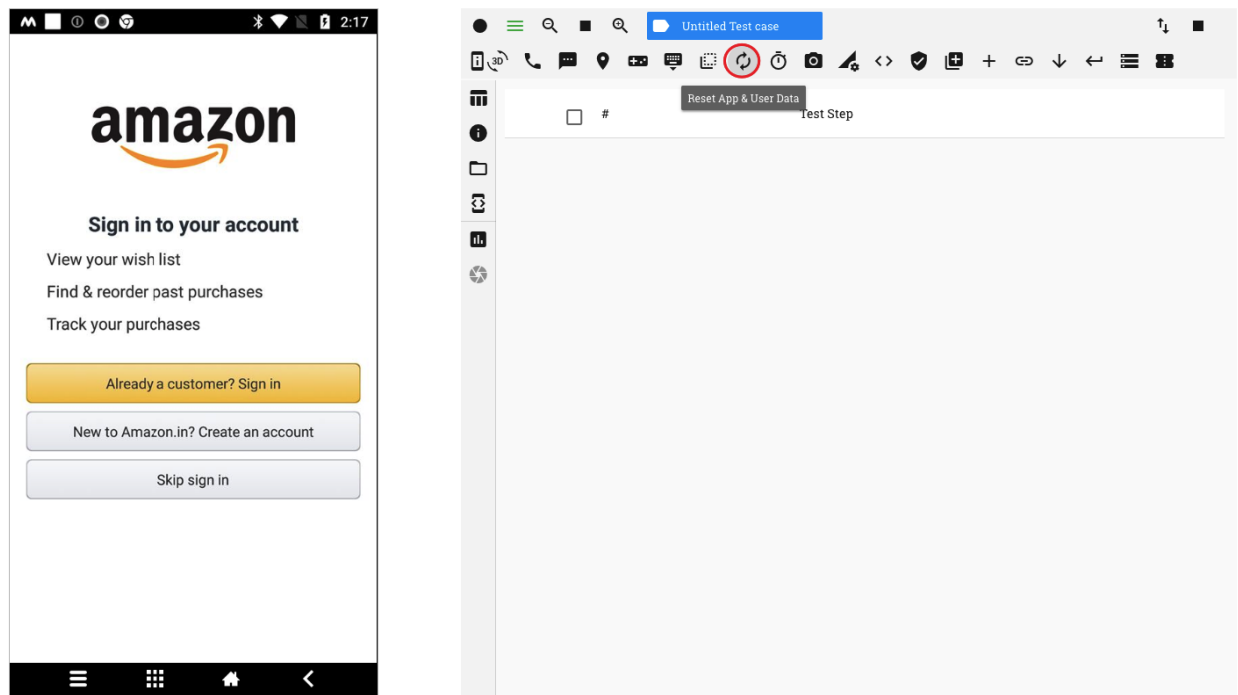


If the 'Resume mobile app' option is selected, the app is brought back to the forefront at the end of the duration



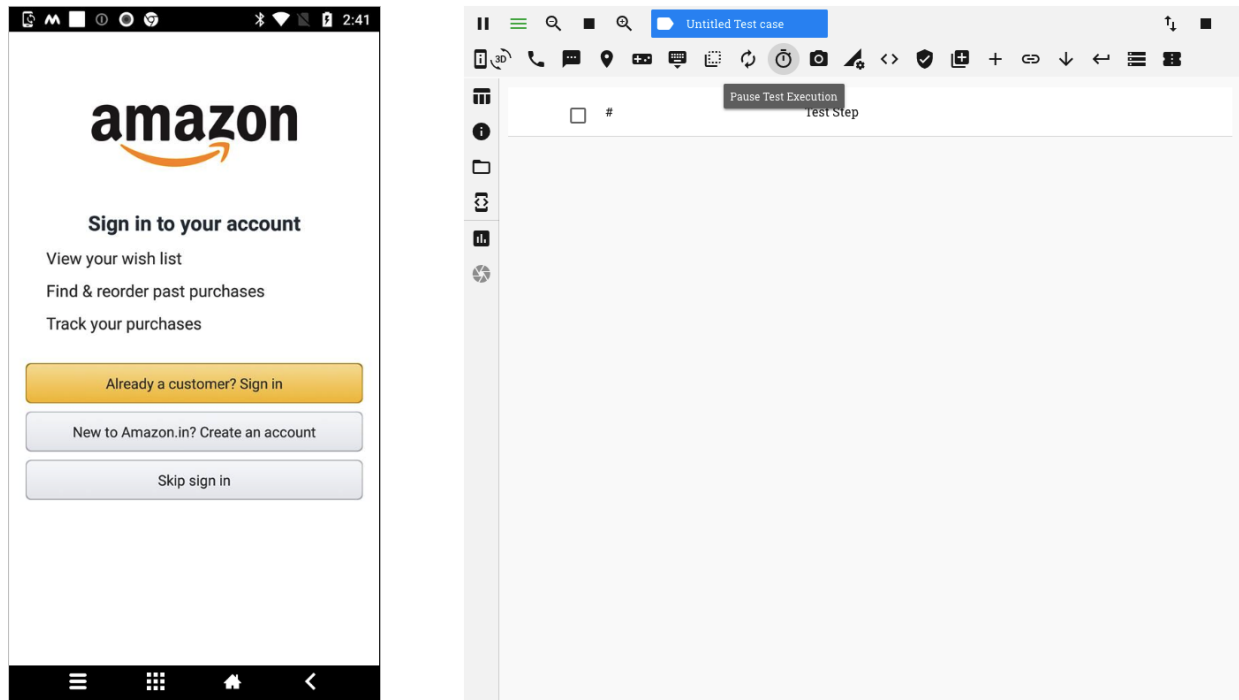
6. Reset app and clear user data

On clicking on the 'Reset app and clear user data' button, all data related to the app that has been cached is cleared and the app is then relaunched. The app then behaves as if it has been launched for the first time

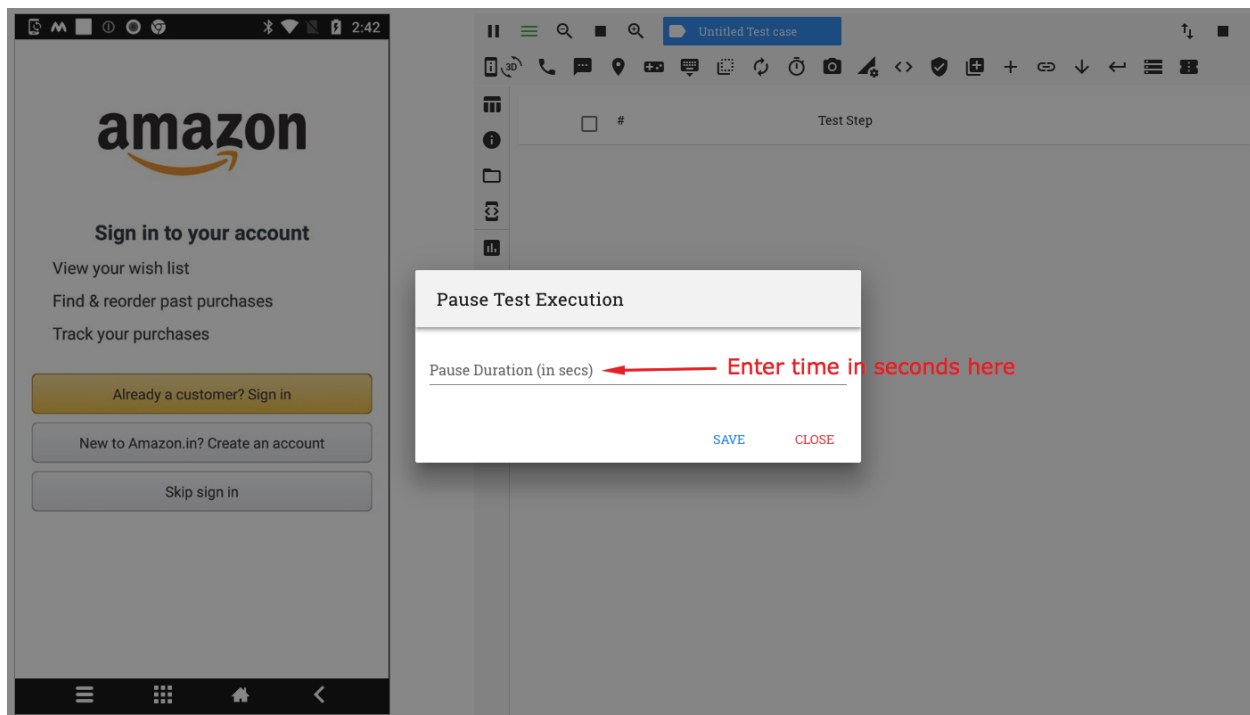


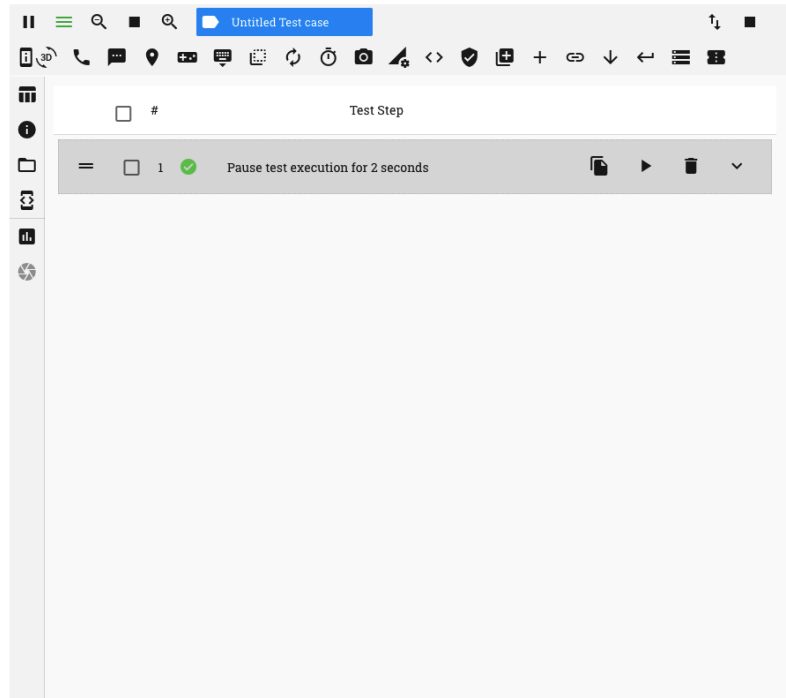
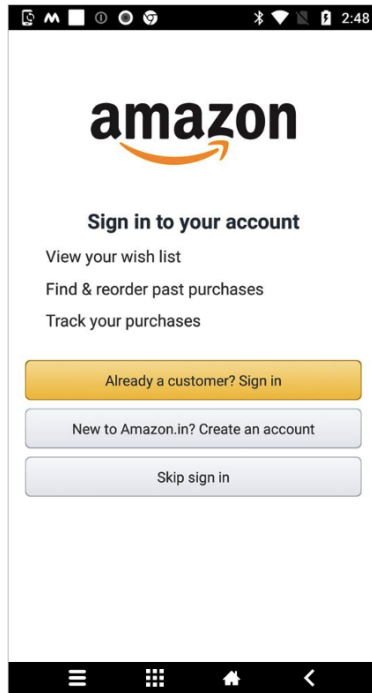
7. Pause Test Execution

You can add a pause time or a wait period after a test step by clicking on the 'Pause test execution' button



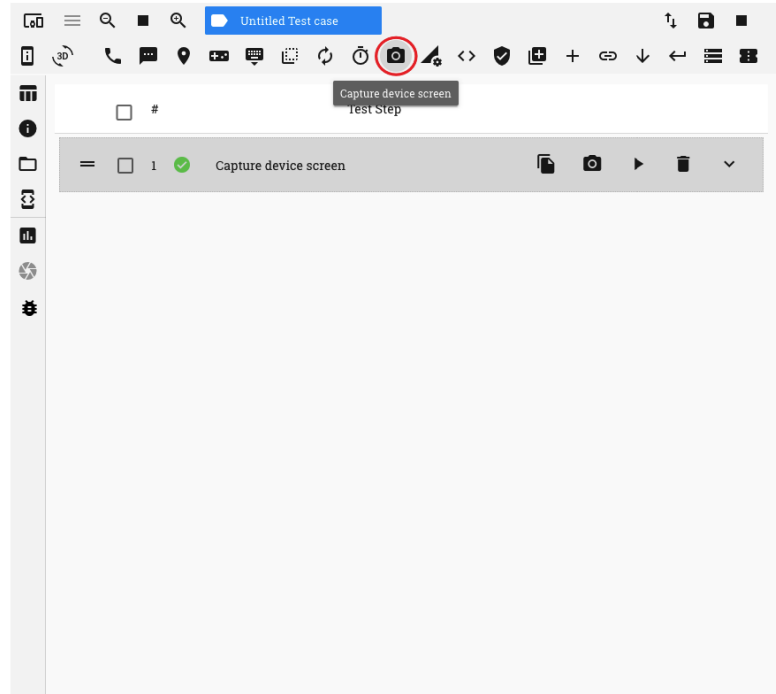
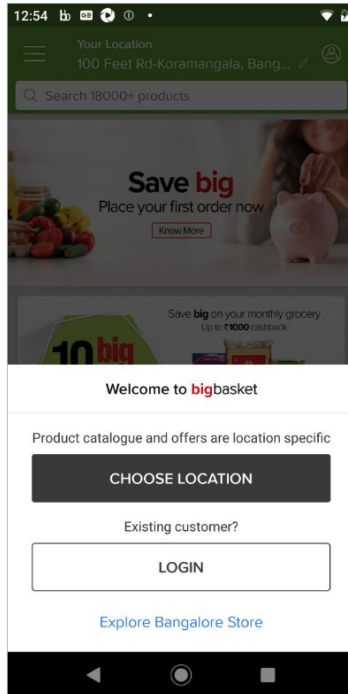
You can provide a duration (in seconds) for which test execution will be paused





8. Capture device screenshot

You can record a test step to capture a screenshot of the device screen at any point during the execution of a test case. This can be done by clicking on the 'Capture Device Screen' button

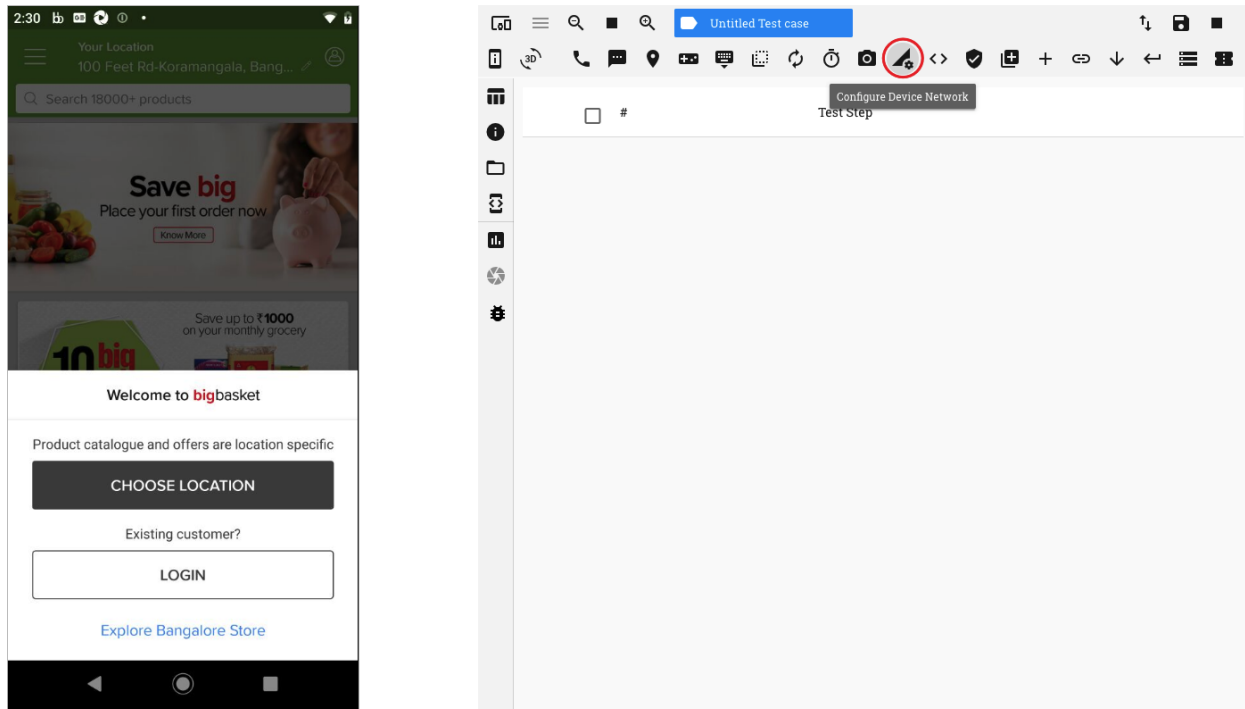


When this test step is executed in a test run, the screenshot captured is available in the functional reports

9. Configure Device Network

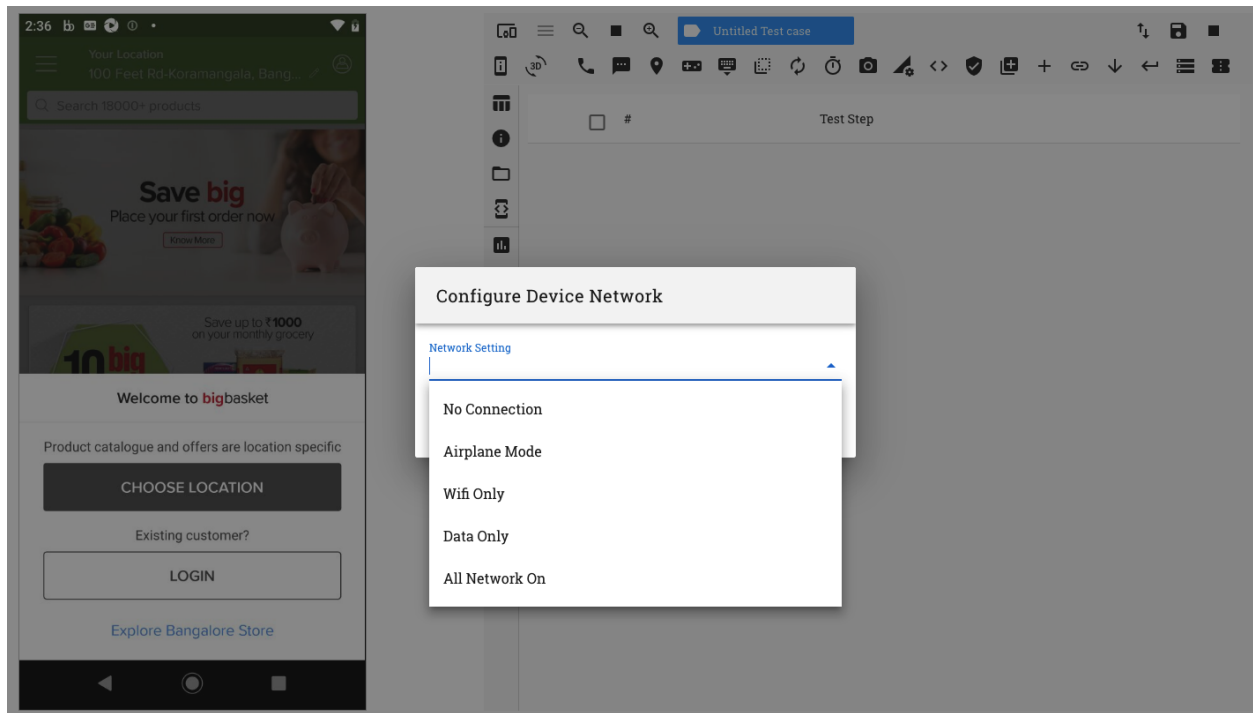
RobusTest provides you an option to choose a network configuration on your device as part of your test case.

To do so, click on the 'Configure Device Network' button



On Clicking on this button, a pop-up window opens from which you may select one of the following options:

1. No Connection - This disables both Wifi and Mobile Data networks on the device
2. Airplane Mode - This enables Airplane mode on the device
3. Wifi Only - This enables Wifi network on the device
4. Data Only - This enables the Mobile Data network on the device
5. All Networks On - This enables both Wifi and Mobile Data networks on the device



Once an option is selected, a test step is created to enable/disable the selected option

10. Execute ADB command

11. Verification

Check out the [Using the 'Verification' button option](#) page to understand this functionality

12. Execute REST API

RobusTest enables you to record test steps to make REST API calls. You can then check the response of these calls for later verification.

