MOZARK

User Guide
App Functional

Version 01

Contents

1. What is App Functional?

App Functional is an app testing service that you can use to test mobile apps and mobile websites on real, physical phones that are hosted by App Functional. App Functional facilitates automated testing of apps using a variety of testing frameworks such as Appium, Calabash, Robot-Appium, Robot-UIAutomator, UIAutomator, etc with different languages such as Java, Node.JS, Python, Ruby.

App Functional allows you to upload your own tests in the form of scripts. Because testing is performed in parallel, tests on multiple devices begin in minutes. As tests are completed, test results that contain high-level report, low-level logs, live logs, screenshots, recordings, and performance data are updated.

To use App Functional, the first step is to sign up.

If you do not have the App Functional Sign Up account, please complete the following steps:

1. Open http://demo-appfunctional.mozark.ai/

2. Follow the online instructions to ensure a successful sign up on App Functional.

2. Getting Started

This walkthrough shows you how to use App Functional to test a native Android or iOS app or mobile website. You use the App Functional console to create a project, upload a build in the form of an .apk or an .ipa or choose a default one, select devices, run a suite of standard tests, and then view the results.

2.1 Sign in to the Console

You can use the App Functional console to create a project, upload a build, select devices, run test suites, check results. You can learn about projects, builds, devices, test suites, results later in this walkthrough.

Step 01: Sign in to App Functional Console at http://demo-appfunctional.mozark.ai/

You will see a page as shown in Figure 01. Provide your user id, password and click on login to sign in to App Functional Console successfully.

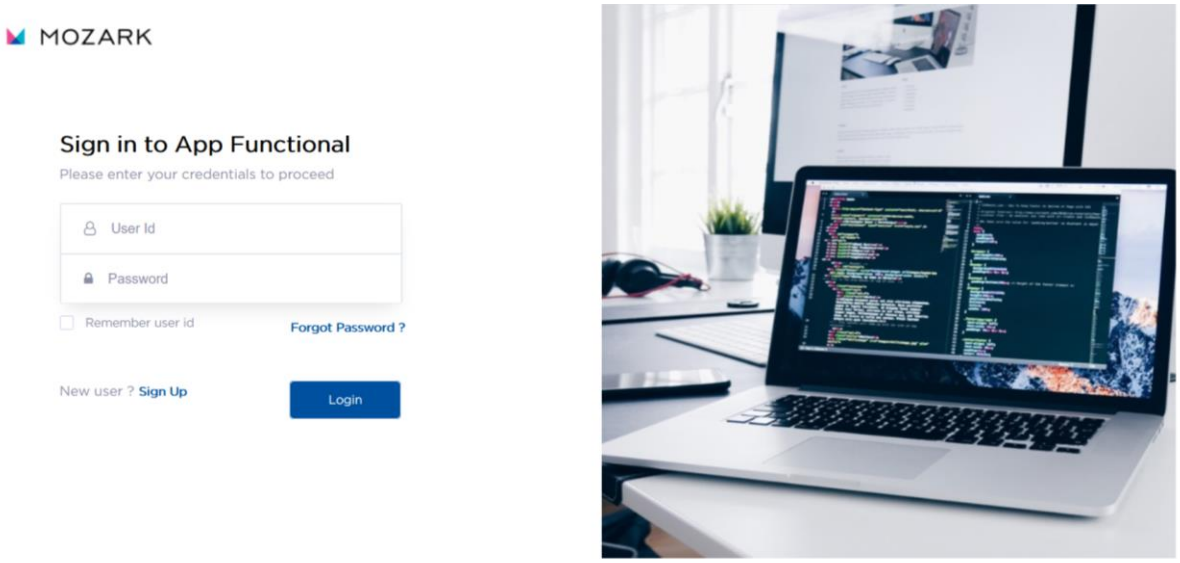

Figure 01: Sign in to App Functional Console

Step 02: Signed in App Functional Console

After you successfully sign in to App Functional Console, you will see the page given in Figure 02.