1. On clicking on the 'Execute REST API' button, a window pops up

Execute REST API

Basic ^

URL

Request Type ▼

Timeout

Authorization ▼

Header ▼

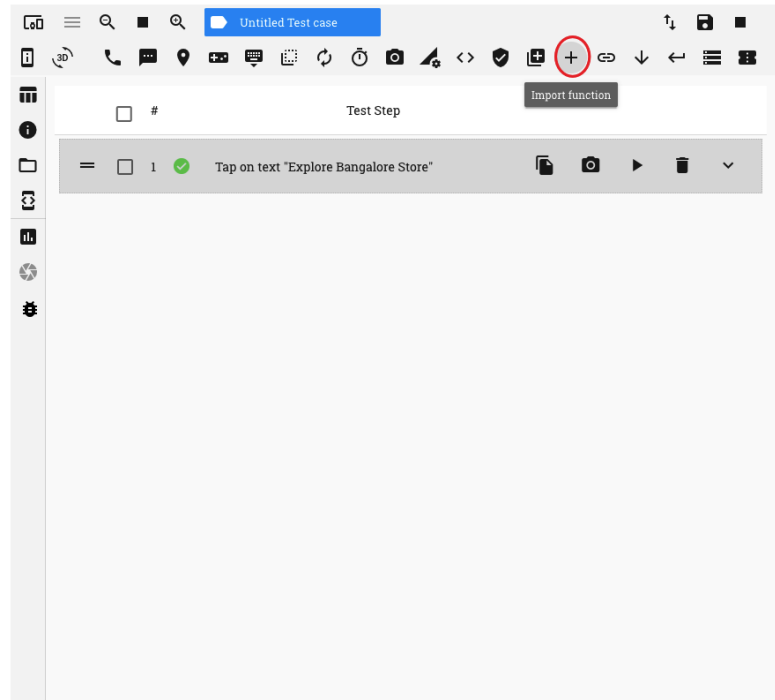
SAVE

CLOSE

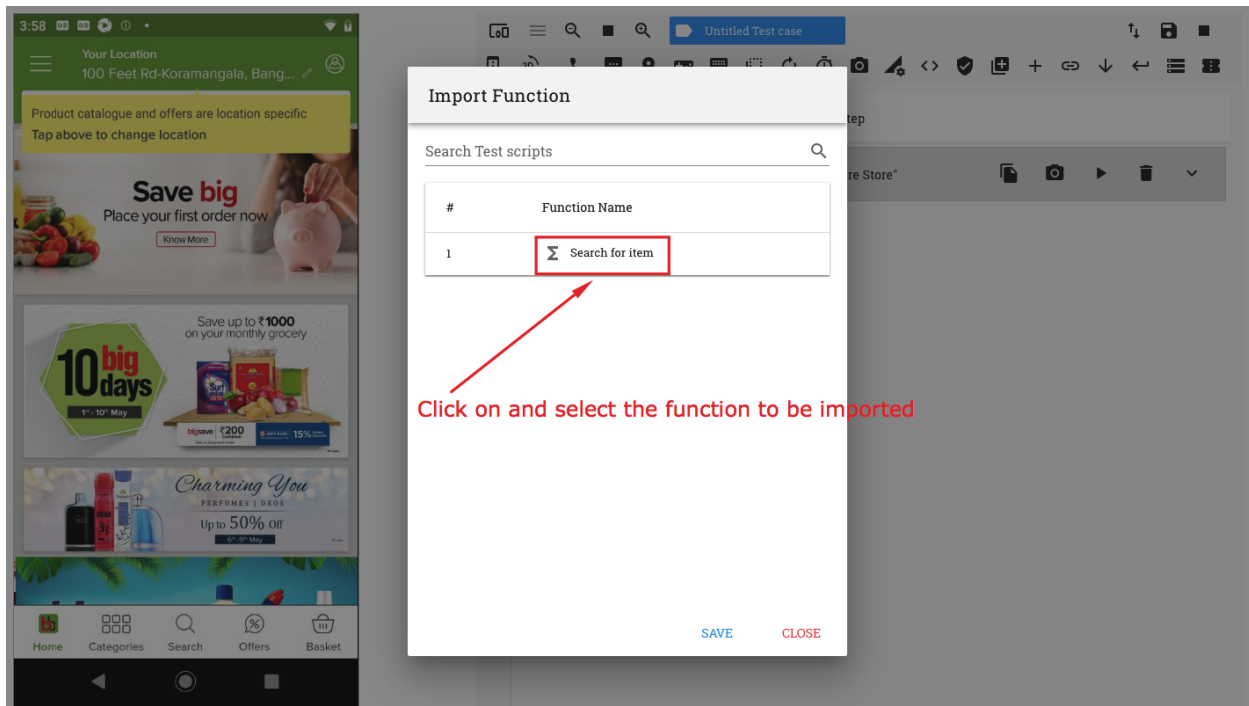
2. On the pop-up window, there are 3 sections available:
 - a. Basic * Provide the REST API and the Request Type (i.e., GET, POST, DELETE, etc) here * If required provide a timeout for the API call
 - b. Authorization * You can specify the type of Authorization to be used for the API and provide appropriate credential values as required. E.g. For Basic Authorization you need to provide the username and password.
 - c. Header * Specify in this section any additional parameters that need to be passed as part of the REST API call
3. Provide details to invoke the REST API and click on the 'SAVE' button.
4. A test step corresponding to the invoking of the API is now seen created.
5. The response of the API call can be seen in the 'Return Data' section of the test step. These values can be used for verification.

13. Import Function

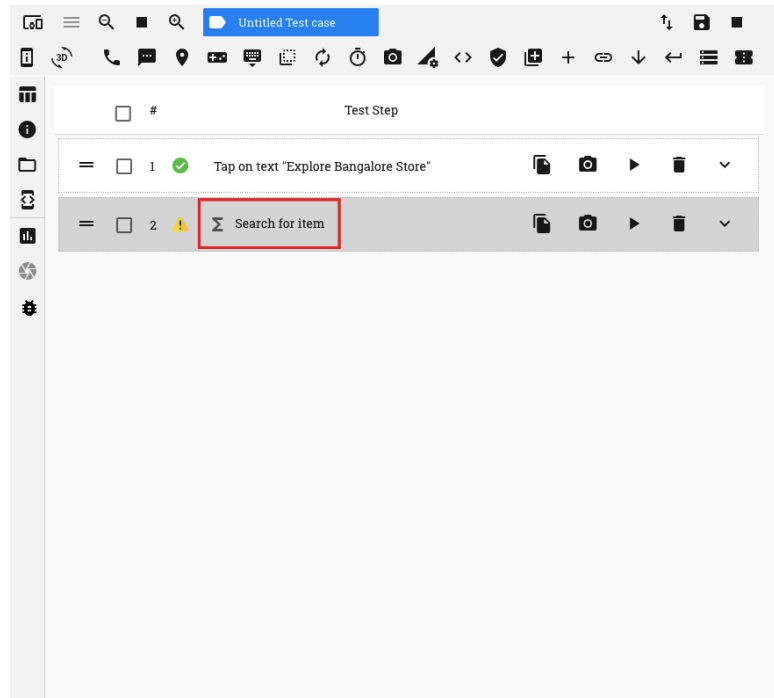
User can import functions created by clicking on the 'Import Function' button



On the window that pops up, a list of functions is displayed. Click on the required function and click on the ‘Save’ button.



The function is now seen in a test step as part of the test case



14. Execute Deeplink

You can use this functionality to record a test step to execute a deeplink in your app. For this, provide the deeplink URL (and a Package name, if required) and click on the 'Save' button.

15. Get Current Context

16. Set Current Context

17. Execute Database calls

Clicking on this button enables you to execute Database queries.

Execute DB Query

Provider

Oracle

Connection String

username/password@db_host:port/db_name

Query

Result Row #

SAVE

CLOSE

1. On the 'Execute DB Query' window that opens, enter the following information:
 - *DB Provider* - Presently we support Oracle
 - *Connection String* - This should be in the format: *username/password@db_host:port/db_name*
 - *DB Query* - provide the database query to be executed
 - *Result Row*
2. Click on the 'Save' button to execute the query
3. A corresponding test step is created in the test step table. The output of the query is visible in the 'Return Data' section of the test step.

18. Manage Android Permission Alerts

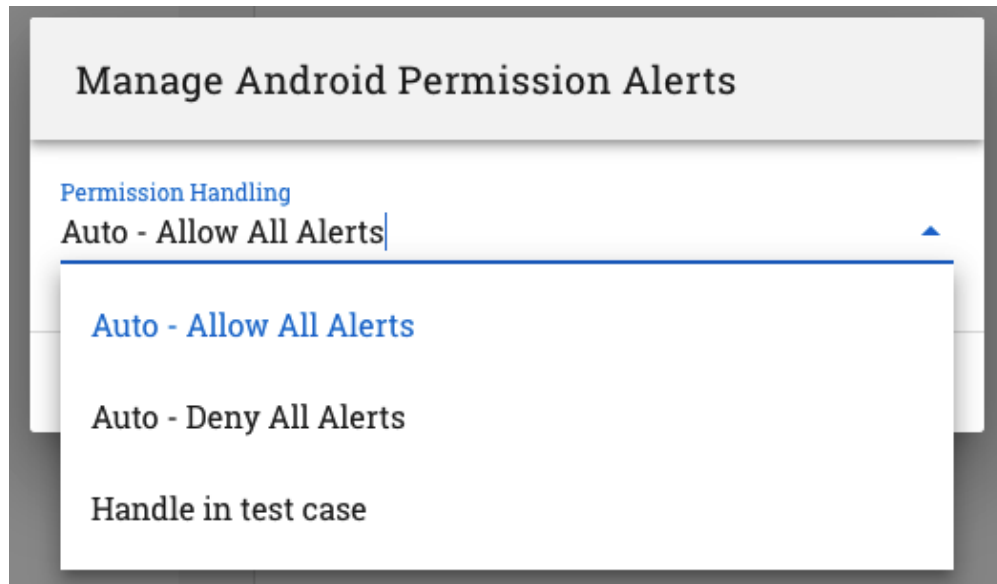
On Android version 6.0 and later, Android provides the user the option to 'Allow' or 'Deny' different kinds of permissions to an app on the user's mobile device. These include permissions to access 'Contacts', make phone calls, send SMS, access location, etc.

On RobusTest, after you have started an automation test session for an app, if Android Permission Alerts pop-up, then, *the default way of handling these alerts is to 'Allow' all such requests.*

No test steps are created for the handling of these alerts.

However, RobusTest provides you a highly customisable way to handle these permissions too.

1. Click on the 'Manage Android Permission Alerts' button on the header



2. On the pop window that opens, you have the following options:

- *Auto - Allow All Alerts*: This is the default option on RobusTest. If this option is selected, then the 'Allow' button is clicked on for all Android Permission Alerts that pop up.

These alerts are handled automatically. No test step is created for the same.

- *Auto - Deny All Alerts*: If this option is selected, then the 'Deny' button is clicked on for all Android Permission Alerts that pop up.

These alerts are handled automatically. No test step is created for the same.

- *Handle in test case*: If this option is selected, then the user is provided the flexibility to determine how each permission alert is to be handled.

For each Android Permission Alert that pops up, you can choose to click either on the 'Allow' button or on the 'Deny' button.

A corresponding test step is created for the same. This will be a part of the test case.

6.5 Test Session - Session Management

The vertical menu to the left of the test step table provides you various options regarding management of the test session

1. Show Recorded Steps

Tapping on this button displays the test step table. You can use this option to get back to recording your automation test steps, in case you had earlier moved away from it while using any of the options to be described below

2. Session Information

Clicking on this button provides you all details regarding the automation test session in progress. This includes information about:-

- the build - App name, build version, last uploaded, etc
- the device - Device name, ADB ID, OS version, etc,
- the project - Project name, Last build uploaded, etc

3. Device Log

You can view the logcat report in a tabular format with options to filter by the log level. You also have the option to download the log in CSV format.

4. Run ADB command on device

Sometimes, as part of your testing, you may want to run a few ADB commands on the device under testing. You can do so by clicking on the 'Run ADB commands on device' button.

This brings up the Command Line Interface from where you can execute adb commands directly on the device.

5. Copy session information

Clicking on this button captures all relevant information about the session in progress onto the clipboard. This information includes information about when the test session began, details of the app build, details of the device being used, etc.

If you face an issue while in an automation test session, you can capture the session details using this option and then paste this info in a support ticket that you log.

6. Turn ON image-based element verification

Clicking on this enables image-based verification. You can read more about this in the section on image-based-verification

6.6 Creating and Managing Test Cases

On the top right corner of your Automation session, you will see a group of buttons that help you manage your test cases

<input type="checkbox"/>	#	Test Step					
=	<input type="checkbox"/>	1	Tap on text "LOGIN"				
=	<input type="checkbox"/>	2	Tap on text field "text_input_email_login"				
=	<input type="checkbox"/>	3	Type in text field "user@robustest.com"				

Let's go through each of them

- Import test case - You can import an existing test case into an automation test session by clicking on this button.
 - On clicking on the 'Import test case button', a window pops up with a list of all test cases being displayed.
 - Click on the test case you want to import and then click on the 'Save' button
 - The test case is now seen loaded into the test session. You can now execute test steps in the test case
- Remove imported test case - Let's say you imported a test case, worked on it and updated it. Now you would like to create a new test case afresh. How do you clear the test step table?

You can do so by clicking on the 'Remove imported test case' button. This removes the test case from the test session and clears the test step table. You can now go ahead and start recording a new test case.

This button will be visible only after an existing test case has been imported or a new test case is created & saved.

- Update test case - Sometimes you may want to update an existing test case. In such cases, you first import the test case into a test session and then proceed to make the necessary changes.

Once your changes are complete, click on the 'Update test case' button. An 'Update test case' window pops up.

You can choose to modify the name or description of the test case on this window.

You also have the option to add tags to mark the test case. E.g. Smoke, Regression, Login, Payments, etc.

To create a tag:

- Enter a tag name on the 'Test case tag' field
- Hit 'Enter'. The tag is now created

Now click on the 'Update' button to save the changes made to the test case.

This button will be visible only after an existing test case has been imported or a new test case is created & saved.

- Save new test case - You can save a newly created automation test case clicking on this button.

If you have imported an existing test case, then, on clicking on the 'Save new test case' button, you can save this test case with a new name and create a new test case. The originally imported test case remains the same as it were.

Creating and running automation test cases considerably speeds up the testing process and helps reduce human error.

RobusTest enables you to create automation test cases in a quick and easy way with minimal to zero coding.

Any Automation testing plan should comprise of the following 3 processes:

1. Creating an automation test case
2. Running or Executing an automation test case
3. Generating test run reports

In this section, let's look at how we can create automation test cases

In order to start an automation test session:

1. Click on the 'Automation' icon on the Project Dashboard

A device selection screen now pops up. You may search for a device on the 'Device selection screen' based on device name, platform version, screen size, hardware configuration (e.g. Memory and CPU), node name, node IP, etc.

2. Select the device you wish to test on by clicking on the 'Select' button under the device

The device screen now comes up and you can see that your app is installed on the device.

Your automation test creation session is now in progress.

Creating Automation Test Cases

RobusTest not only allows you to record different kinds of user actions but also provides you with a host of features that enable you to automate a wide variety of complex user scenarios.

Let's have a look at how RobusTest helps you create automation test cases

1. *Recording user actions*
2. *Test Step Management*
3. *Test Session - Device Management*
4. *Test Session - Test Case Management*
5. *Test Session - Session Management*

6. *Creating and Managing Test Cases*

You can click on the 'End Session' button to end the recording session.

Executing Test Runs

7.1 Understanding Test Suites

A Test Suite is a collection of test cases that you would like to execute. When a test suite is submitted for a run, all test cases present within the test suite are executed.

A test suite can also be a logical grouping of test cases.

E.g. You can create test suites consisting of test cases that pertain to a Smoke Test or a Regression Test or one that groups together test cases specific to a module, say, payment, customer acquisition, order creation, etc.

a. Creating a test suite

1. Click on the Test Suites icon on the Project Dashboard. You are now on the 'Test Suite' page.
2. Click on the 'Create Test Suite' button (i.e., the '+' icon) on the top right corner. The 'Create Test Suite' page opens up.
3. On the 'Create Test Suite' page, enter a name and description for the test suite and hit the Enter key. You now see that the 'Add Test Cases' button is enabled. (*Note: Entering the test suite description is not mandatory*)
4. Click on the 'Add Test Cases' button. * You are now on the 'Modify Test Suite' page. * On this page you can see a list of all automation test cases that you have created. * You can search for a test case using the 'Search' text box provided, if required. * Once you have identified the test case or test cases you would like to execute, add them to the test suite by clicking on the green coloured 'plus' icon to the left of the test case name. On being selected the 'plus' icon changes into a red coloured 'minus' icon. * You can remove a test case that has already been added by clicking on the red coloured 'minus' icon. * Test cases are executed in the order in which they are present in the test suite. You can change the order by dragging and dropping the test case entries in the test suite. * You can view the list of test cases you have selected by clicking on 'Show selected test cases'. Only the selected test cases are now displayed. This is how your final test suite would look like. * You can view the complete list of test cases at any time by clicking on the 'Show all test cases' button.
5. Now that you have selected the required test cases, save your changes by clicking on the 'Save Test Suite Details' button at the top right.

You have now created a Test Suite!

b. Modifying Test Suites

All test suites that have been created are visible on the ‘Test Suites’ page.

For each test suite, the following information is visible: * test suite name * test suite description * name of the user who created the test suite * name of the user who last updated the test suite and the date and time of updation

To the right side of each test suite entry there are a set of ‘action’ buttons provided.

Let’s have a look at each of these buttons:

Run Test Suite: This option is denoted by the ‘Play’ button icon. Clicking enables the user to execute a test run. We shall cover this functionality in the “*Creating and Executing a Test Run*” section

Modify Test Suite: You can add or remove test cases or rearrange their order in the test suite by clicking on this button

Clone Test Suite: This option enables you to create a copy of the test suite. You can then rename and modify the test suite as required.

Delete Test Suite: You can delete a test suite by clicking on this button and providing confirmation.

7.2 Creating and Executing a Test Run

Now that we know how to create and modify a test suite, let us understand how to go about executing a test suite in a test run.

A test run constitutes of a combination of the following 5 components:-

- a. *Test Suite*
- b. *Run Settings*
- c. *App build*
- d. *Devices*
- e. *Data sets*
- f. *Run Mode*

a. Test Suite

This is the primary component. It determines the group of automation test cases that will be executed in a test run. Creation of a test run begins by selecting the test suite to be run.

On the Test Suites page, click on the ‘Run Test Suite’ button (i.e., the ‘*Play button*’ icon) for the test suite that you wish to execute. You are now on the ‘Create Test Run’ page.

b. Run Settings

Run Settings determine how your automation test run session is going to behave.

From the ‘Run Settings’ drop down, select the setting that you would like to use. If no custom Run Setting is used, then the ‘System Default’ run setting will be used.

Details of a chosen Run Setting can be viewed by clicking on the ‘information’ icon to the right side of the ‘Run Settings’ drop down

To know more about Run Settings, go through the *Run Settings* page.

c. App Build

The 'Select Build' drop down displays a list of all the app builds uploaded to the project in which you created your automation test cases.

From this drop down, choose the right build version of the app on which you would like to execute your test cases.

d. Devices

One of the highlights of the RobusTest platform is its ability to execute a set of test cases on multiple devices in parallel.

On clicking on the 'Select Devices' drop down, you are provided with a list of devices on which you can run the test suite.

You can select a device by clicking on it in the dropdown. You can select multiple devices by clicking on each of those devices in the drop down.

Selected devices are visible in the 'Select Devices' field. Additionally, in the drop down, these selected devices are displayed in a blue coloured font.

Once a device is selected an entry is displayed on the 'Create Test Run' page corresponding to the selected device. This entry provides you information about the device name, the OS version running on the device, the device identifier and the dataset that is to be used for that specific device for the duration of the run.

To *unselect* a selected device, perform any one of the following actions:

- click on the same device in the drop down for a second time; *or*
- delete the entry corresponding to this device in the 'Select Devices' field using the keyboard; *or*
- on the list of device rows displayed below the 'Select Devices' drop down, click on the 'Remove Selected Devices' button for the entry corresponding to the device to be removed from the run.

e. Data sets

Data sets are used to execute test cases on data values that are different from that used originally when these test cases were first created.

As mentioned above, for each device selected for a run, an entry is created and displayed below the 'Select Devices' dropdown.

On this entry is displayed the data set that is to be used while executing the test run on this device.

The data set drop down displays a list of all data sets that were created in the project in which the automation test cases were created. You can choose the data set of your choice from this dropdown.

If you do not specifically select a data set, then, the 'Original' data set, that was created by the system, is selected by default and the same will be used during the course of execution of the test run.

Details of the data set chosen can be seen by clicking on the 'View Data Set' button on the same row.

To know more about data sets, read the 'Using Datasets' section of the [Parameterisation of Data](#) page.

f. Run Mode

Test runs can be executed in one of two modes:

1. *Parallel*
2. *Distributed*

1. Parallel Run Mode

- In this mode, all test cases in the test suite are executed on each device that has been selected for the run.
- This mode is helpful in improving device coverage for your tests.

2. Distributed Run Mode

- In this mode, the test cases in the test suite are distributed among the devices that have been selected for the run and then executed.
- Initially each device is given a different test case for execution.
- When a test case has been executed to completion, the device is free. This device is then provided another test case from the test suite for execution.
- This mode is useful in increasing the speed of execution of your tests, resulting in faster turnaround.

Now that you have created your automation test cases, the next step is to execute or run them.

RobusTest enables you to run your automation test cases by first grouping them logically using test suites and then creating and executing a test run.

1. *Understanding Test Suites*

2. *Creating and Executing a Test Run*

CHAPTER 8

Run Settings

Whenever you start a new test session - whether Manual or Automation - there are certain default configuration values that are being used to determine the way your test session behaves.

'Run Settings' enable users to customise these default configuration values

You can create a new Run Setting by clicking on the 'Run Settings' icon on the Project Dashboard

RobusTest allows you to create 4 types of Run Settings. To know more, click on each type of run setting below

1. run-settings-manual
2. run-settings-automator
3. run-settings-runner
4. run-settings-espresso

Configuring Run Settings

Now that you have created a run setting, let's see how you can configure your test sessions to use a customised run setting

a. Manual Test session

- Click on the 'Manual' icon on the Project Dashboard. The device selection screen now comes up.
- While on the device selection dialog, click on the 'Configure Session' icon on the top right corner.
- On the pop window that opens up, you will find a drop down called 'Run Settings'. You can select the Manual Run Setting that you would like to use by clicking on it in this drop down.
- You can view the various key-value pairs in a selected run setting by clicking on the information icon to the right of the dropdown.

b. Automation Test session

- Click on the 'Automation' icon on the Project Dashboard. The device selection screen now comes up
- While on the device selection dialog, click on the 'Configure Session' icon on the top right corner.

- On the pop window that opens up, you will find a drop down called 'Run Settings'. You can select the Automation Run Setting that you would like to use by clicking on it in this drop down.
- You can view the various key-value pairs in a selected run setting by clicking on the information icon to the right of the dropdown.

c. Automation Test Run

- Click on the 'Test Suite' icon on the Project Dashboard.
- Click on the 'Play' icon corresponding to the test suite you would like to execute
- On the page that opens up, select the automation run setting of your choice from the 'Run Settings' drop down
- You can view the various key-value pairs in a selected run setting by clicking on the information icon to the right of the dropdown.

CHAPTER 9

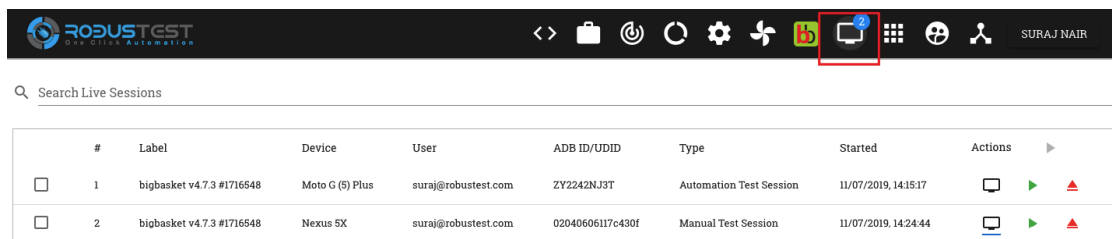
Live View

On the header, in the RobusTest application, you will find a link for Live View.







This page displays a list of all active manual, recording and automation run sessions belonging to the user who has logged in. Even Appium test runs made using RobusTest Hub can be accessed here.

Each session entry provides you information about:

- the app build being tested
- the device that the app is running on
- the ADB ID of the device being used
- the type of test session - i.e., Manual, automation or test run
- the user who has started the test
- the date and time of commencement of the test session

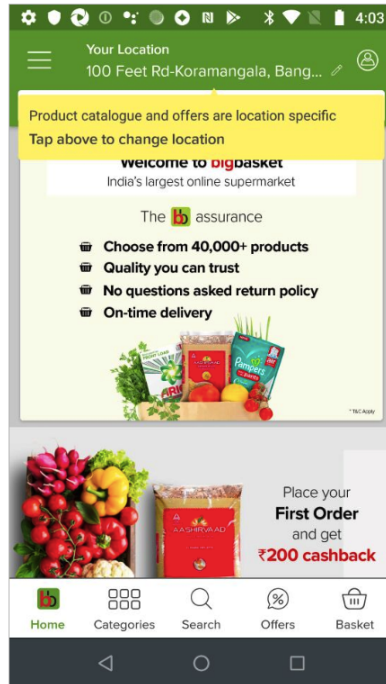


The screenshot shows the RobusTest application interface. At the top, there is a header with the RobusTest logo and a navigation bar containing various icons. A red box highlights the 'Live View' icon (a monitor with a play button). Below the header, there is a search bar labeled 'Search Live Sessions'. The main content area displays a table of live sessions.

#	Label	Device	User	ADB ID/UDID	Type	Started	Actions	
<input type="checkbox"/>	1	bigbasket v4.7.3 #1716548	Moto G (5) Plus	suraj@robusttest.com	ZY2242NJ3T	Automation Test Session	11/07/2019, 14:15:17	  
<input type="checkbox"/>	2	bigbasket v4.7.3 #1716548	Nexus 5X	suraj@robusttest.com	02040606117c430f	Manual Test Session	11/07/2019, 14:24:44	  

In addition to the above info, you will also find the following buttons:-

- Live Screen button
 - You can view the live screen of a device currently being used by the user by clicking on this button (i.e., the monitor icon).
 - A new window will open with the screen of the device. In case of a test run, one can also see the test steps on the live screen.
 - To be able to control the device screen during a test run session, append the following to the live screen URL: **?deviceControl=true**



Test Case : Search for wheat App : Big Basket		Device : Moto G (5) Plus v 8.1.0 Node : 192.168.0.145	
#	Serial	Step Name	
1	1	Start Test	
2	2	Start App	
3	3	Tap on 'Explore Bangalore Store'	
4	4	Tap on 'Search'	
5	5	Type search item	
6	6	Hit Enter	

- Resume Session
 - This button is denoted by a green coloured 'Play' button icon on the live session entry.
 - It comes in handy when you have accidentally closed the browser tab on which your Manual or Automation test session is running. Now, on clicking this button, you can resume your Manual or Automation test session.
- Release Device
 - You can end an active manual test or test recording session from this page by clicking on the this button.
 - This frees up the device for a new test session.

View multiple device screens in parallel

On the Live View Sessions page, you also have the option to view multiple device screens at the same time on your monitor.

For this:-

1. enable the check box at the left side of each live view session entry, whose device screens you would like to view. At least two live session entries should be selected
2. Click on the 'Start Multiplexing' button
3. The user can now view the selected devices in Multi-device mode

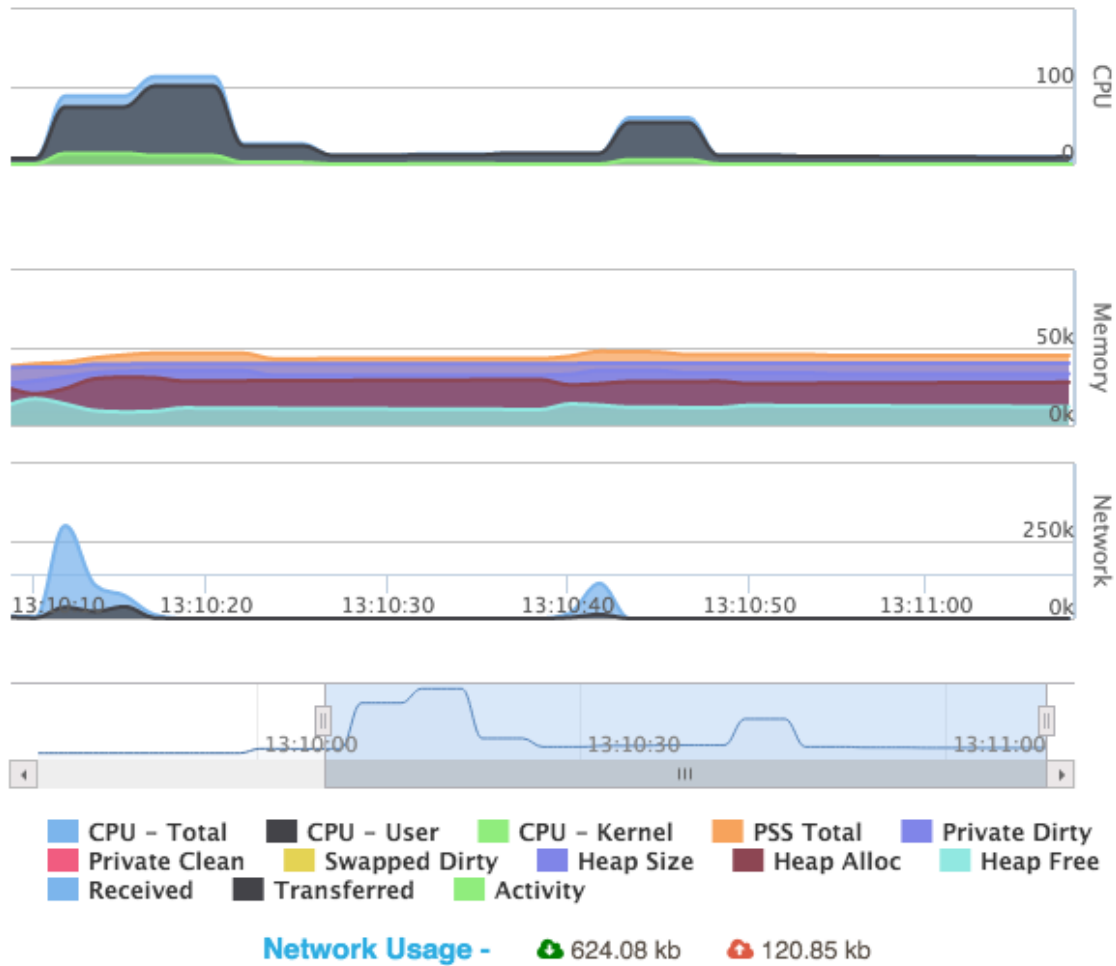
CHAPTER 10

Performance Testing

Your customers are using a variety of devices with varying hardware configuration, a range of Android versions (yes Android fragmentation is for real. Click here <http://developer.android.com/about/dashboards/index.html> for the official numbers from Android and check here for an even more in-depth analysis of what things look like <http://opensignal.com/reports/2015/08/android-fragmentation/>). We missed mentioning that many (in fact, most) manufacturers create their own flavor using stock Android.

In such a situation, it is important that you ensure that your app runs (without problems) on the devices that your customers are using. One of the most important aspects of mobile app quality is the performance of the app and at RobusTest we adopt a performance-first approach to mobile app testing. What this means for you is that whether you are in a manual test session or are running your automation tests, performance metrics of the app is just a click away.

Manual Testing



When you are running your manual tests for exploratory testing or executing your tests on the device, you have the option of monitoring the memory, CPU and network usage for your app. You can monitor both real time as well as cumulative network usage.

Memory Usage

You can get all the important memory usage parameters from the performance stats section. You can view the following link to investigate each memory usage metric <http://developer.android.com/tools/debugging/debugging-memory.html>

CPU Usage

In the CPU Usage section you can track the total CPU usage for your app. Do not panic when it goes over 100% - that happens in case of multi-core processors. On second thoughts, an over 100% usage can be a matter for concern especially if there are other processes/applications fighting for CPU time.

Network usage

Monitor the amount of data your application is exchanging - consuming and uploading - in real-time as well for a particular test session.

11.1 Debugging Test Case Failures

Once your test run completes its run, you can view the detailed execution report in the Reports section.

Reports are categorized by devices and test cases.

Device based categorization

Test Case based categorization Each test run generates three different reports

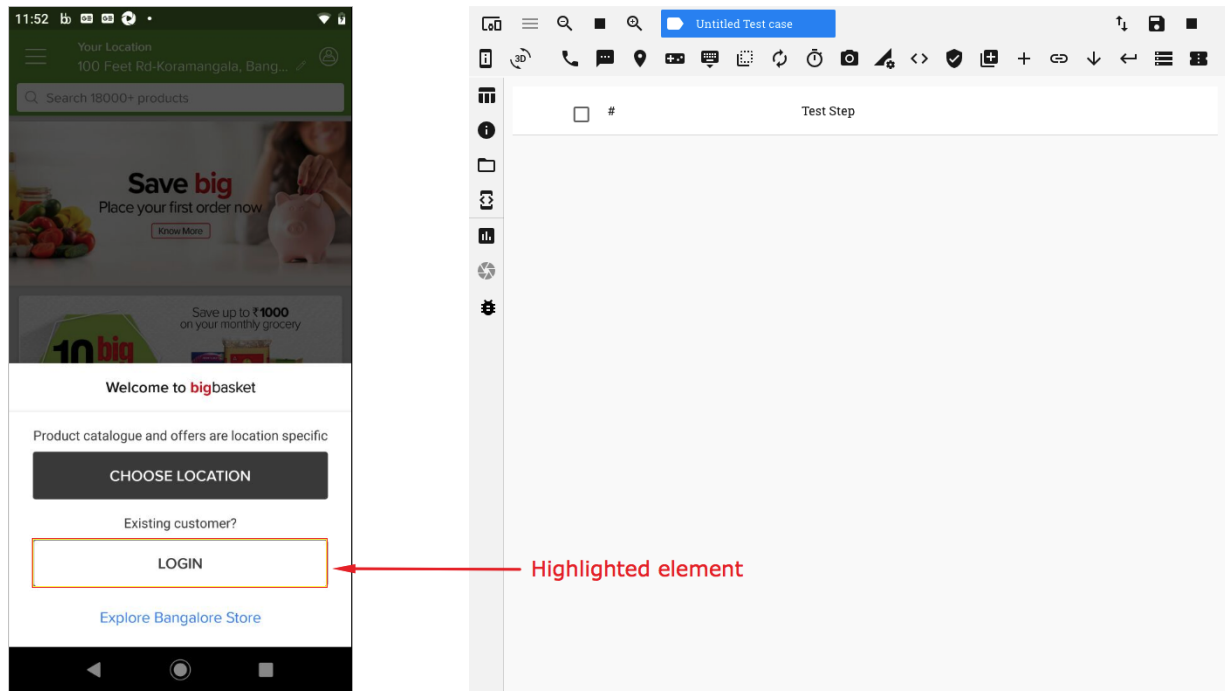
1. Slideshow: of all the different screens encountered when the test was run
2. Analytics: of all important device, platform and device metrics like memory, CPU, Activities, GC Analysis.
3. Functional Report: of the test run which includes details of each step that was run include pass/fail status and screenshots for each step.

12.1 Verification

12.1.1 Using the 'Verify Element' option

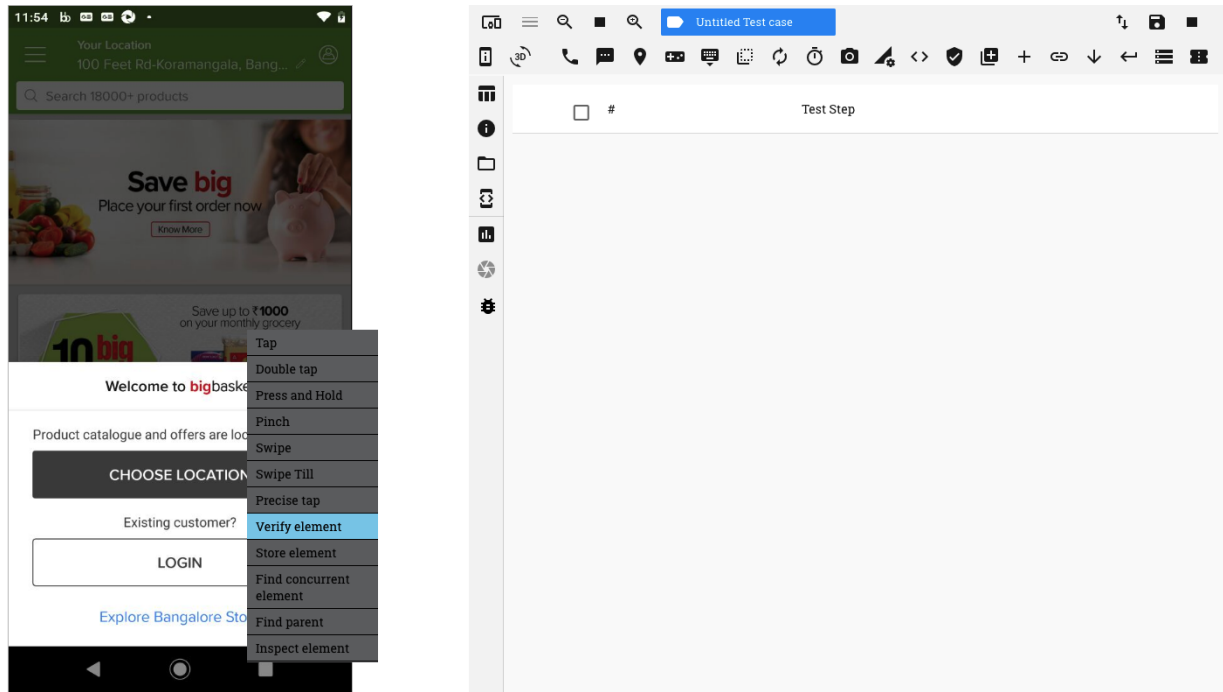
In an Automation test session, as you move the mouse pointer over the device screen, you will see different rectangular boxes being displayed.

Each box highlights an element on the current app screen/page.

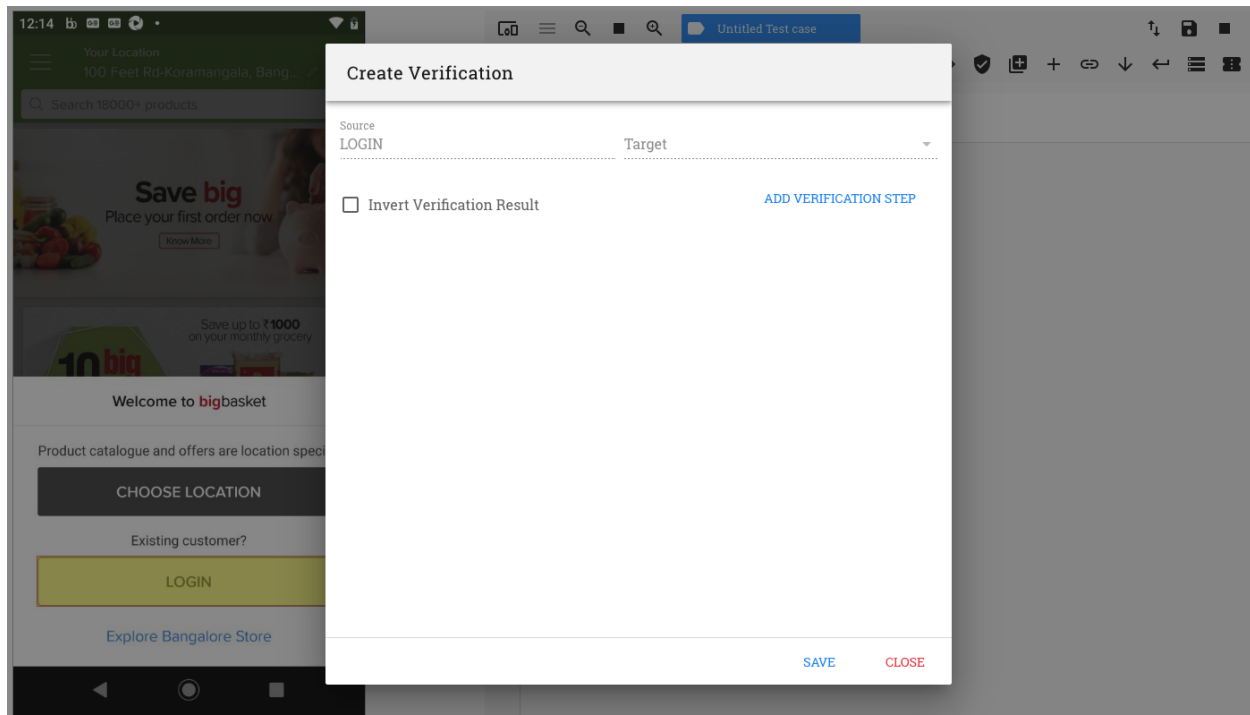


Hover the pointer over an element (so that a rectangle is displayed) and perform a left-click. You will see a context menu open up.

On the context menu, click on the option 'Verify Element'



You now see a 'Create Verification' window pop up



On this window you can see the following:-

1. Source

2. Target
3. 'Add Verification Step' button
4. 'Invert Verification Result' checkbox

We shall understand each of the above terms as we proceed further

The first step for any verification is to identify the 'Source' element

When executing a 'Verify Element' action, the 'Source' is always going to be the element on which the left-click was performed. Hence, the 'Source' field on the top left corner of the window is disabled by default.

If the Source element has a value present for its 'text' attribute, then, this value will be displayed under 'Source'. In the example above, the element has the value 'Login' for its 'text' attribute.

The second step is to identify a 'Target' element

This can be done in two ways:-

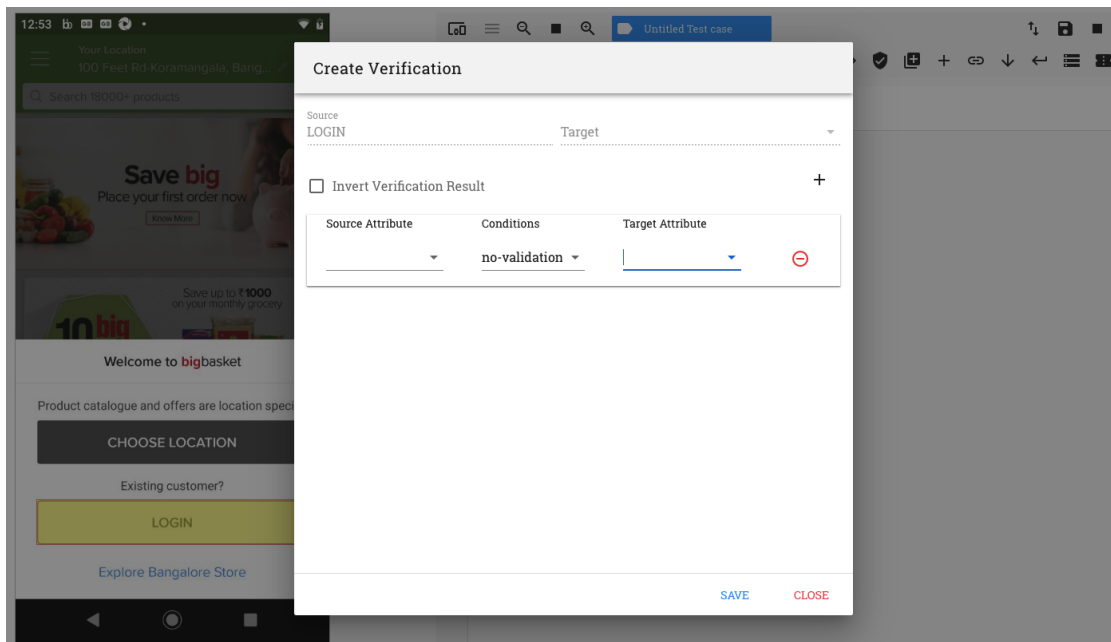
1. By directly using the 'Add Verification Step' button
2. By using the 'Target' drop down

1. Directly using the 'Add Verification Step' button

The 'Add Verification Step' option is used when you want to compare the value of the Source attribute against:

- a. its default value populated; or;
- b. a custom value

Click on the 'Add Verification Step' button. A verification row is seen added



On the verification row, you need to select 3 values:-

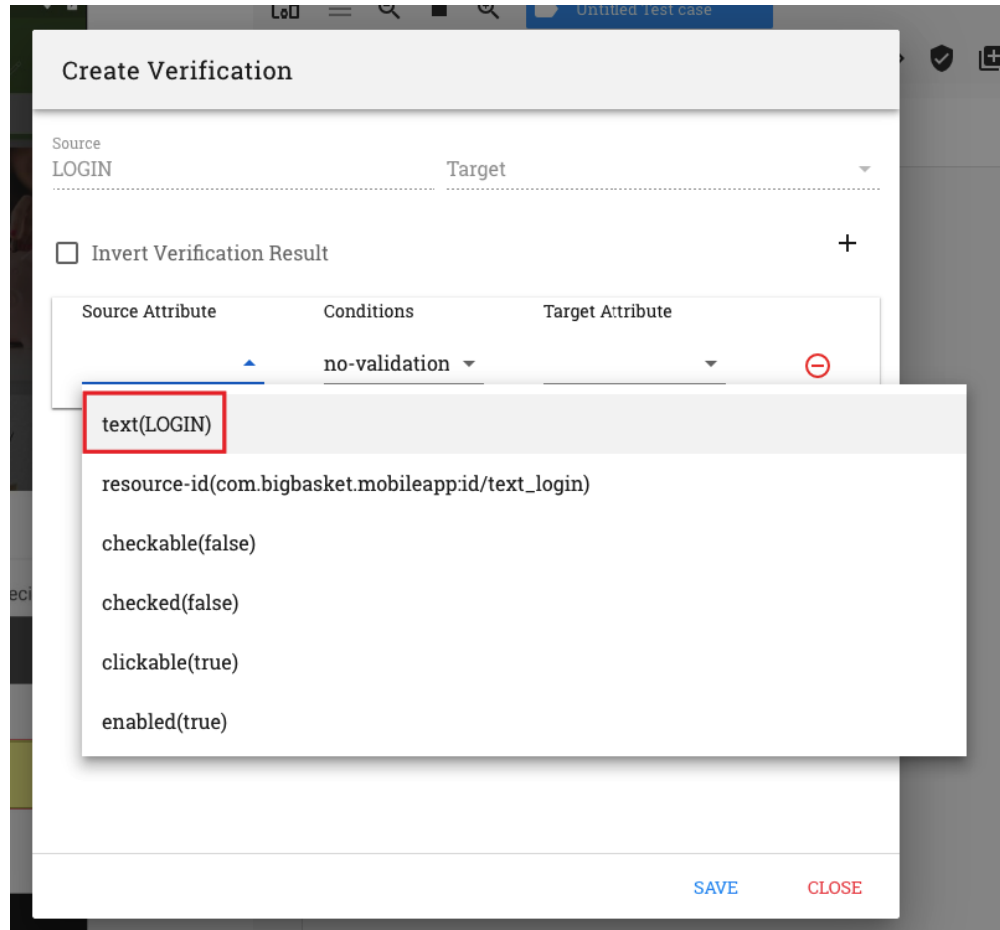
- a. the Source attribute
- b. the condition to be evaluated (or comparison criterion)
- c. the Target attribute

a. the Source attribute

On clicking on the 'Source' attribute drop down, a list of all the attributes associated with the source element, along with their values, are displayed.

Select the attribute whose value you want to compare by clicking on it.

Let us click on the 'text' attribute in this example

**b. the condition to be evaluated**

Click on the 'Conditions' drop down. A list of various conditions can be seen.

The conditions '*is exactly*', '*contains*' and '*is contained in*' are used for string comparisons

The conditions '=', '<', '>', '<=' and '>=' are used for numeric comparisons

Select the condition to be checked by clicking on it.

Let us click on the condition 'is exactly' in this example

Create Verification

Source
LOGIN

Target

☐ Invert Verification Result

Source Attribute	Conditions	Target Attribute
text(LOGIN)	is exactly	

no-validation

is exactly

contains

is contained in

=

<

SAVE CLOSE

c. the Target attribute

By default, the Target element in this case is the same as the Source element

On clicking on the 'Target' attribute drop down, a list of all the values associated with the attributes of the element are displayed.

Also note that the text in this drop down is editable.

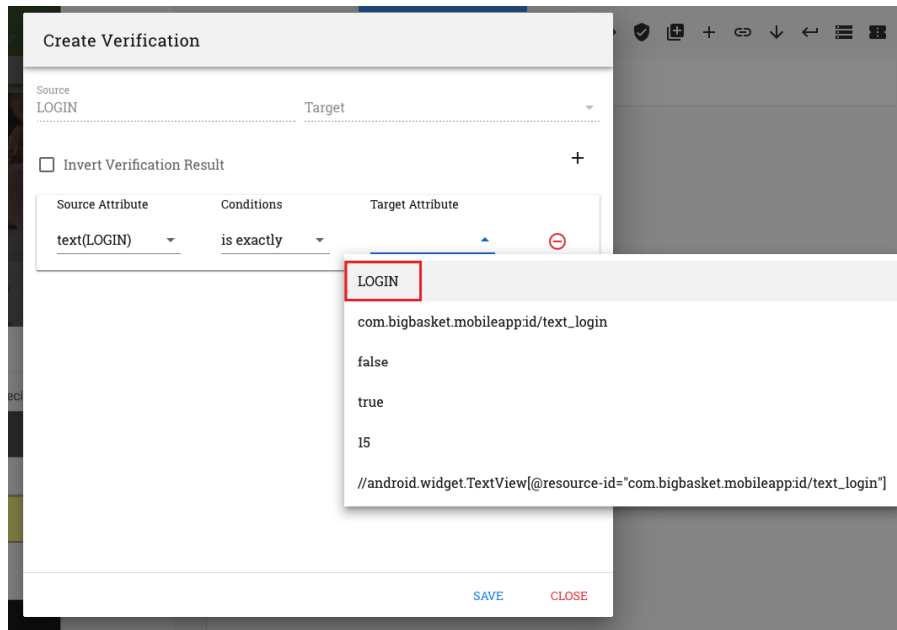
You now have two options here:-

1. Click on a value on the drop down. This value will be compared against the Source attribute value

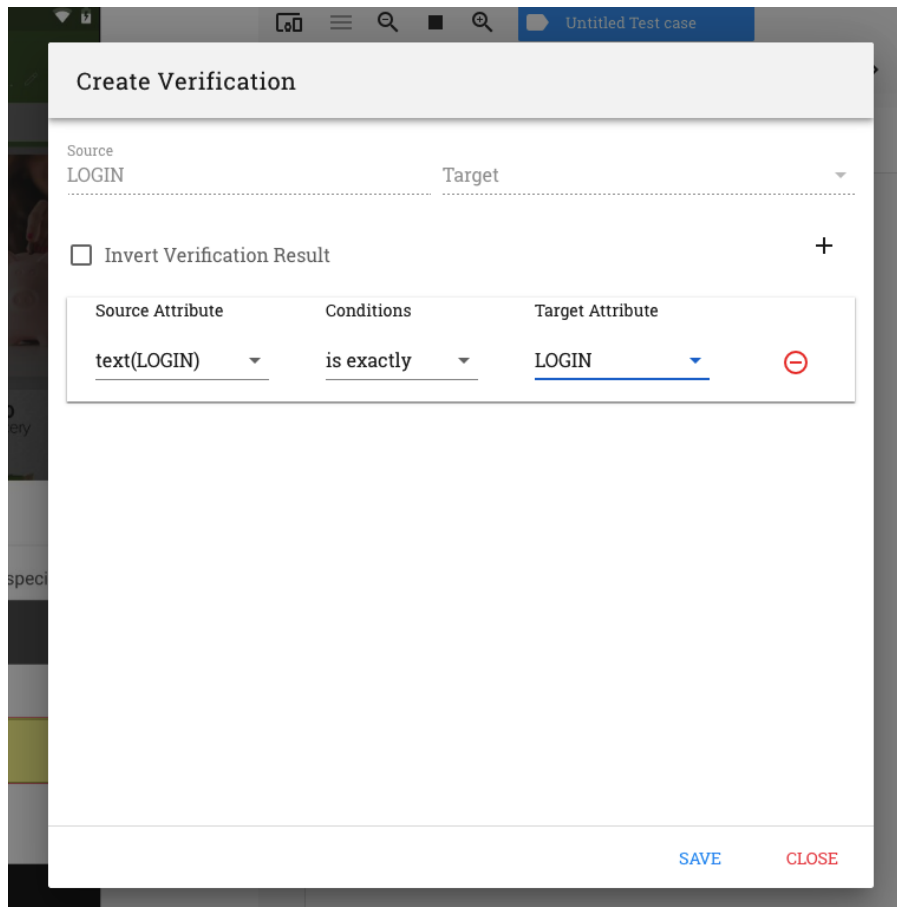
OR

2. Manually enter a value in the target attribute field to be compared against

1. Click on a value on the drop down.



This option can be used to ensure that the expected/default value is being displayed or populated for that attribute



The above verification can be read as follows:

Is the value in the text attribute of the Source element exactly the same as the value present in the Target attribute field ?

E.g. Is the text on the Login button exactly the same as 'Login'?

2. Manually enter a value in the target attribute field

This option can be used to enter custom values in the target attribute field manually

Create Verification

Source: LOGIN Target: [dropdown]

☐ Invert Verification Result

Source Attribute	Conditions	Target Attribute
text(LOGIN)	is exactly	log in

SAVE CLOSE

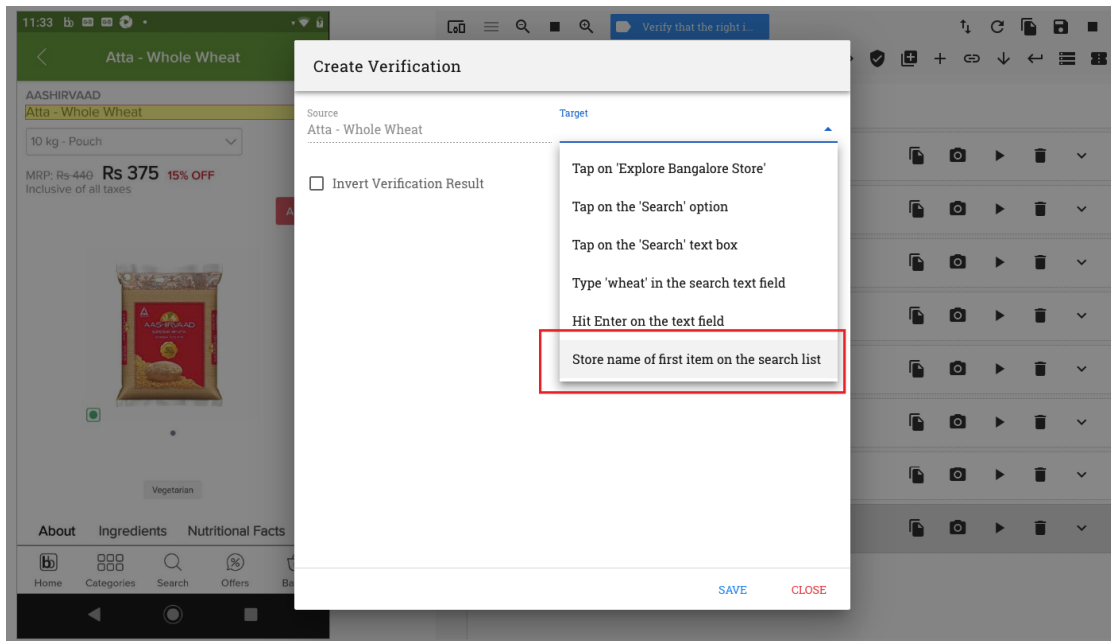
2. By using the 'Target' drop down

This option may be used when you want to compare the value of the Source attribute with that of a value stored in a previous test step

The previous test step, in this case, should always be a 'Store Element' test step. Hence you need to perform a 'Store Element' action on that element.

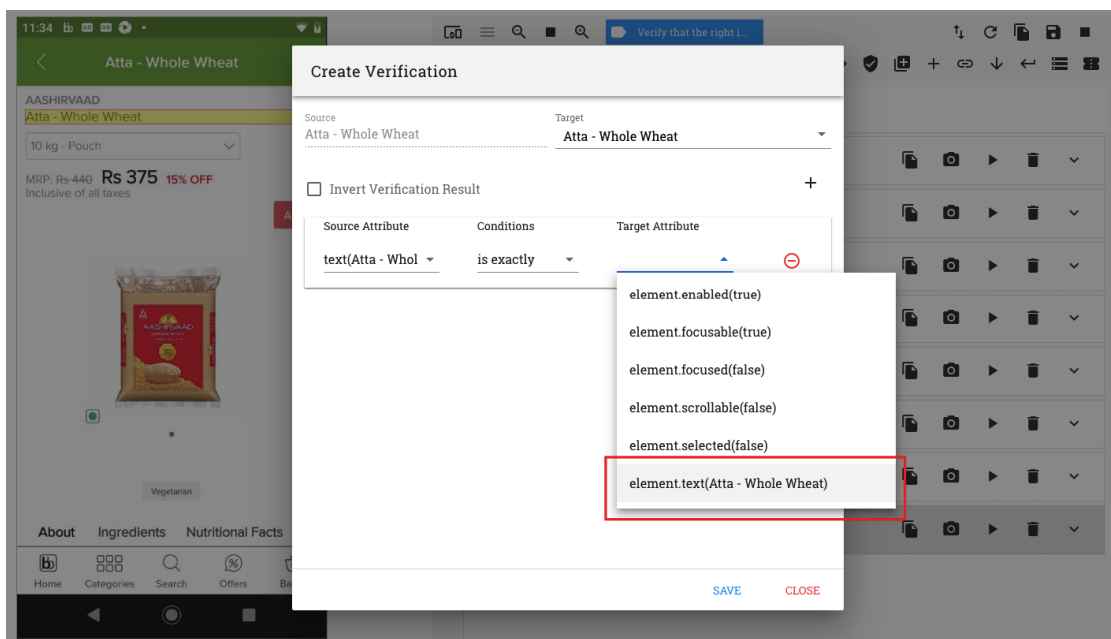
The details of the attribute values available for comparison may be viewed in the 'Return Data' section of the 'Store Element' test step that is recorded

The Target element may be selected from the list of test steps displayed in the Target drop down.



On clicking on the 'Add Verification Step' button. A verification row is seen

Select the Source attribute, the comparison criterion and Target attribute from their respective drop downs.



We now have the following comparison recorded:

Create Verification

Source: Atta - Whole Wheat Target: Atta - Whole Wheat

☐ Invert Verification Result

Source Attribute	Conditions	Target Attribute
<u>text(Atta - Whol</u>	<u>is exactly</u>	<u>element.text(Atl</u>

SAVE CLOSE

The above verification can be read as follows:

Is the value in the text attribute of the Source element exactly the same as the value present in the text attribute of the Target element ?

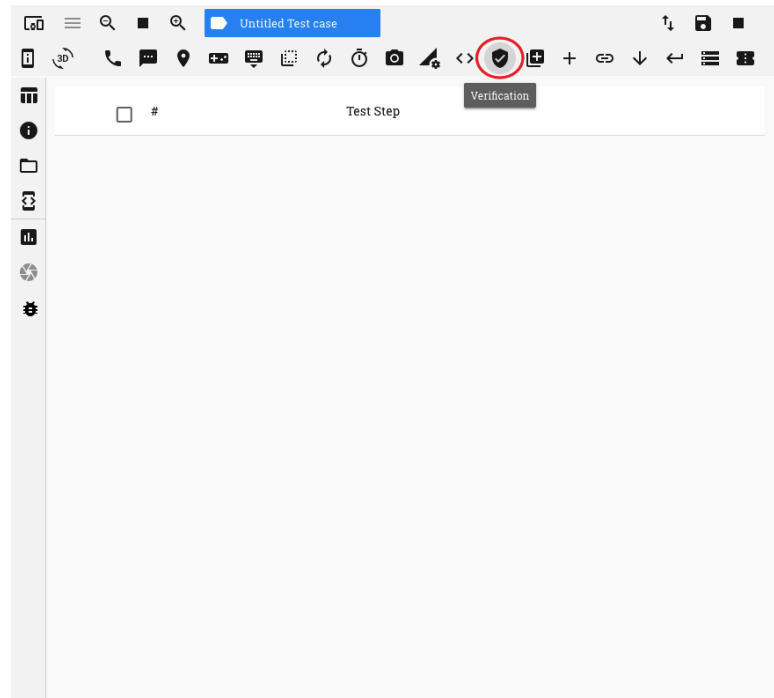
E.g. Is the item name on the product page exactly the same as the item name on the search list

Note: The Target element will always be a 'Store' test step.

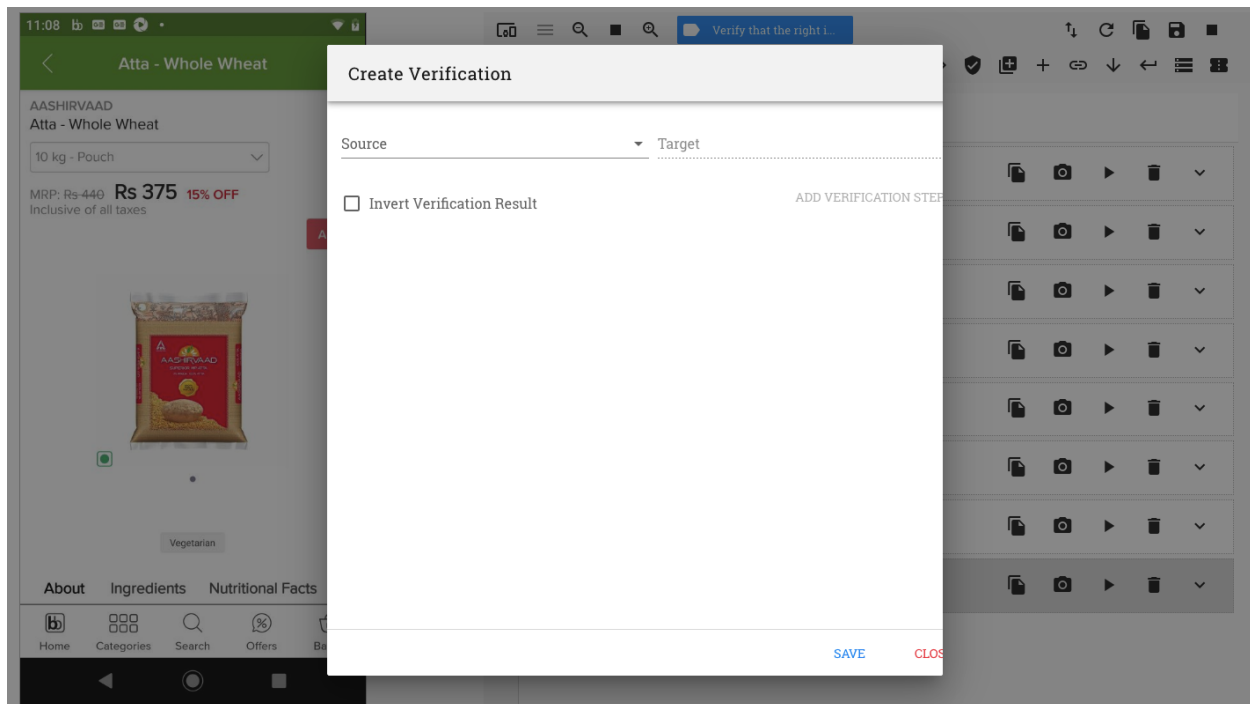
12.1.2 Using the 'Verification' button option

The second way in which a verification step can be recorded is by clicking on the 'Verification' button on the top horizontal menu

This method is used when the attribute values of two different test steps are to be compared with one another.



On clicking on this button, the 'Create Verification' pop up window comes up



In this case, both the Source and Target elements to be used for comparison have to be selected from their respective dropdowns

A point to be kept in mind is that in order to do a comparison, it is imperative that **the test steps selected in the two drop downs are both 'Store' test steps**

Hence, a 'Store Element' action should be performed on both, the Source and Target elements that are to be compared

The details of the attribute values available for comparison may be viewed in the 'Return Data' section of the two 'Store Element' test steps that are recorded

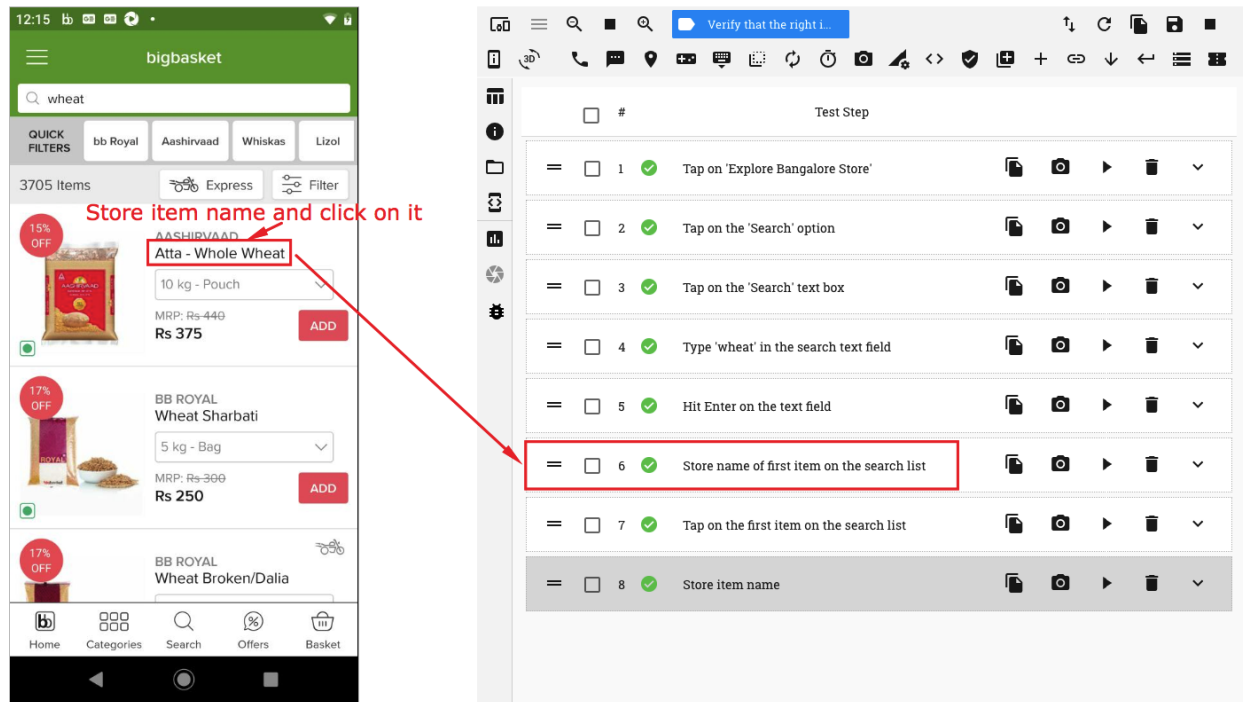
Let us look at an example to understand this method of verification

E.g. Let us say, we are on an app page where a list of items are being displayed. We need to verify that the name of the first item on the list is displayed correctly if we go to the item's product page by clicking on it

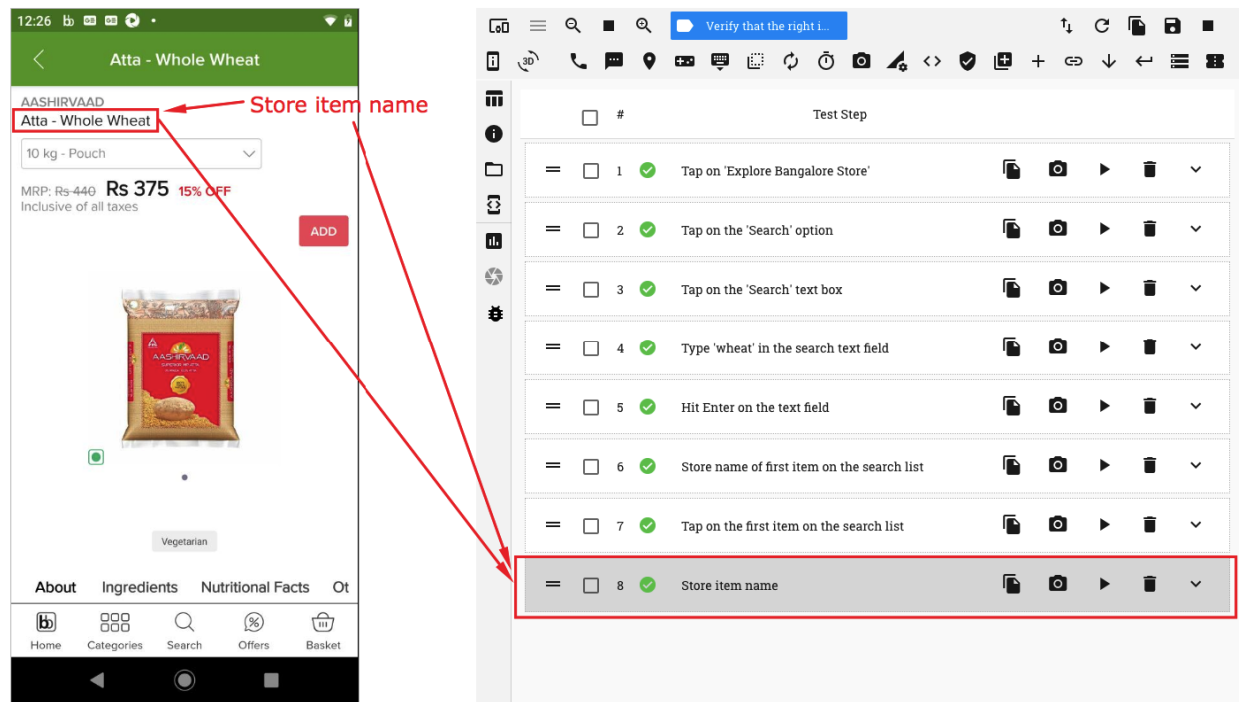
The image shows a screenshot of the BigBasket app interface on the left and a RobusTest test step list on the right. The app interface displays search results for 'wheat' with three items: Aashirvaad Atta - Whole Wheat (10 kg - Pouch, Rs 375), BB Royal Wheat Sharbati (5 kg - Bag, Rs 250), and BB Royal Wheat Broken/Dalia. The first item, 'Aashirvaad Atta - Whole Wheat', is highlighted with a red box and labeled 'Item Name' and 'Item 1'. The RobusTest test step list on the right contains 8 steps, with the 8th step, 'Store item name', highlighted in grey.

#	Test Step
1	Tap on 'Explore Bangalore Store'
2	Tap on the 'Search' option
3	Tap on the 'Search' text box
4	Type 'wheat' in the search text field
5	Hit Enter on the text field
6	Store name of first item on the search list
7	Tap on the first item on the search list
8	Store item name

Perform a 'Store Element' action on the item name. A store test step is seen recorded. Now click on the item name to go to the product page

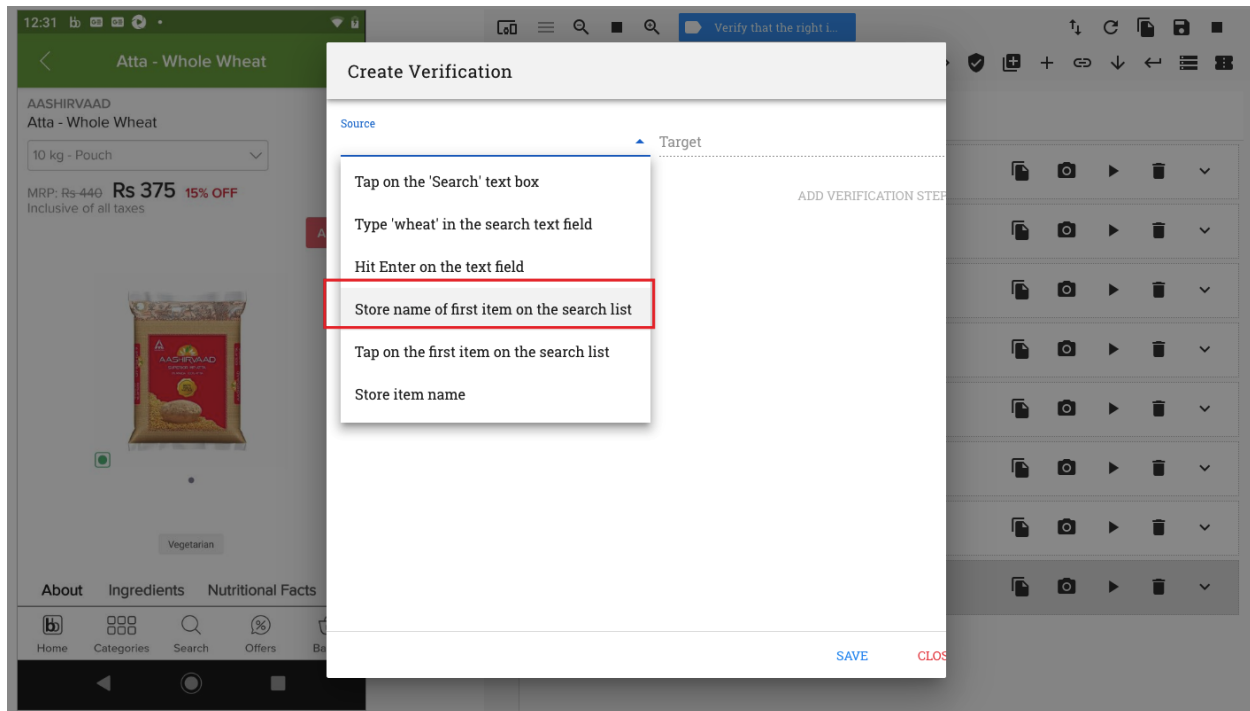


On the Product page, perform a 'Store Element' action on the item name

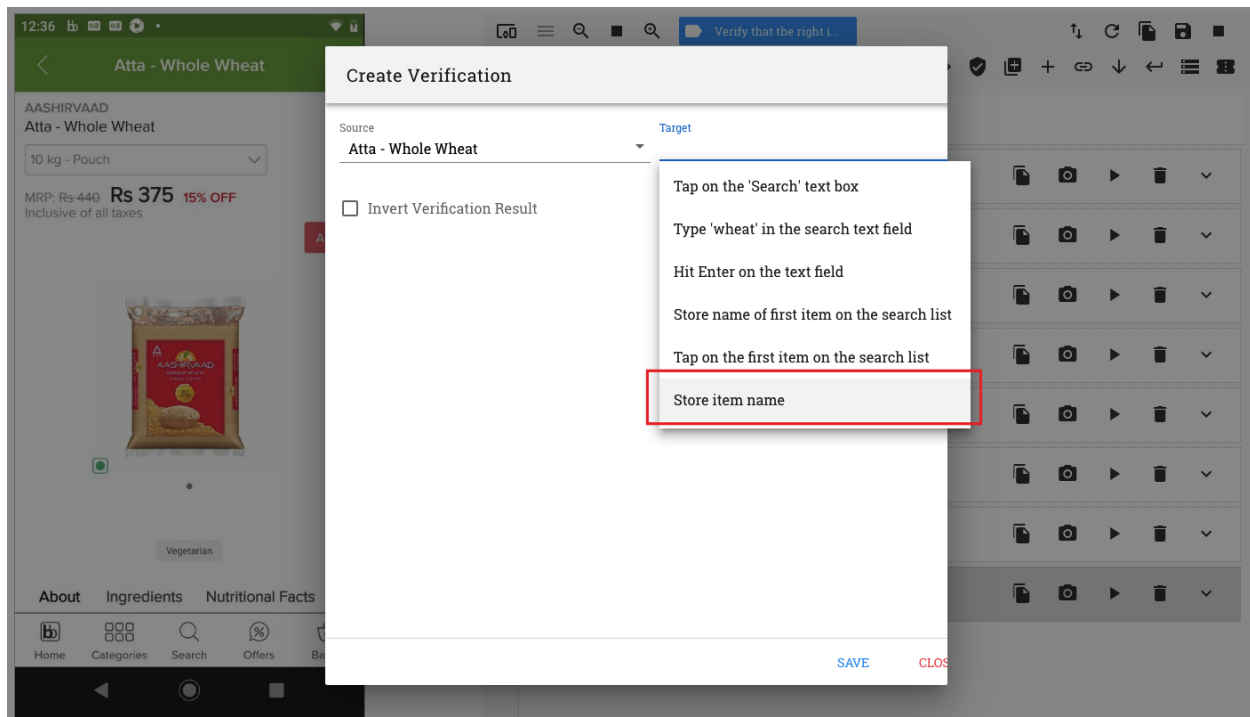


Now, click on the 'Verification' button on the horizontal menu bar at the top. A 'Create Verification' window opens up

On the 'Create Verification' window, click on the 'Source' drop down and select the first 'Store' test step, i.e., the one where details of the item name of the first item in the list are stored



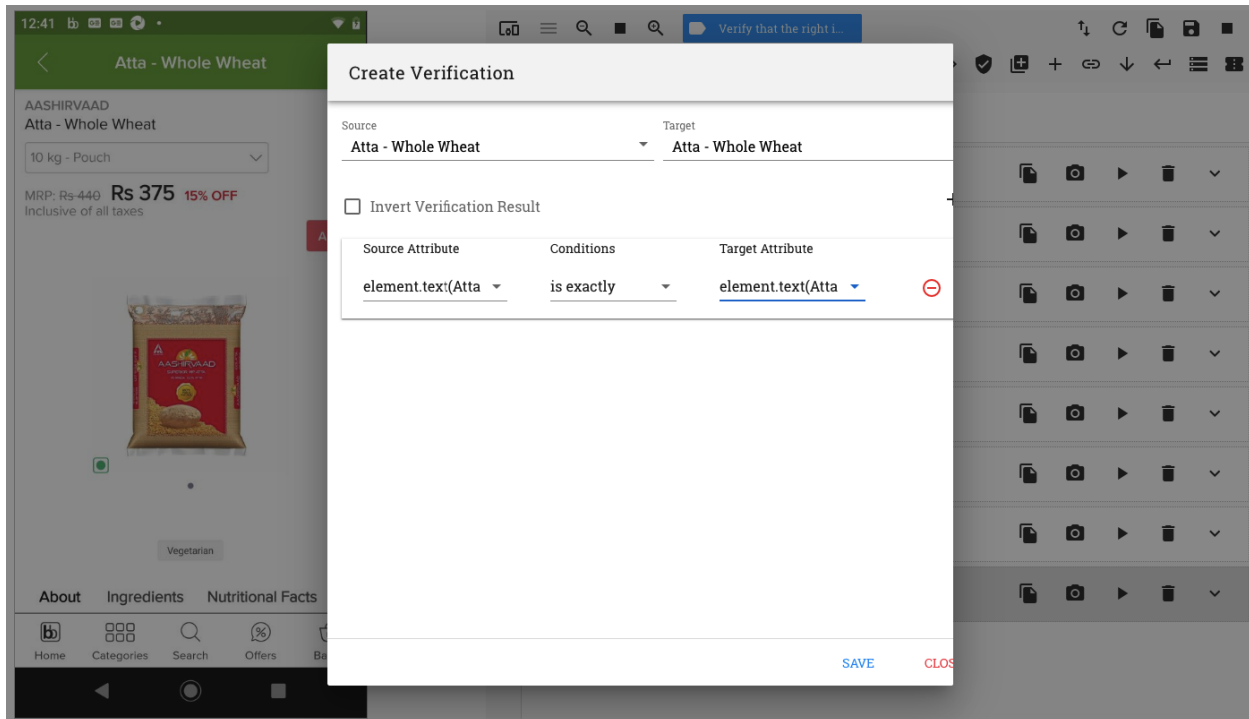
On the 'Target' drop down, select the second 'Store' test step, i.e, the one where the details of the item name on the product page are stored.



Click on the 'Add Verificaiton Step' button. A verification row is seen created.

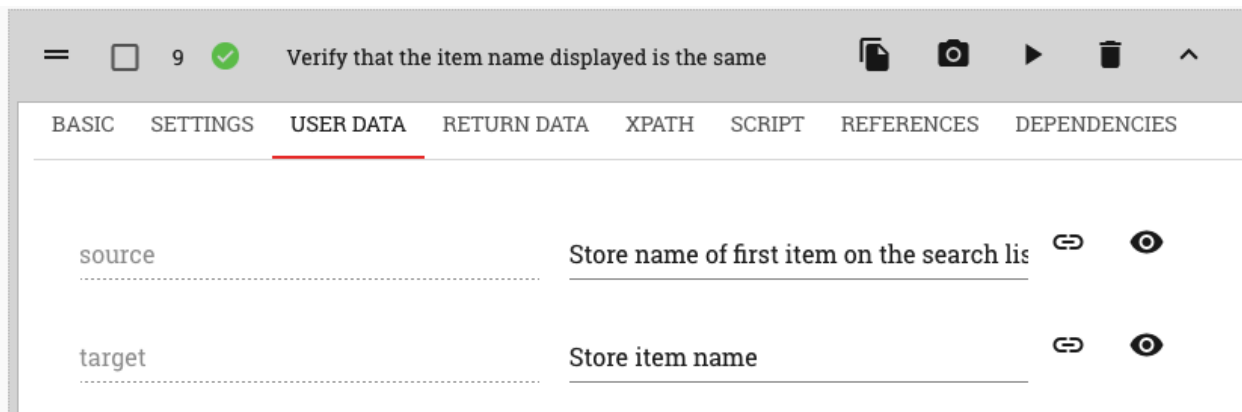
On the verification row, select the source and target attributes as well as the condition to be evaluated.

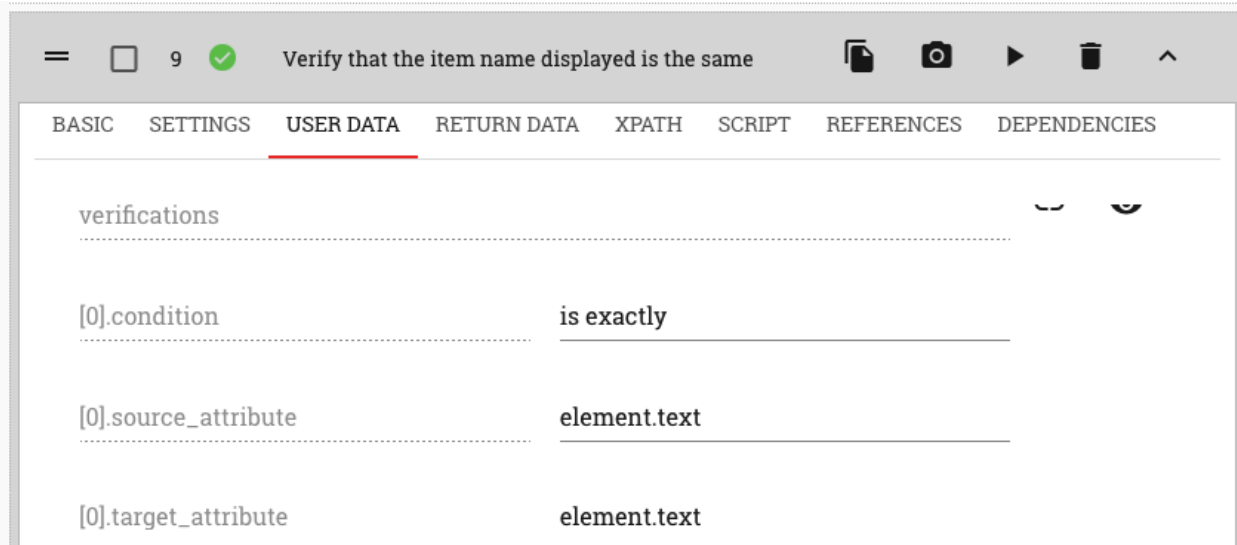
In this example, we will select the Source 'text' attribute and the Target 'text' attribute. We will also select the condition 'is exactly'.



Now click on the 'Save' button. A verification test step is recorded.

Details of the verification condition can be seen in the 'User Data' section of this test step

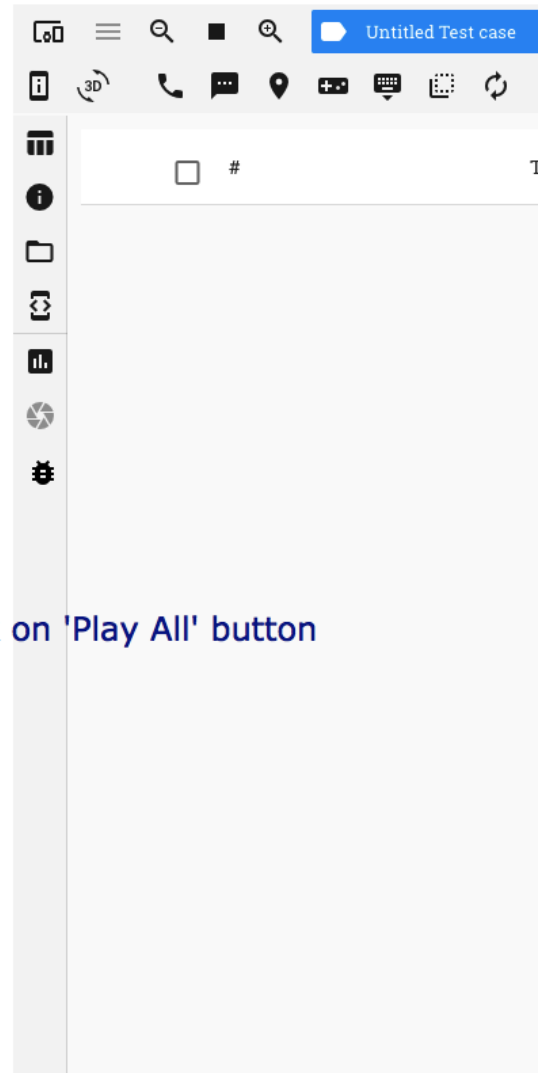
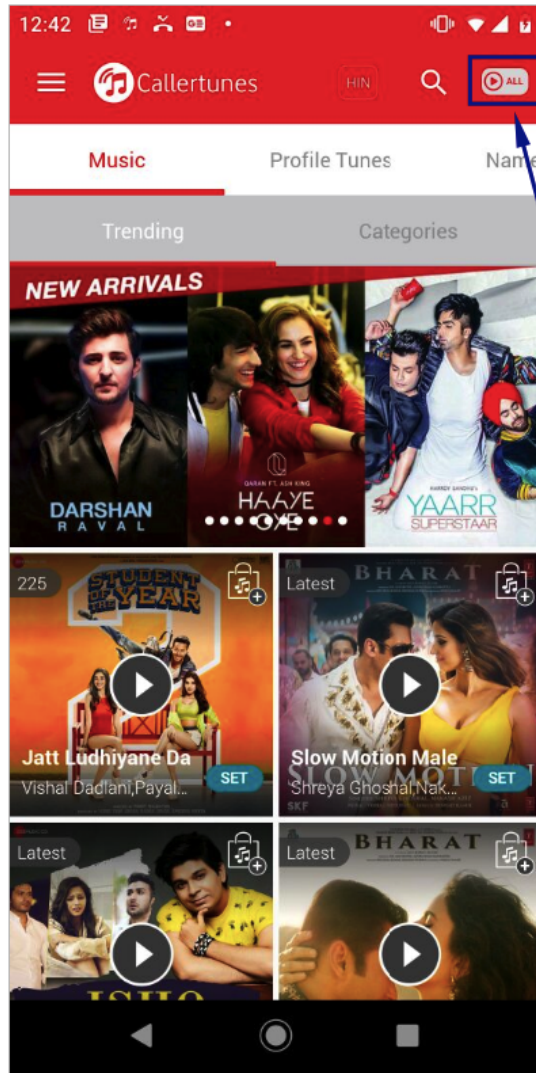




12.1.3 Verify that an element exists

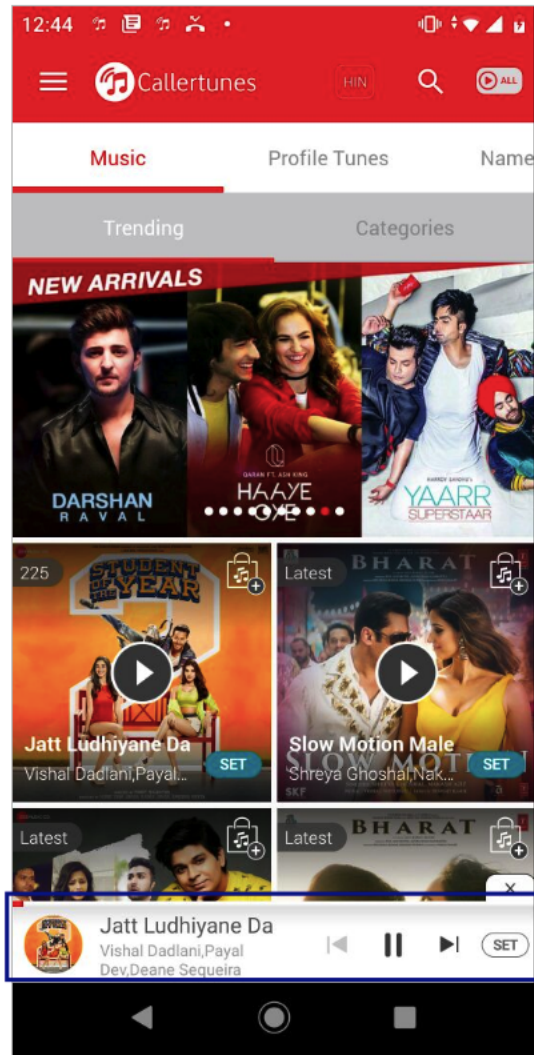
Sometimes, you may want to just verify whether a specific element is displayed on the app page or not. This does not require comparison of attribute values for verification.

E.g. say, in an app where music plays, you click on a button and a music player opens. You need to verify that the music player has opened on the page.



Click on 'Play All' button

In this case, all that you need to check is whether the element containing the music player is now visible on the page.

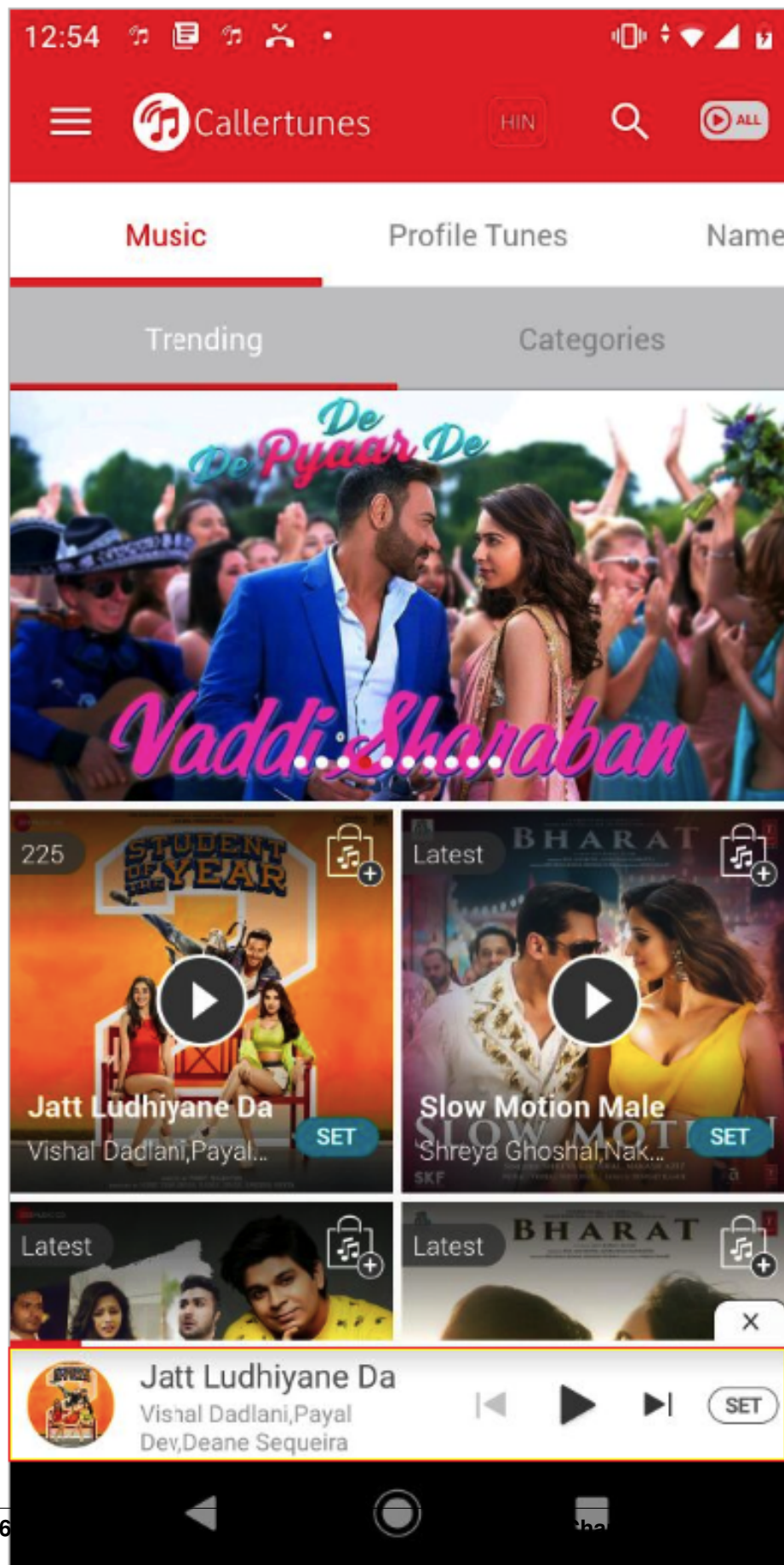


The music player opens

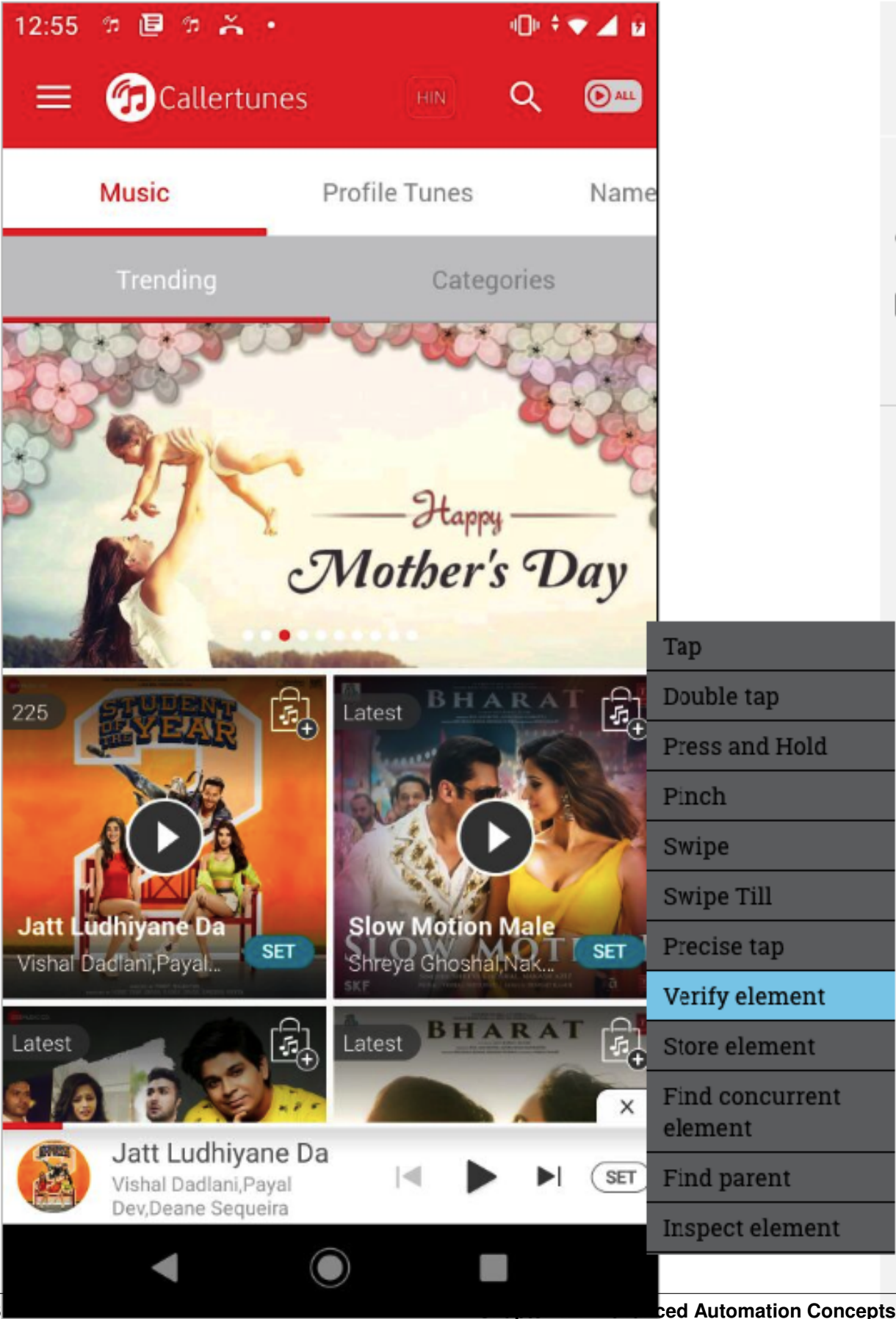
We need to verify that this player (its element, actually) exists on the page

RobusTest enables you to perform this kind of a verification as follows:

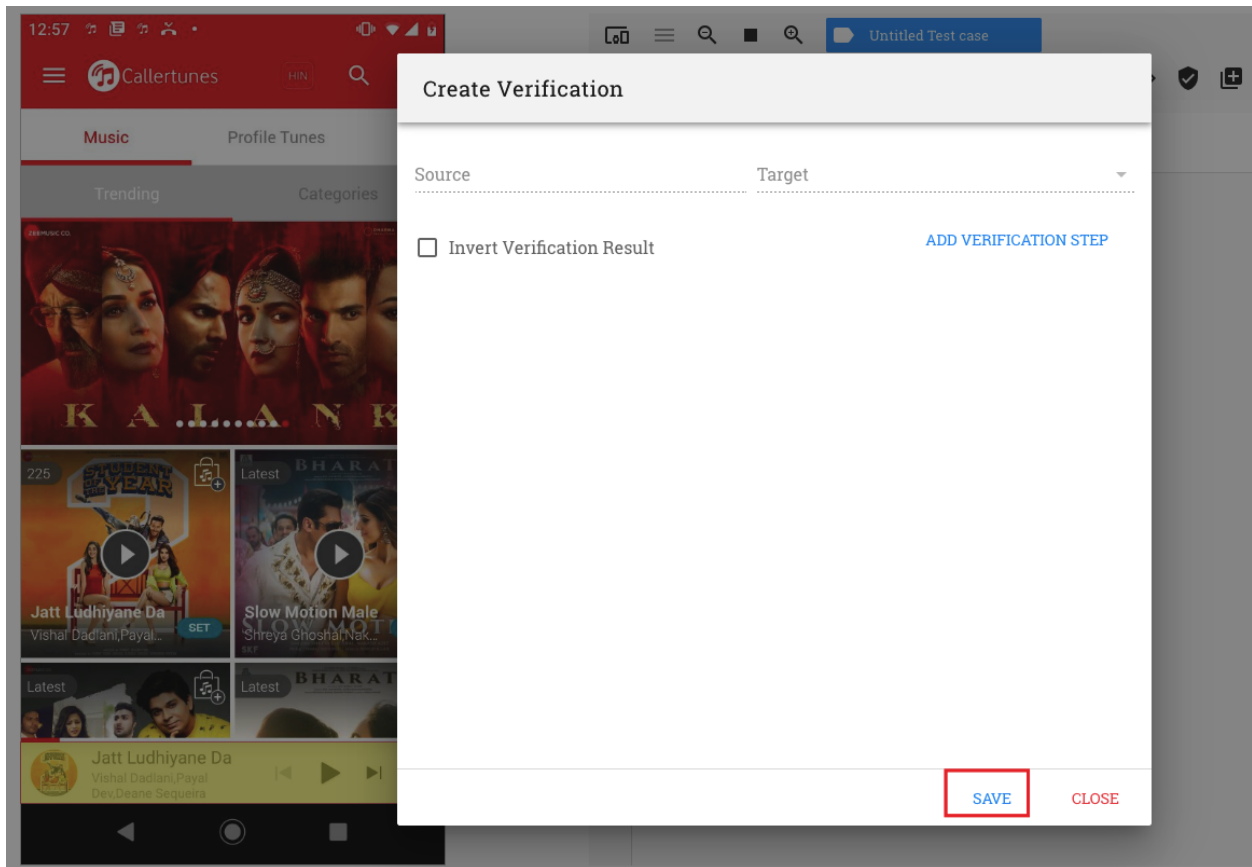
1. In an Automation test session, hover the mouse on the device screen over the element which has to be verified.



2. Left click on the element and click on 'Verify Element'



3. On the “Create Verification” window that opens, click on the ‘Save’ button



The verification test step that is created verifies that the element exists

12.1.4 Invert Verification

Another interesting feature available on RobusTest is to invert the results of a verification action.

Usually, if a condition mentioned in a verification test step is a match, the test step passes on execution.

If ‘Invert Verification’ is enabled for the same test step, the result of the comparison is inverted, i.e., the test step will fail in execution

This option can be of help in many use cases.

E.g.1: Let us say that a music player is visible on an app (see example from *Verify that an element exists* section)

Our goal is to ensure that the music player is closed (i.e. no longer visible on the page) after the ‘Close’ button is clicked on.

We can do this using the ‘Invert Verification’ option as follows:

1. Open the music player
2. Left-click on the music player element and click on ‘Verify Element’
Note: If the ‘Save’ button is clicked at this point, the verification test step recorded checks if this specific element is available on the page or not
3. Enable ‘Invert Verification’ by clicking on the check box
4. Click on the ‘Save’ button

RobusTest provides you a means to compare attribute values of an element with:-

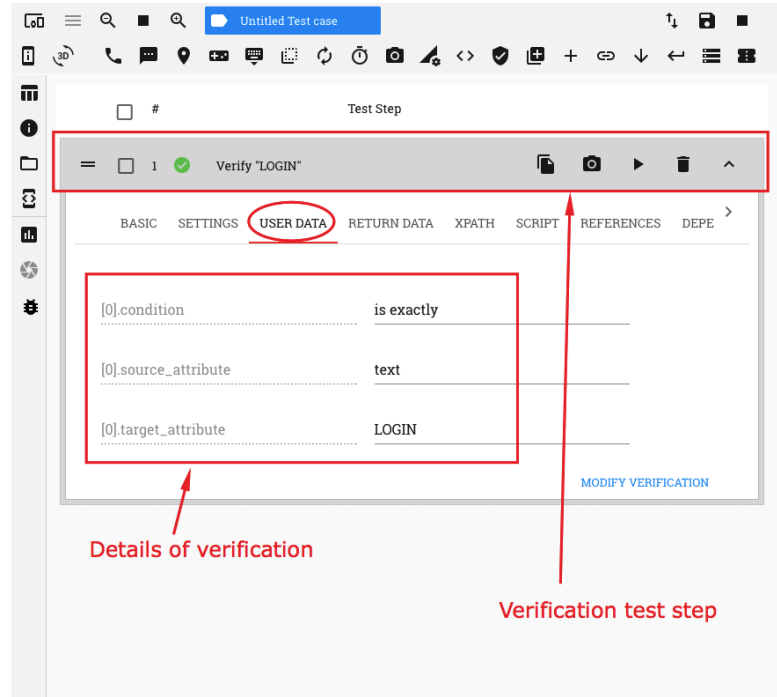
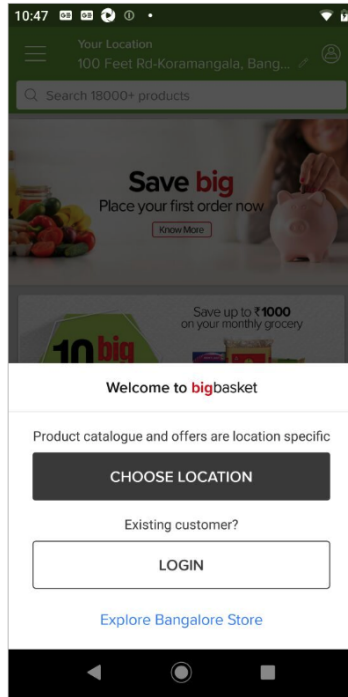
- a custom value ; Or;
- that of the attribute value of another element

We refer to this process as a **verification**

A verification is done by comparing a '**Source**' attribute value with a '**Target**' attribute value.

When a verification is recorded, a corresponding test step is created.

Details of the verification can be seen in the 'User Data' section of the verification test step that is recorded.



On RobusTest, a verification can be done in two ways:-

1. *Using the 'Verify Element' option*
2. *Using the 'Verification' button option*

The 'Verification' feature on RobusTest provides you a powerful way to perform a many more operations such as:-

1. *Verify that an element exists*
2. *Invert Verification*
3. *image-based-verification*

12.2 Functions

One of the most powerful features that the RobusTest platform provides is the use of '*Functions*' in automating test cases.

A function, in RobusTest, is a block of test steps that perform a particular task. E.g., a group of test steps that capture customer information.

Why use functions

Functions help:-

- in reducing the time & effort spent in automating test cases through re-usability of code
- in easy maintenance and updating of test cases in case of any changes in the application flow
- with improved readability of test cases

Use functions:-

- wherever a series of steps form a logical unit, e.g. a series of test steps for selecting a rental plan, entering billing information, etc.
- whenever a group of actions are repeated across more than one scenario

Using functions will significantly reduce the amount of future time spent in incorporating changes to app flows, data, element resource ids, etc.

E.g. Say, there are a series of standalone test steps (i.e., test steps not within a function) where a 4G Plan is being selected and that these steps are repeated in a number of test cases. In future, if the specific plan being selected changes, you would have to go to each and every test case and update them.











Instead, have these steps within a function named 'Select Rental Plan'. Now, any change in the way a rental plan is created would only require a one time change within this function script. This change is reflected in all test cases that call this function.

Let's have a look at how we create and use functions in RobusTest

1. Creating a Function

RobusTest enables you to create functions out of existing test cases.

- Open an automation test session
- Record the test steps that you would like to be converted into a function
- Save the test case
- Go to the 'Test Cases' page
- For the test case that you created, click on the 'Create Function' button

Serial	Test Case Name	Created By	Updated By	Last Edited	Actions
1	Search for wheat	Suraj Nair		11/07/2019, 15:30:18	    
2	Test function	Suraj Nair		10/07/2019, 13:10:04	    

f. On the pop-up window that opens:

- provide an appropriate name for the function
- choose the type of function to be created from the 'Category' drop down



Add Function to testscript

Function Name

Category
None

SAVE CLOSE

g. Click on the 'Save' button

You have now created a new function

2. Using a Function

Now that you have created a function, let's see how we can use them

1. In an Automation test session, create a new test case or open an existing test case in which you would like to use the function.
2. Click on the 'Import Function' button.
3. On the 'Import Function' window that opens, click on the function you would like to use in the test case.
4. Click on the 'Save' button.
5. Save/Update your test case.

The function you created earlier is now a part of the test case.

It is seen as a separate test step in the test case.

On expanding this test step, you can see a tab named 'Function Steps'

On clicking on the above tab, you can view the following information:

- name of the test case that was converted into a function
- name of the function
- list of test steps that constitute the function

If any test step being executed within a function fails, then, the test case that calls the function also fails

[Click here to know more-on-functions](#)

12.3 Parameterisation of Data

12.3.1 Mapping to Step

12.3.2 Mapping to Data Set

To be able to provide your test cases with non-default data you need to do the following

1. Create a function

The step or set of steps that you wish to parameterize should be inside a function and your test cases should use the function. e.g. in case you are trying to parameterize the login steps, it would be a good idea to put the steps to enter username, password and submit the data inside a function. This also creates a logical entity which takes care of login process.

2. Parameterize the data

- Open the function test case for editing.
- Open the User Data tab for the test step that you are trying to parameterize.
- Click on the “Enable Parameterization” button for the field that you wish to parameterize.
- Provide a userfriendly variable name that identifies that field. Make sure there are no spaces in the name.
- Do the same for all the test steps that you wish to parameterize
- Save the function test case

3. Create a Data Set and note down the Data Set ID. (To know more about data sets, read data-sets)

4. Create a run specifying the data set to be used and how to use the data set‘1

HTTP Method: POST

URL: <http://<RobusTest URL>/v3/run/new?accesskey=<access key>>

Payload:

```
{
  "testsuite": "<Test Suite ID>",
  "project": "<Project ID>",
  "build": "<Build ID>",
  "devices": [
    "<comma separated device IDs>"
  ],
  "settings": {
    "appium": {
      "automationName": "UiAutomator2",
      "disableAndroidWatchers": "true",
      "forceEspressoRebuild": "true",
      "fullReset": "true",
      "ignoreUnimportantViews": "true",
      "noReset": "false",
      "noSign": "true"
    },
    "general": {
      "checkElementIsVisible": "yes",
      "collectLog": "yes",
      "collectPerformance": "yes",
      "elementWaitTimeOut": "30",

```

(continues on next page)

(continued from previous page)

```

    "enterTextMethod": "appium",
    "handleAndroidPermissionPopup": "allowAll",
    "pagesourceTimeout": 100,
    "recordingMode": "normal",
    "retryFailedTests": 0,
    "runOnlatestbuild": "true",
    "streamPagesource": "yes"
  },
  "notification": {}
},
"setting": "",
"datasetID": "<Data Set ID>",
"datasetMode": "<valid value is strict or blank>"
}

```

If you set the datasetMode to strict, while running the tests, the dataset will be run only on the corresponding devices. In case the datasetMode is not set to strict, then the system randomly assigns the dataset to the devices on a first come first serve basis.

Sample

HTTP Method: POST

URL: <http://devicelab.acme.com/v3/run/new?accesskey=d2342dsdad231313>

Payload:

```

{
  "testsuite": "5e0d18075752875f4d723e01",
  "project": "5d6f3d1f57528725c1afa13b",
  "build": "5df1e691575287692822d4d9",
  "devices": [
    "5d6f3ef4c74f741abb97e23c", "5ada4c74f741abb97e23c"
  ],
  "settings": {
    "appium": {
      "automationName": "UiAutomator2",
      "disableAndroidWatchers": "true",
      "forceEspressoRebuild": "true",
      "fullReset": "true",
      "ignoreUnimportantViews": "true",
      "noReset": "false",
      "noSign": "true"
    },
    "general": {
      "checkElementIsVisible": "yes",
      "collectLog": "yes",
      "collectPerformance": "yes",
      "elementWaitTimeOut": "30",
      "enterTextMethod": "appium",
      "handleAndroidPermissionPopup": "allowAll",
      "pagesourceTimeout": 100,
      "recordingMode": "normal",
      "retryFailedTests": 0,
      "runOnlatestbuild": "true",
      "streamPagesource": "yes"
    },
    "notification": {}
  }
}

```

(continues on next page)

(continued from previous page)

```
},  
"setting": "",  
"datasetID": "5e135c765752875a2a64d33a",  
"datasetMode": "strict"  
}
```

Let's have a look at a few advanced concepts being used in automating test scripts:

1. *Verification*
2. *Functions*
3. *Parameterisation of Data*

Scheduling your tests

You can use RobusTest to schedule your automation tests. To schedule your tests you should follow the steps below

1. Create a test suite - Create a test suite by adding all the test cases that you wish to be part of the test run
2. Create a test run - Run the test suite created above by selecting the devices that you wish to run it on
3. Get the “Run ID” - For the run that was just created, a unique Run ID is generated. To get the Run ID, click on the Reports icon for the run. In the Reports page, copy the last part of the URL. e.g. if the Reports link is

<http://mobile.robustest.com/#/project/57d0f2e4aca33b21f5724cd7/report/58498732aca33b11ca655660>

the Run ID is 58498732aca33b11ca655660

4. Customize the scheduling API - Scheduling API looks like <http://<RobusTest URL>/api/1/run/<Run ID>?updateFromTestSuite=true&build=latest>

e.g if you Run ID is 58498732aca33b11ca655660 and your RobusTest URL is <http://192.168.1.1:8081>, then your scheduling API will look like

<http://192.168.1.1:8081/api/1/run/58498732aca33b11ca655660?updateFromTestSuite=true&build=latest>

Every time above API is called, the test suite will be run on the devices selected on the latest build of the project.

5. Schedule run API - Once you have customized the run API, you need to add it as a cron job on your test run server.

14.1 Appium Hub

With the refreshed version of RobusTest Hub for Appium, you get a cleaner way of running your Appium tests on mobile apps as well as mobile browsers.

14.1.1 Run Appium Tests

To run your Appium tests using RobusTest, follow the steps below:

1. Appium server URL

- Use the RobusTest URL as the Appium server URL.
- This will be of the form **http(s)://[RobusTest Device Lab URL]/wd/hub**

2. Desired Capabilities

All RobusTest specific desired capabilities should be provided by appending ‘robustest:’ to the desired capability name e.g. **robustest:accessKey**. The default attributes provided by Appium should be used as it is.

app - In case you are running your tests on a mobile app, you also need to provide the *app* desired capability. This can be a local file or a remote file.

platformName - Depending on whether you wish to run your tests on Android or iOS, please select the appropriate platform name, provided using the *platformName* desired capability.

robustest:deviceID - Provide the device details using the **robustest.deviceID** desired capability. *This is not a mandatory field. If you provide deviceID for your tests, then the system will try to allocate the specific device requested and fail if the device is not available for use*

robustest:projectID - The RobusTest project under which you wish to run your tests should be provided using the *robustest.projectID* desired capability.

robustest:accessKey - The user is authenticated using the RobusTest Access Key, provided using the *robustest.accessKey* desired capability.

robustest:buildID - If you are running your tests on a build uploaded to RobusTest and want to see the build details in your report, pass the *robustest:buildID* desired capability. The buildID is the unique identifier for a build that is uploaded to RobusTest.

robustest:jobIdentifier - To create a job or add a test case to a job, you need to direct the system to do so. This can be done by using the desired capability mentioned here. The value that you set for the job would be such that you are able to differentiate one instance of a job from another.

robustest:testcaseName - To create a test case from an appium test session, you need to direct the system to do so. This can be done in two ways. Either by giving the desired capability as mentioned here or by making an API based request to the device lab. Details for the API are available at http://api.robustest.com/#tag/appium/paths/~1v3~1appium~1{appium_session_id}~1testcase/post

browserName - In case you are running your tests on a mobile browser, provide the *browserName* desired capability.

adbExecTimeout - In case you are running your tests on a mobile browser, it is highly recommended to add the *adbExecTimeout* desired capability and give it a high value of say 2000000. This ensures that tests do not error due to timeout.

3. Setting Timeout Values

- **robustest:runSetting** - User can create a Run Setting and provide the Run Setting ID as the value of the *robustest.runSetting* desired capability to configure various aspects of the Appium job. The attributes currently supported are:
 - *runTimeout* - this value (in secs) can be used to specify the max amount of time a job can run before it is closed by the system on account of exceeding the value set for runTimeout.
 - *idleTimeout* - this value (in secs) can be used to specify the amount of time a job can be idle before the job can be closed by the system
 - *testcaseTimeout* - this value (in secs) can be used to specify the maximum run time of a test case. This value will be used by the system only when Advanced Integration with RobusTest Appium Hub is done.

4. Additional custom desired capabilities

- **robustest:fallbackScreenshotMode** - this desired capability is specific to iOS test sessions. To ensure a higher success rate of screenshots, user can use the *robustest.fallbackScreenshotMode* desired capability. Valid values are i. idevicescreenshot ii. appium. If this desired capability is not used, fallback method will not be employed to take a screenshot in case the default screenshot method does not work.

14.1.2 Organize Appium Sessions Into Test Cases

1. Grouping multiple Appium sessions into a single job

When you run an Appium test job, the job may comprise many Appium sessions. For easier reporting and management, it is possible to group all the Appium sessions created as part of a single job.

To do so, use the same value for the **robustest:JobIdentifier** desired capability for Appium sessions belonging to the same job.

All appium sessions with the same value for *robustest:JobIdentifier*, will be grouped together.

2. Naming your Appium sessions

Many automation frameworks are designed in such a way that they create a new Appium session for every test case.

To be able to read such reports easily in RobusTest, user can use the **robustest:SessionIdentifier**. This desired capability is meant to be unique for every Appium session within a job.

3. Retrieving RobusTest Test Session ID

- Once your Appium tests starts, you can access details about your Appium session using the unique RobusTest Session ID. This ID is created when user tries to start an Appium session on RobusTest.
- RobusTest Session ID can be accessed in two ways:

a. By using robusttestSessionIdentifier

To retrieve the RobusTest session ID using the **robusttestSessionIdentifier**, use the following API:

GET /v3/hub/sessionIdentifier/{robusttestSessionIdentifier}

The advantage of using the **robusttestSessionIdentifier** to retrieve the RobusTest session ID is that even if the Appium session does not get created, the RobusTest session ID will help in accessing the appium log and other details.

As mentioned earlier, the **robusttestSessionIdentifier** will have to be passed as a desired capability.

b. By using the Appium Session ID:

To retrieve the RobusTest session ID using the Appium session ID use the following API:

GET /v3/hub/session/{Appium Session ID}

When using this method, you can get the RobusTest Session ID only when you have a valid Appium session ID.

14.1.3 Appium Test Data

Retrieving Appium Test Session Data

When running your Appium tests, data is generated in the form of logs and test details. Additionally, the user may want to annotate test sessions with attributes and their values.

Data for an Appium test session can be retrieved at the end of every session or at the end of the complete job. User can choose their approach based on their need.

Test run data can be retrieved from RobusTest using two different ways:

1. By using a RobusTest Test Case (RECOMMENDED)

- To enable seamless reporting in RobusTest, it is recommended to create a test case in RobusTest for every Appium test session that is completed.
- To achieve this, the appium test creation API needs to be called after an Appium test session is complete.
- Details of the API can be found by clicking [here](#)
- By providing the Appium session ID, user can assign name, status, desc and message for a test case. If status is not provided in the API call, then the test case is marked as fail.
- Fetch the data right after the completion of the appium test session.
- The response to this API will provide the various details of the test case created in RobusTest, including:

1. Start time

2. Updated time

3. Duration - calculated from the difference between the value of updated time of the appium test session at the time of the API being called and the start time of the appium test session.

4. By using an Appium Session ID - available inside the *framework* object of the response.

- If the test case has run properly and has been ended properly in the test code, the *Updated time* is the completed time of the test session.

- Note: *It is important that the above API is called only after the test session is complete to ensure that the values stored are correct*
- Fetch data after the completion of the appium job
 - Get the RobusTest Job ID by invoking the following API and extracting the value of the “_id” attribute:
GET /v3/job/{robustest job identifier}
 - The list of all test case entries for the job can be retrieved using the following API:
GET v3/job/{RobusTest Job ID}/testcases

2. Appium Test Session

In case, user does not wish to create a test case entry in RobusTest, data can still be retrieved by the following ways:

1. Fetch data right after the completion of the appium test session

- Use any of the following APIs:
GET /v3/hub/sessionIdentifier/{robustestSessionIdentifier}
GET /v3/hub/session/{Appium Session ID}

2. Fetch data after the completion of the appium job

- Get the RobusTest Job ID by invoking the following API and extracting the value of the “_id” attribute:
 - **GET /v3/job/{robustest job identifier}**
- Using the Job ID, get all the test sessions for the job:
 - **GET /v3/job/{RobusTest Job ID}/testsessions**
- In the response of the API request, there will be an array of objects - each object being a test session entry. Each entry will contain all the details related to the session e.g. robustestSessionIdentifier and Duration

14.1.4 Appium Hub to Find Locators

When developing your Appium tests, you may want to connect to a device to check the page source and create your element locators.

You can easily create and connect to an Appium Hub session to inspect your app's screens.

Follow the following steps:

1. Start the Appium app
2. Go to Appium -> New Session Window
3. Go to Custom Server Tab
4. Enter the values for:
 - **Remote Host** - this will be the RobusTest Hub URL
 - **Remote Port** - this will be the port for the RobusTest Hub URL
 - **Remote Path** - this will “/wd/hub” which is the default value
5. Set the following desired capabilities:
 - **platformName** - this will be either Android or iOS
 - **accessKey** - this is your RobusTest Access Key

- **projectID** - this is your RobusTest Project identifier
- **app** - this is the build URL generated by RobusTest. Remember to provide the “accessKey” parameter at the end of the URL to authenticate the request
- **deviceID** - this is the unique identifier generated by RobusTest for the device

6. Click on **Start Session**

14.1.5 Running tests for your Unity based app on the RobusTest Hub

Running your Unity test cases on the RobusTest Hub

For more information about Unity and Appium, read <https://altom.gitlab.io/altunity/altunitytester/pages/tester-with-appium.html#>

One of the most important aspects of running your AltUnity based tests on a device lab is to ensure that the port on which AltUnity server runs on the device (default port number is 13000), is forwarded to a port on the host machine.

For this, you need to use the desired capability

```
robustest.forwardDevicePort
```

The value that you need to set for this capability is 13000 which is the default port on which the AltUnity server runs on the mobile device. Once the Appium session starts to get details of the host machine and port, get the device details using the device API

<http://api.robustest.com/#tag/device/paths/~1v3~1device~1{deviceID}/get>

For more information regarding the advanced usage of AltUnity refer to the following link

<https://altom.gitlab.io/altunity/altunitytester/pages/advanced-usage.html#>

14.2 Run Espresso Tests

Introduction to Espresso

Espresso is a testing framework for Android that makes it easy to write reliable UI tests for an app. More information at <https://developer.android.com/training/testing/espresso/>

How do Espresso jobs run in RobusTest

To run an Espresso job on RobusTest, an API based request should be made to RobusTest. In this request, user can specify the priority of the job (a higher number means higher priority), minimum number of devices required to run the job.

Once espresso job is submitted, the job enters the Pending state.

The RobusTest system checks for jobs in pending state every 100 seconds and picks up a job based on its priority and the number of devices required. If available device count is less than min devices required by job then job will not run in current cycle.

- Once a job is picked up to run, it is added in the queue.
- Once in the queue, the number of devices required are blocked.
- **Each device that is reserved for the job is put through a dry run. In dry run both APKs - the app binary and test binary -**
 - If the dry run fails on a device, that device is removed from the job.
 - The job will run even if one device of all the reserved devices passed the dry run.

- If all the devices fail to execute the dry run, then the job will be marked as Failed and will be retried in next cycle. The number of such re-attempts can be configured in the job API request by changing the value of field maxAttempts. Default value of maxAttempts is 5.
- Once the dry run is successful even on one of the reserved devices, the associated PR will be blocked as pending with the context set as robustest/espresso and description as Espresso tests are running on RobusTest for job.
- Once the dry run is completed, the job moves to the Running state, where the job is now ready to give test cases to devices which have completed their dry runs successfully.
- Once the job starts running, the devices start executing tests. Once a test case completes execution on a device, the device requests for the next test case.

During the course of test case execution, one or more of the following may happen:

- If a test case executes successfully, the device requests for the next test case.
- When a device requests for a test and if there are tests remaining, one test case is assigned to the device. If there are no more tests to assign, the device is freed.
- If a test case fails, then it is marked as failed and added back to the pool of tests to be retried. The number of such of retries is configurable. When a failed test case is retried, it may run on the same device or on a different device.
- If a test case crashes during execution, then it is not retried.
- If no response is received from a job for 800 seconds, the job is restarted i.e. the job is suspended, all associated devices are freed and the job is resumed. When the job is resumed, the devices that reserved based on availability. Therefore, the set of devices may differ from the earlier set. In such a case, the tests that are already executed are excluded from the run and only the pending tests are executed.
- Once a job completes, it checks to see if there are any devices that are still associated with the job. If such a device is found, it is freed.
- Once a job completes, a comment is added to the associated GitHub commit with the following data:-
 - Total number of tests
 - Number of tests passed
 - Number of tests failed
 - Number of tests errored
 - Number of tests skipped
 - Number of tests crashed
 - Link to Report

The PR is updated based on the following conditions:

- Even if one test case crashes, the PR will be marked as Failed
- If the number of failed test cases is equal to or higher than the threshold set in the Job API request, the PR will be marked as Failed
- If the number of failed tests cases is lower than the threshold set in the Job API request, the PR will be marked as Successful
- If the job does not complete within the maxAttempts, the PR will be marked as Failed.

14.3 Run XCUITest Tests

1. Identify the project which you shall use for running your tests. If you do not already have such a project, then you need to first create a project. Once identified, note down the Project ID for the project.

- This needs to be done only once.
- *Project ID* is the alphanumeric number in the URL of the project dashboard.

2. Upload the app enterprise build to your project and note down the build ID.

- This needs to be done everytime you have a new app build.
- *Build ID* is the name of the file that you get when you copy the URL to the build.

Refer to the Upload Build - Remote section in the following link for build upload API

<http://docs.robustest.com/en/latest/projectdashboard.html>

3. Upload the zipped up xctest file to your project and note down the build ID.

- This needs to be done everytime along with a new app enterprise build.
- Build ID is the name of the file that you get when you copy the URL to the build. Let's refer to this as the *Test Build ID*.

Refer to the Upload Build - Remote section in the following link for build upload API

<http://docs.robustest.com/en/latest/projectdashboard.html>

4. Get your ACCESS KEY from your profile page in RobusTest.

Refer to the following link to get help with getting your Access Key

<https://robustest-docs.readthedocs.io/en/latest/userprofile.html>

5. Once you have the three pieces of information above, invoke the test using the following API details:

- **URL:** <RobusTest URL>/v3/xcuitest/job/new?accesskey=<ACCESS KEY>
- **METHOD:** POST
- **PAYLOAD:**

```
{
  "identifier": "<IDENTIFIER_NAME_GIVEN_BY_ENTERPRISE>",
  "deviceVersion": "<iOS version of the device>",
  "project": "<Project ID>",
  "build": "<Build ID>",
  "testBuild": "<Test Build ID>"
}
```


6 When you run your job using the API above, the response will contain the job id in form of the key `_id`. Use this value to get the JSON for the run report

<http://<RobusTest URL>/v3/report/<job id>/xcuitest>

7 To get the JSON report for an individual test case, use the id value for each test case run instance in the following URL

<http://<RobusTest URL>/v3/xcuitest/testcase/<test case instance ID>>

14.4 Run Selenium Tests

Build Url	: http://10.12.25.162:8082/build/566170ce1466f34d97af5bd3.apk	
Description	: Provide description	



Manual



Record



Test Cases



Test Suites



Run



Reports

1. create a web project
2. Hub url - point to nerve server
3. Desired capabilities will look as below

```
{  
  "browserName": "[Browser Name]",  
  "accessKey" : "[User Access Key]",  
  "projectID": "[Project ID]"  
}
```

4. Get browser status

HTTP Method: GET

URL: [http://{\[DEVICE LAB URL\]}/grid/browsers](http://{[DEVICE LAB URL]}/grid/browsers)

RobusTest provides you an efficient and simple way to run your existing test cases on our Hub.

You can view the execution of your test cases real-time using our Live View option.

At the end of the run, detailed run reports are also generated

1. *Appium Hub*
2. *hub-espresso*
3. *Run XCUITest Tests*
4. *Run Selenium Tests*
5. *hub-api*

On the top right corner of the RobusTest page, you see your RobusTest username being displayed.

On clicking on the username, a menu with the following options is displayed:

1. Profile
2. Help
3. Support
4. Logout

1. Profile

Clicking on 'Profile' opens the 'User Selection Profile' window.

On this window, the following information are displayed:

- a. *Name*: The username that was used at the time of creation of the profile is displayed here. This field is editable.
- b. *Email*: The email provided at the time of creation of the profile is displayed here. This value is not editable.
- c. *Access Key*: RobusTest provides a number of APIs that you can use to perform a wide variety of actions. For security reasons, only authorized users are permitted to execute these APIs.

For this, RobusTest provides each user an access key. This access key is a unique identifier associated with the user's profile on RobusTest. You will need this access key to execute RobusTest APIs.

- *Copy Access Key*: This button copies the access key to the clipboard
- *Hide/Show Access Key*: This button hides or unhides the Access Key that is displayed on the profile
- *Reset Access Key*: Clicking on this button resets the Access Key. A new Access Key will now be generated for the user

2. Help

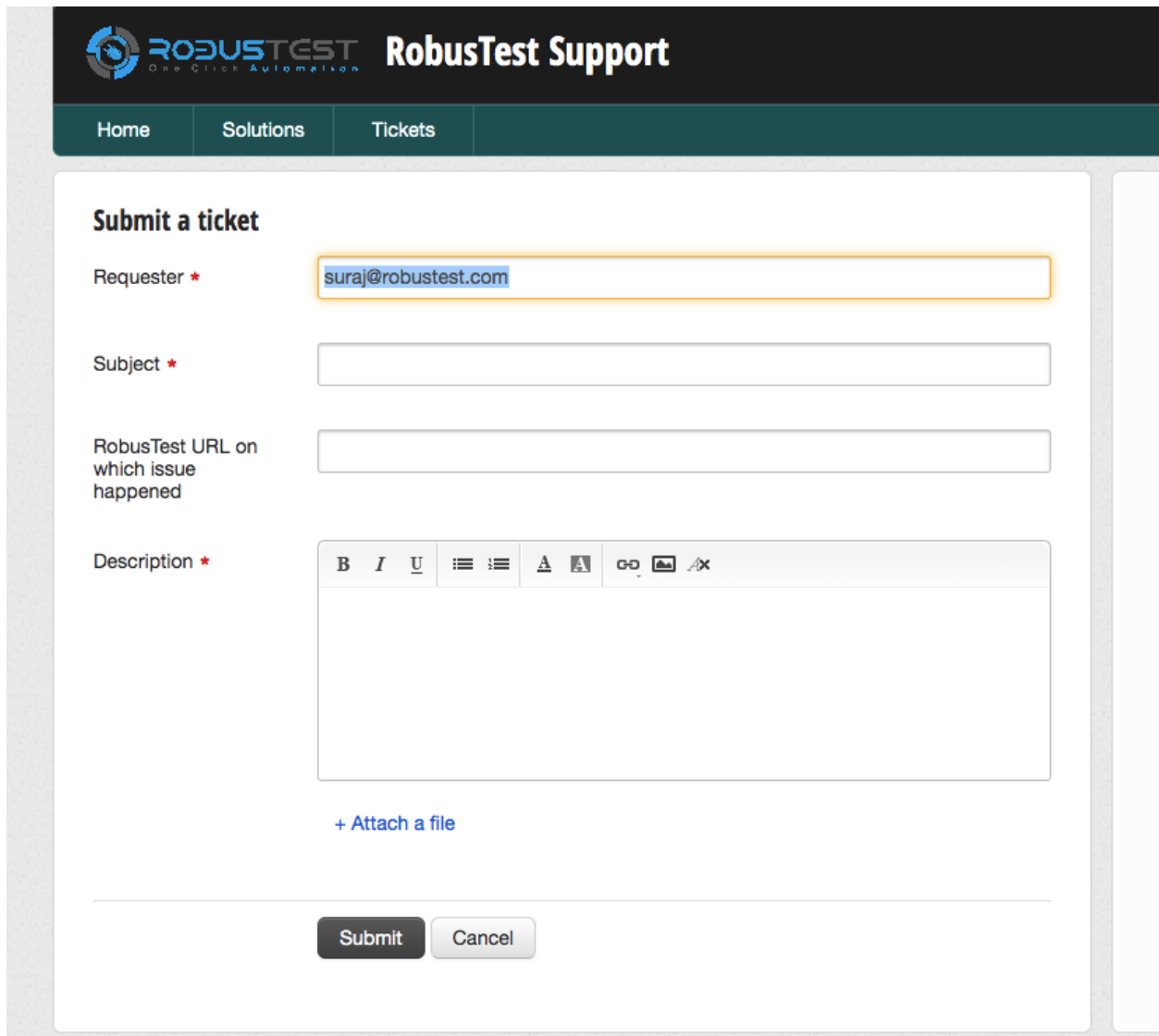
Clicking on 'Help' opens the RobusTest Product Documentation page on a new browser tab. The documentation will help you in understanding in detail how the RobusTest platform works.

3. Support

At RobusTest, we aim to provide you, the users of our platform, technical support of the highest quality.

When you have a query, you can reach out to us using the 'Support' option.

On clicking on 'Support', you will be re-directed to the third-party ticketing platform that we employ to handle all support requests.



The screenshot shows the 'RobusTest Support' page. At the top, there is a navigation bar with 'Home', 'Solutions', and 'Tickets' links. The main heading is 'Submit a ticket'. The form includes the following fields:

- Requester ***: A text input field containing the email address 'suraj@robustest.com'.
- Subject ***: An empty text input field.
- RobusTest URL on which issue happened**: An empty text input field.
- Description ***: A rich text editor with a toolbar containing icons for bold (B), italic (I), underline (U), bulleted list, numbered list, link, unlink, insert image, and delete. Below the toolbar is a large text area for the description.

Below the description field, there is a link that says '+ Attach a file'. At the bottom of the form, there are two buttons: 'Submit' and 'Cancel'.

On this page, you can provide an appropriate subject line and a detailed description of the issue you are facing or a requirement you would like to be met by the platform. You can also attach relevant screenshots or log files where necessary.

Once a support ticket is logged, the RobusTest Support team will respond to your email based on agreed upon service guidelines.

4. Logout

You can log out of the RobusTest platform by clicking on the '*Logout*' option

The Health page provides you a quick look at the health of your device cloud. It provides you information on the connection/disconnection status of nodes (servers) and devices on the cloud.

You can access the Health page by clicking on the icon for the same on the main RobusTest navigation header.

On the Health page, you can see the following 2 tabs:

1. Devices
2. Nodes

1. Devices

The '*Devices*' tab is open by default. It provides you a list of devices connected to the RobusTest platform as well as their connection/disconnection status.

You can search for details pertaining to a specific device using the '*Search Devices*' text box.

On this tab you see the following 3 sub-tabs:

a. All Devices

This tab provides the connection status of all devices on the RobusTest platform. This list includes both - connected and disconnected devices.

The following information is provided for each device:

1. *Device Name*: The name of the device, along with the Android version on which it is being run, is displayed. E.g. Redmi 6A - v8.1.0
2. *Node Name*: The name of the node or server on which the device is connected is displayed here. The node name is not displayed for disconnected devices.
3. *Device ID*: The device ID is displayed in this column
4. *Status*: This column displays the values '*Connected*' or '*Disconnected*' depending on the status of the device.
5. *Connected Since*: For devices in '*Connected*' status, this column displays the time period for which the device has remained connected.

For devices in '*Disconnected*' status, this column displays the time period for which the device has remained disconnected.

b. Connected

This tab acts as a filter on the '*All Devices*' tab to display device related information only for those devices in '*Connected*' status. The total count of connected devices is displayed next to the tab header.

c. Disconnected

This tab acts as a filter on the '*All Devices*' tab to display device related information only for those devices in '*Disconnected*' status. The total count of disconnected devices is displayed next to the tab header.

2. Nodes

The '*Nodes*' tab provides you:

- a list of nodes (i.e., RobusTest servers) that constitute the RobusTest device cloud;
- the connection/disconnection status of each individual node

You can search for details pertaining to a specific node using the '*Search Nodes*' text box.

On this tab you see the following 3 sub-tabs:

a. All Nodes

This tab provides the connection status of all nodes on the RobusTest platform. This list includes both - connected and disconnected nodes.

The following information is provided for each node:

1. *Node Name*: The name of each node is displayed in this column. If no name has been provided to the node, then the IP of that node server is displayed.
2. *Node IP*: The IP of each node or server is displayed here. The node IP is not displayed for disconnected nodes.
3. *Status*: This column displays the values '*Connected*' or '*Disconnected*' depending on the status of the node.
4. *Connected/Disconnected Since*: For nodes in '*Connected*' status, this column displays the time period for which the node has remained connected.

For nodes in '*Disconnected*' status, this column displays the time period for which the node has remained disconnected.

5. *Devices*: For each node, this column displayed the count of connected and disconnected devices on that node.

b. Connected Nodes

This tab acts as a filter on the '*All Nodes*' tab to display node related information only for those devices in '*Connected*' status. The total count of connected nodes is displayed next to the tab header.

c. Disconnected Nodes

This tab acts as a filter on the '*All Nodes*' tab to display node related information only for those nodes in '*Disconnected*' status. The total count of disconnected nodes is displayed next to the tab header.

CHAPTER 17

Admin Console

Users who have Admin access on the RobusTest platform will find an extra icon on the platform header to access the same



The Admin Console enables you:-

- a. to perform administrative tasks on RobusTest.
- b. to monitor and capture important and useful info on usage of the platform

Let's have a look at each section of the Admin Console:

1. Activity

This section captures all user activity with respect to device groups. These include:

- creation of a device group
- modification of a device group * addition/removal of devices in a group * addition/removal of projects in a group
- deletion of a device group

2. Project

This section provides details of all active and inactive projects on RobusTest

On selecting a project by clicking on it, the following additional information pertaining to that project are available:

- a. *Sessions* - The last 100 test sessions that were opened on the project are visible. For each test session the following info are displayed:
 - *Type of test session* (Manual, Automation, Run, Hub)
 - *Device used in the test session*
 - *Name of user who created the test session*
 - *Start and End time of test session*

- *Duration of the test session*
- *Reason for termination of test session* (Appropriate messages are displayed for normal and abnormal termination of a test session)
- b. *Usage* - This tab provides the breakup of the total duration of each type of test session (Manual, Automation, Run & Hub) that was created in this project
- c. *Builds* - This tab displays details of each build that was added to the project
- d. *Members* - This tab provides:
 - details of all members within a project
 - privileges available for each member
 - a means to add/remove members
 - a means to grant/revoke admin rights
- e. *Devices* - This tab provides a list of devices that have been reserved for the project
- f. *Settings* - In this tab, you can do the following:-
 - *Activate or Deactivate a project* - You can do so by enabling/disabling the checkbox
 - *Making the project a Universal Project* - When a project is designated as a universal project, any new user who signs up onto the RobusTest platform gets access to the Universal project

3. User

This section provides details of all active and inactive users on RobusTest

On selecting a user by clicking on their name, the following additional information pertaining to that user are available:

- a. *Sessions* - Details of the last 100 test sessions that were started by the user are visible.
- b. *Usage* - This tab provides the breakup of the total duration spent on each type of test session by the user
- c. *Projects* - This section provides a list of all projects that the user is a part of
- d. *Settings* - This section enables you to:-
 - Activate/Deactivate a user on RobusTest
 - Grant/Revoke admin privileges for a user on RobusTest

4. Model

This section provides details of all active and inactive mobile device *models* being used on RobusTest.

A device model is a combination of the Model name (e.g. Mi A2, Samsung Galaxy S7, iPhone 7, etc.) and the Android/iOS version

On selecting a model name, the following additional information pertaining to that device model are available:

- a. *Devices* - This provides a list of all devices on RobusTest that have the same model and OS version running on them
- b. *Settings* - Under this section, you can:
 - provide various information pertaining to the model such as Model name, Model brand, Model manufacturer, CPU, RAM, Screen Ratio, Screen Size, Resolution, etc.
 - enable the device navigation bar/menu to be displayed in the device screen. This is menu where you would have buttons such as Back, Home, History, etc

- make available for automation all devices belonging to the mmodel by enabling the 'Support Automation' checkbox. If this check box is not selected, the all devices that fall under the model category will only be available for Manual testing

5. Device

This section provides details of all devices available on RobusTest. You can view a list of devices that are connected, disconnected or in a busy (in-use) state.

Android and iOS devices can be visually differentiated by the logo displayed on the left of the device name. The logo also helps determine the state of the device by the colour in which the logo is displayed

- Green colour - the device is connected and available for use
- Red colour - the device is in use
- Grey colour - the device is disconnected
- Blue colour - the device is in the state of being added to RobusTest. This is seen under the following circumstances: * when a device is being added for the first time * when an existing device is being restarted * when the RobusTest server to which the device is connected to is being restarted

On selecting a device by clicking on its name, the following information are visible:

- device name
- OS version running on the device
- device model
- device ID
- ADB ID
- device IMEI number
- Node server name and IP to which the device is connected
- date and time when device was last used

You can also perform the following actions using the buttons displayed on the top right:

- *Free device* - This button is visible only if the device is in use in a test session. Clicking on it, releases the device from its current test session and makes it available for a new test session
- *Restart device* - This button can be used to restart a connected device remotely
- *Flash Screen* - On clicking on this button, a red screen appears on the device for a few seconds and then goes away. This button can be used for identifying a specific device when there are multiple devices of the same make and model. It can help with proper labelling of devices for later identification

On selecting a device, a few more tabs become visible. These tabs provide the following additional information pertaining to that device:

- a. *Sessions* - Details of the last 100 test sessions that were started on the device are visible.
- b. *Usage* - This tab provides the breakup of the total duration of each type of test session in which the device was used
- c. *Apps* - This tab provides a list of apps that have been pre-installed on the device
- d. *History* - This tab provides the history of the connection and disconnection events of the device with the RobusTest server along with the date & time of these events as well as the reason for the same
- e. *Contact* - Any contact provided here will receive notification emails in the event of the device getting disconnected from the RobusTest server. To add a contact to a device, first create a contact in the 'Contacts' section

of the Admin Console. Once the contact has been added in the 'Contacts' section, you can add the same to the device from the current tab

f. *Shell* - You can run adb commands on the device from here

g. *Settings* - This section enables you to:

- provide a name for the device
- opt in or out of receiving a device disconnection email
- provide a mobile number associated with the device
- add device tags to identify the device
- identify the device groups to which the device belongs, if any
- identify the contacts associated with the device

6. Group

A group or a device group is a means by which you can restrict the usage of specific devices to specific projects.

In other words, it is a binding between one or more devices and one or more projects.

Devices that are part of a group can only be accessed by members of the projects that are part of the same group. These devices will **NOT** be available for members of other projects which are not part of the group.

This functionality comes in handy when there are multiple teams accessing the same device cloud and each team has their own set of devices on the cloud. Grouping your devices helps you ensure that the devices that you need for testing your projects are always available to your team.

7. Session

This section displays details of the last 100 test sessions created by all users on RobusTest. It also displays the count of test sessions that are in progress at that moment of time.

8. Node

This section provides details of all nodes available on RobusTest.

Each node is basically a RobusTest server to which devices are connected. The RobusTest device cloud is made up of a number of interconnected nodes or servers with devices attached to one or more of them.

On selecting a node by clicking on its name, the following information are visible:

- node name
- node IP
- date & time the node was last updated
- date & time till which the node will function (this is usually in sync with the RobusTest license period)

You can also perform the following actions using the buttons displayed on the top right:

- *Flash Screen* - On clicking on this button, a red screen appears for a few seconds on each device connected to the server/node and then goes away. This button can be used for identifying all Android devices connected to that node.
- *Create Snapshot* - Clicking on this button captures details of all devices that are successfully connected to the server at that point in time. These details are now visible on the 'Snapshot' tab for each node. This can be used for comparison at a later point of time to identify the devices that are no longer seen connected to the server.
- *Delete Node* - This button is to delete a node entry under the Node section for a node that is no longer valid.

On selecting a node, a few more tabs become visible. These tabs provide the following additional information pertaining to that node:

- a. *Devices* - This provides a list and details of all devices that are connected to the RobusTest node
- b. *History* - This tab provides the history of the connection and disconnection events of the RobusTest server along with the date & time of these events as well as the reason for the same.
- c. *Snapshot* - Clicking on the 'Create Snapshot' button captures details of all devices that are successfully connected to the server at that point in time. This can be used for comparison at a later point of time to identify the devices that are no longer seen connected to the server
- d. *Contact* - Any contact provided here will receive notification emails in the event of the node getting disconnected (i.e, the machine is either switched off or is unreachable). To add a contact to a node, first create a contact in the 'Contacts' section of the Admin Console. Once the contact has been added in the 'Contacts' section, you can add the same to the node from the current tab.
- e. *Settings* - This section enables you to update the following information about the node:
 - Node Name
 - Node Location
 - Node Mac Address
 - Node Machine Serial

9. Integrations

RobusTest enables you to integrate with any API enabled CI tool like JIRA, Asana, etc.

In order to integrate with such tools, you first need to create a configuration in the 'Integrations' section of the Admin Console.

To integrate with a tool:

- a. click on the 'Create New Integration' button
- b. select a tool from the drop down provided. A list of fields that enable integration with the tool are now displayed
- c. enter relevant values for the fields displayed and click on the 'Create Configuration' button

E.g., let's say you need to integrate with JIRA to log bugs encountered while testing your app. You need to do the following: * select JIRA from the tool drop down list * provide information such as the JIRA Server URL, JIRA username, JIRA API Token, etc. * create the configuration setting * now, on the Project Dashboard go to the 'Settings' tab and select the name of the configuration you created on the 'Bug Tracker' dropdown

In case you do not find the CI tool of your choice on the tool drop down list, please reach out to the RobusTest support team by emailing us at support@robustest.com and our team shall get back to you for further assistance with integration

10. Contact

This section enables you to add contact details about one or more persons. These are folks who should be informed in the event of a device or node disconnection.

Once you have created contacts, you can add these to the contact list on the Device and Node sections of the Admin console so that they receive notification emails.

11. Settings

This section enables you to configure various parameters on the RobusTest platform.

More details are available in the settings page

CHAPTER 18

Continuous Integration

1. Remote uploading a build
2. Triggering a test run
3. Monitoring a test run
4. Accessing the results

CHAPTER 19

Integrating Bug Tracker

20.1 Adding new devices to RobusTest - Android

When adding a new Android device to the RobusTest platform for the first time, you need to first prepare the device by performing the following set of actions on it:

1. Turn on Developer Mode
 - Go to the 'About Phone' section under Settings on your mobile device
 - Now, tap on the build number field 7 times continuously to turn on 'Developer Options'
2. Stay Awake should be turned on
3. USB Debugging should be turned on
4. Allow mock locations
5. Set USB connection behavior to act as media device
6. Set location as only GPS (option may be "only device" on some devices)
7. Remove any lock screen
8. Security settings - Allow installation from unknown sources
9. Set as true "Do not verify apps installed over USB"
10. Display - Set rotation as off
11. Display - Set screen timeout to Never (in case Never is not an option set it to maximum possible time available)
12. Once the above steps are completed, plug the device into the node server through a USB cable. Make sure you use a good quality cable for better data transfer, reliable connectivity and better device charging.
13. Upon connecting the device, an alert message with the title "Allow USB Debugging?" should show up on the device screen. Select the checkbox to "Always allow from this computer". Tap on OK on the device
14. Once the device is plugged in, the system will take upto 2 minutes to identify the device and add it to devices list.

15. Confirm that you are able to view the device in your device list from manual and automation testing.

Additional Steps for Xiaomi or MI devices:

Instructions for Mi/Xiaomi devices

1. Security App

- a. Tap on the Security App
- b. Tap on the Permissions Icon
- c. Tap on the Settings icon on the top right
- d. Make sure Install via USB is TURNED OFF

For new device models:

- a. Tap on the Security App
- b. Tap on the Settings icon on the top right
- c. Tap on 'Security Scan'
- d. Disable 'Scan before installing'
- e. Disable 'Auto-updates'
- f. Disable 'System Updates'

2. Developer Settings

- a. Tap on Settings
- b. Tap on Additional Settings
- c. Tap on Developer Options
- d. Developer Options - TURN ON
- e. USB Debugging - TURN ON
- f. Install via USB - TURN ON
- g. USB Debugging (Security Settings) - TURN ON
- h. Verify apps over USB - Turn OFF
- i. Turn on MIUI Optimization - TURN OFF

3. Privacy

- a. Tap on Settings
- b. Tap on Additional Settings
- c. Tap on Privacy
- d. Unknown Sources - TURN ON

P.S. Some older MI devices may have only one of the above options (i.e., i and ii) available and/or may be named differently. Enable the relevant option

Additional Steps for RealMe devices:

- a. In 'Developer Options', enable the setting 'Disable Permission Monitoring'

Additional Steps for Vivo devices:

- a. Go to 'Settings' -> 'Battery' -> 'High Background Power Consumption'
- b. To the 'Allow' list, add `io.appium.uiautomator2.server`

Caveats:

1. On Samsung device models, you might encounter a pop-up mwindow with the message - "Attention - The connected device is unable to access data on this device. Reconnect the USB cable and try again". * This is a known issue in Samsung models. * You can try one or more of the following solutions to resolve this:
 1. Disconnect and reconnect the USB cable.
 2. Install 'Android File Transefer' or 'Samsung Smart Switch' applications on your Mac machine.
 3. Reboot the device and reboot the server to which your device is connected.
 4. Update your device Android OS software.
 - *Note:* The above solutions do not guarantee that the window does not come up again.
2. Devices running on older Andorid versions tend to be slower in response.

20.2 Adding new devices to RobusTest - iOS

Setting up your iOS device

1. On connecting your iOS device to the RobusTest server, you will see a pop-up asking you if you would like to trust the server. Click on the 'Trust' button.
2. On the iOS device, go to *Settings* -> *Safari* -> *Advanced* and enable the option named 'Web Inspector'
3. In *Settings* -> *General* -> *Display & Brightness*, set the 'Auto-Lock' option to 'Never'.
4. Ensure that the device is added to your developer certificate.

Interaction with the iOS device screen in a Manual test session sometimes get blocked on account of certain notification pop-ups that appear on the iOS device.

In order to avoid the same, you need to disable the following pop-ups:

Disable iOS Software Update Notification

1. Go to *Settings* -> *iTunes & App Store*
2. Click on the toggle button to disable *automatic app updates*
3. Click on the toggle button to disable *automatic downloads*

Delete existing software downloads

Sometimes, an update has already been downloaded onto your iOS device. This results in the software update notification coming up even if you have disabled the *Auto-Update* option. To prevent such pop-ups, do the following:

1. On your iOS device, go to *Settings* -> *General*
2. Tap on *iPhone Storage* (for iPhones) or *iPad Storage* (for iPads), depending on the type of iOS device that you are using
3. On scrolling down, you will see a list of apps and the amount of storage that each of them are using. In this list, look for the latest iOS update. It would be in the format 'iOS 13.1.1' or 'iOS 14 Beta 2', etc.
4. Tap on the update.

5. Now click on the “Delete Update” option and confirm. The downloaded update is now deleted.

Disable iMessage Notification

1. On your iOS device, go to *Settings* -> *Messages*
2. Disable *iMessage* by clicking on the toggle

Disable FaceTime Notification

1. On your iOS device, go to *Settings* -> *FaceTime*
2. Disable *Facetime* by clicking on the toggle

20.3 Best Practices in Automating Tests using RobusTest

1. Name test steps appropriately

- When recording a test case, RobusTest automatically generates test steps based on the actions performed. Test steps are created with the intention of being easily understandable.
- However, in many cases the test steps are not intuitive because of various reasons e.g. no content-desc provided for the element.
- It is, therefore, recommended that the test steps are reviewed and checked for easy understanding for all users. In case required, test steps can be easily renamed.
- **This will help in :-**

- easily understanding the test case
- faster and easier debugging of errors from the functional reports

E.g. the default step name that says ‘Tap on Relative Layout’. This makes it difficult to understand from the reports or from the test script on what the intended action is.

Instead, the test step should be renamed in a more meaningful manner such as ‘Tap on the option Recharge’.

2. Use functions wherever possible

- RobusTest enables creation of functions out of existing test cases. This is a powerful feature that helps in reducing the time & effort to automate. It also enables easy maintenance and update of test code in case of changes in the mobile application.
- Use functions wherever a series of steps form a logical unit, e.g. series of steps that capture customer data, selecting a usage plan, selecting a rental plan, etc.
- For actions that are repeated across more than one test scenarios, the use of functions helps with re-usability, thereby reducing the time spent in automation.
- Using functions will significantly reduce the amount of future time spent in incorporating changes to app flows, data, element resource ids, etc.
- At the very minimum have one function to capture the actions performed on each screen of the app.
- You can break this down further based on how a particular app screen is structured. Eg., on a single page where you are entering customer details, capturing images, selecting a number, etc, each can have a function of its own. The ‘customer details’ function can itself be broken down into functions for ‘personal details’, customer address’, etc. as applicable

E.g. Say, there are a series of standalone test steps (i.e., test steps not inside a function) where a 4G Plan is being selected and that these steps are repeated in a number of test cases. In future, if the specific plan being selected changes, you would have to go to each and every test case and update them.

Instead, have these steps within a function named 'Select Rental Plan'. Now, any change in the way a rental plan is created would only require a one time change within this function script. This change is reflected in all test cases that call this function.

3. Perform a 'Swipe' action in small increments

- While automating a 'Swipe Up' or a 'Swipe Down' action, it is recommended to do so in small incremental movements, instead of one single long swipe across the screen. This would ensure that the same automation script will work across device screens of different sizes.

4. Use the element attributes to define better XPath

- RobusTest is designed to create highly accurate and robust XPath for elements in the mobile app. However, in certain cases the auto-generated XPath may not work due to various reasons e.g. duplicate resource ids, non-existing resource ids.
- When tapping on a button or a link with a text, which is not subject to frequent changes, it would be a good practice to improve the XPath by adding to it element attribute values that can help in identifying the element uniquely e.g. 'text' or 'content-desc'.
- The XPath, thus modified to make it further unique, makes it easier to identify elements when the resource id is not unique or non-existent.

5. Use the 'Find Concurrent Elements' functionality

- Often times we find multiple elements overlaying one another around the same geographical area within a screen of an app. Using the 'Find Concurrent Elements' option will help in this scenario to identify the correct element that the user wants to perform an action on.
- Using the 'Find Concurrent Elements' option will display a list of elements that occupy the area within the app where the user has performed a left-click action. Select the required element and click on 'OK'. Now, any subsequent action performed, such as Tap, Store, Verify, etc, is done on the element selected in the previous step from the list.

6. Use meaningful names for Stored variable

- When storing a value in a variable, ensure that the variable name is something meaningful and unique.

E.g Initial Data Balance for Postpaid, Customer First Name, etc. This makes it easier to pick the right variable when doing a verification against it.

7. Mention the variable name in test step while using the 'Store' option

- When using the 'Store' option to store a value in a variable, by default, the test step gets generated with a description in the format: Store <value> for later verification.

E.g. Store "10" for later verification <- Here 10 may be the initial data balance in GB for a postpaid customer

- Mentioning the variable name in the test step makes it easier to understand the test script and the functional reports. E.g. Store initial postpaid data balance for later verification

8. Provide proper descriptions for test cases

- While saving an automation test script, ensure that a meaningful name and description is provided.
- When a test case fails, a relevant description helps the user to identify the business flow correctly without spending too much time on it.
- In the 'Test Cases' page, the user can search for a test case based on test case name and test case description. A meaningful description goes a long way in identifying the test case and reducing the time spent in looking for it

9. Name functions appropriately

- **When converting a test case into a function:-**

- give the function a name that makes its purpose very clear to all users
- name it in a way that the test case name/description can be easily associated with the function name. This helps if the user wants to make a change in the function and is looking for the corresponding test case

10. Use Reports for debugging functional and performance issues

- The Functional and Analytic reports can help a lot in debugging errors in scripts

- a. Functional Reports

- The error message at the top of the functional report for a test case run tells you why the test case failed.
- The test step at which the test case failed can be seen in 'Red'.
- Make sure you have a look at the test step and screenshot just before the above failed step. Often the real reason for failure can be observed in this step.

- b. Analytic Reports

- If you observe that the app suddenly gives way to the device Home screen, have a look at the device logcat section within the Analytic report
- Go to the section named 'Log' under Analytic report and scroll down to the bottom. If the last few entries in this section show that the app is being closed or a 'Kill App' instruction is being executed, it means that the App is crashing at this point.
- If the App seems to be taking an inordinately long period of time to load a page or transition between screens, have a look at the usage of CPU, Memory, Network, etc. This would provide the developers an idea on where to focus on to improve the app performance.
- Look at the number of external calls to the Garbage Collector (GC). A huge number of external GC calls within a small time period indicates inefficient usage of memory leading to slower performance of the app.

1. *Adding new devices to RobusTest - Android*
2. *Adding new devices to RobusTest - iOS*
3. *Best Practices in Automating Tests using RobusTest*

.._robustest_connect:

RobusTest Connect - Run local, test global

RobusTest Connect enables you to use the RobusTest platform to test on devices that are connected locally, i.e., on your own computer.

This enables you to test on your device while at the same time track runs and generate reports on RobusTest.

Your device now behaves as a part of the RobusTest Device Lab and is accessible to any member of your project on RobusTest.

To enable RobusTest on your local device, you need to do the following:

1. Download the RobusTest Connect software
2. Install essential software on your local machine
3. Set up RobusTest config files
4. Start the RobusTest Server

1. Download the RobusTest Connect software

1. Download the *robustest* folder from the link that is provided to you
2. Within the *robustest* folder create the following folders and sub-folders:
 1. platform_data
 2. tools
 3. tools/android
 4. tools/idevicelocation

2. Install essential software on your local machine

A pre-requisite to running RobusTest on your machine is the installation of essential software for testing your Android and iOS devices

A list of these software is given below. Click on each of them to find instructions for installation:

1. install-java
2. install-android

3. install-python
4. install-python-virtual-env
5. install-nodejs-appium

If you are using a Mac computer and would like to test on iOS devices, you need to additionally install the following software:

1. install-xcode
2. install-homebrew
3. install-idevice-location

3. Set up RobusTest config files

1. Open the config.yaml file inside the robustest folder
2. In the config.yaml file, provide:
 1. the RobusTest server IP (in IP:port format) or the RobusTest server URL
 2. the License Key
3. In the StartRobusTestConnect file provide the following information:
 1. RobusTest URL
 - This will be the RobusTest server IP or the RobusTest server URL
 2. Node Name
 - This is a custom name you provide by which to identify your computer as a Node on RobusTest
 3. License Key
 4. Path to Android ADB (if required)
 5. Path to the NodeJS/bin folder (if required)
 6. Path to virtual environment (if required)
 7. Protocol to be used (http or https) (if required)

4. Start the RobusTest Server

1. Configure your Android device as mentioned in the following page: <http://docs.robustest.com/en/latest/addnewdeviceandroid.html#adding-new-devices-android>
2. Configure your iOS device as mentioned in the following page: <http://docs.robustest.com/en/latest/addnewdeviceios.html#adding-new-devices-ios>
3. Connect your test device to your computer using a USB cable
4. For Windows computers:
 1. Open *Command Prompt* on your computer and navigate to the ‘robustest’ folder
 2. Run the following command: `.\StartRobusTestConnect`
5. For Mac and Linux computers:
 1. Open *Terminal* on your computer and navigate to the ‘robustest’ folder
 2. Run the following command: `./StartRobusTestConnect.sh`
6. The server has now started running on your laptop
7. Login to RobusTest on your browser and go to the Admin Console -> Node section

8. You should now be able to see your computer as one of the nodes on RobusTest
9. You will also be able to see and use your local Android/iOS device on RobusTest

22.1 Unable to access RobusTest Server

If a RobusTest server is unavailable, please contact your IT team to look into this.

The IT team can follow the following points to troubleshoot

A RobusTest server may be inaccessible due to one of the following reasons:

1. The network/LAN connection is down

- Check if the RobusTest server is accessible on the network/LAN
- Check if the LAN cable has been plugged into the server correctly
- Check if the LAN cable is faulty

2. The server machine is down

- Check if the server machine is shutdown due to a power failure at the server room
- Check if the server machine is shutdown due to the backup power failure
- Check if there is an issue with the power strip to which the server machine is connected
- Check if there is a hardware issue with the server due to which it is shut down

3. Remote Login to the machine is disabled

- Login to the server machine
- Click on the 'Apple' logo displayed on the top left corner of the desktop
- In the menu that opens, Click on '*System Preferences*'
- On the System Preferences window that opens, click on the option '*Sharing*'
- Enable the check box named '*Remote Login*'.
- On the right side of the window, under the option '*Allow Access for*', choose the option '*All Users*'
- Enable the check box named '*Remote Management*'.

- On the right side of the window, under the option ‘Allow Access for’, choose the option ‘All Users’



Image credit-Dilbert

There are two ways to deal with this. We either say that you will never land in inexplicable situations with our product or we give you a list of possible places where you may be stuck and subtly shift the blame on you. We will take the latter route.

Following are some issues that you may come across when using RobusTest. In case you do not find the issue you are facing on this page, log a support ticket by clicking [here](#)

I tried to click on Manual or Automation buttons but I get a message Unable to reach node server

This may happen because your computer is accessing RobusTest through a proxy server and the server is blocking WebSocket protocol

I ran my automation tests on the RobusTest Hub but I do not see anything in Live View

A build URL is generated uniquely for every build and user. So, this could be because the build URL you are using to run your tests belongs to another person (and s/he is the one seeing an entry in his/her live view)

I am unable to see the test steps in the Live View for my device

In case you are using RobusTest from behind a firewall, it is possible that the information related to test steps execution is being stopped. One check for this would be to run the following command

```
telnet <RobusTest Installation Address> <port#>
```

In case the connection is not successful, it is probable that the reason for not being able to view the test steps in Live View is the stopping of information by the firewall.

I am unable to access RobusTest

Make sure your computer is connected to the network. Check if the server is running and reachable by executing a ping command on the RobusTest server

```
ping <RobusTest server IP address>
```

If the ping command is unsuccessful and you are unable to get a response from the server, it either means that the server is unreachable or that the server is down. Please check on this with your IT team. If the ping command is successful and you are able to reach the server, then we check if the RobusTest server is running successfully.

Visit the RobusTest processes page and ensure that all processes are running as expected. The URL for RobusTest processes page looks as below

<http://<robustest server url>:9001>

Make sure all the processes are in running state. In case the URL above is not reachable, it means the RobusTest server is not running. Start the RobusTest application by running the following command on the system. You will need to login to the RobusTest server using putty or similar tool.

```
# cd /opt/code/RobusTest # screen -S FrontEnd # sudo su (enter the password when prompted) # ./StartServer.sh
```

To find out the IP addresses of the deployment servers or other such details, please contact your RobusTest administrator.

Issue: *Unable to access RobusTest Server*