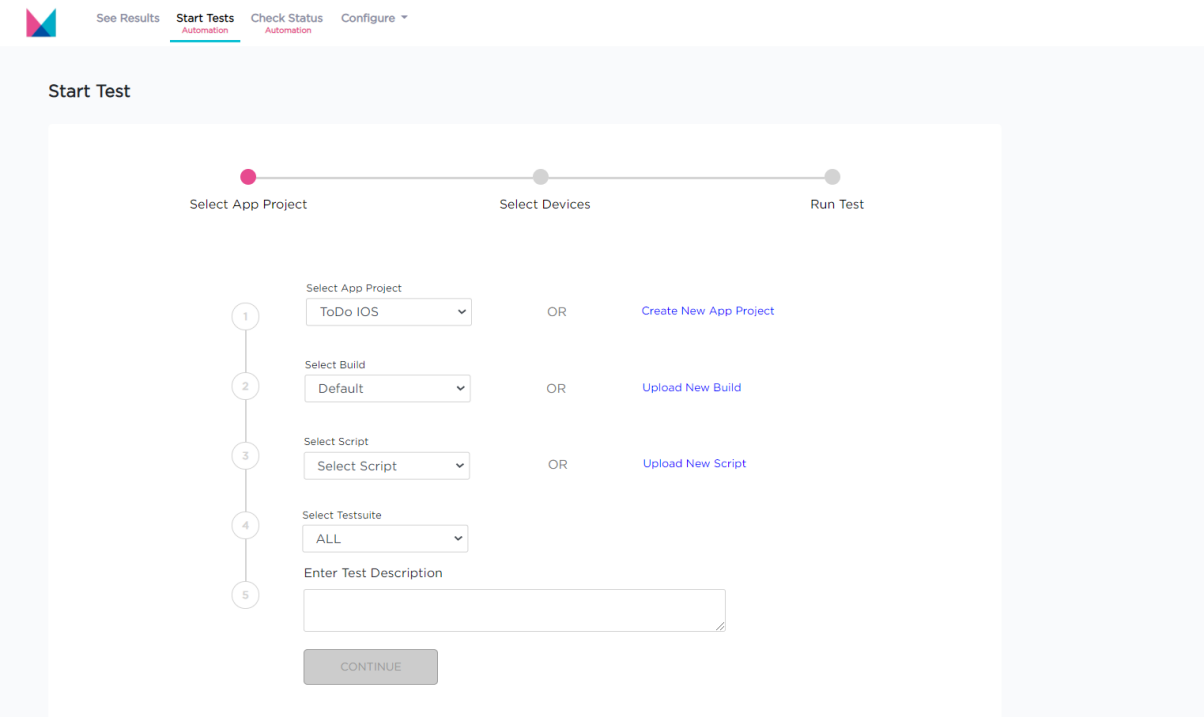

Figure 02: Signed in App Functional Console

Once you are signed in to the App Functional Console, you can create or select existing projects, upload or select default or existing builds, upload or select scripts, select test suites, choose devices, run or schedule tests, check status of test runs, view results, configure your App Functional Console.

2.2 Configure Projects

To test a mobile app or a mobile website, you must create or select an app project.

(a) Create App Project

If you are using App Functional for the first time, you will have to create a project by clicking on Create New App Project in Figure 03 or by going to Configure in Figure 04.

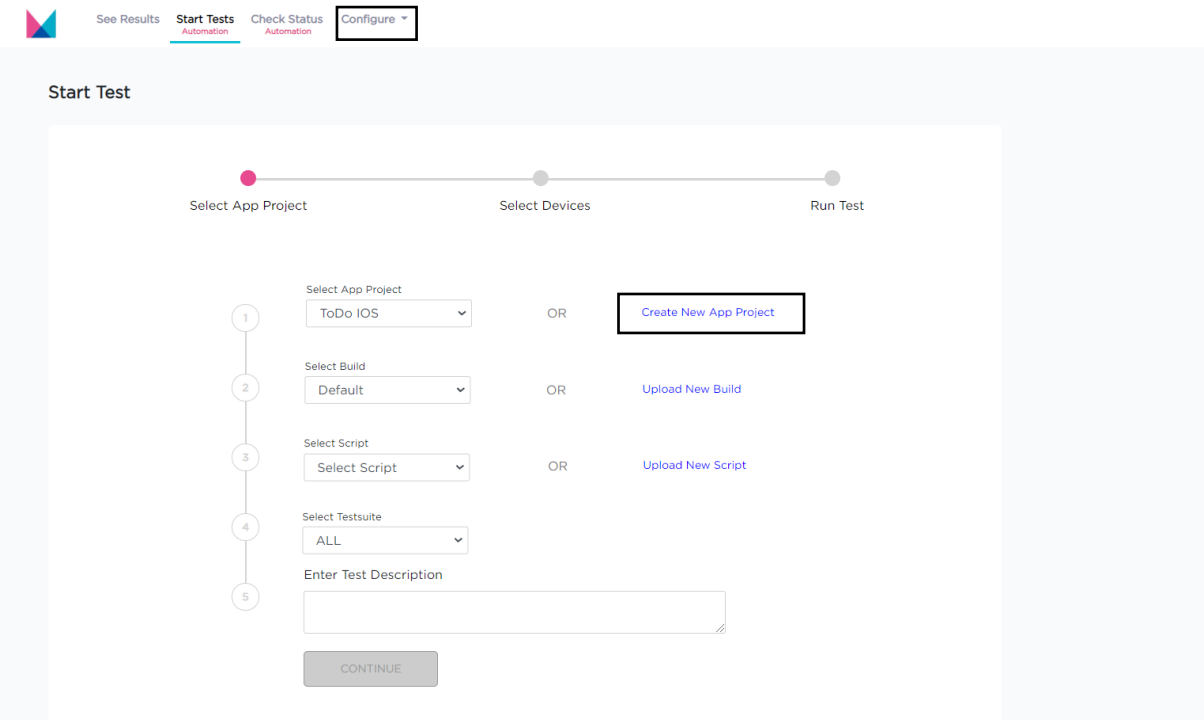

Figure 03: Click on Create New App Project or Go to Configure

1. In the Create New App Project, enter a name for your project (for example, MyDemoProject) and description in Figure 04. Both **Name App Project** and **Description** are mandatory fields.

2. Click **Create App Project** to create a project with the given project name and description in Figure 04.

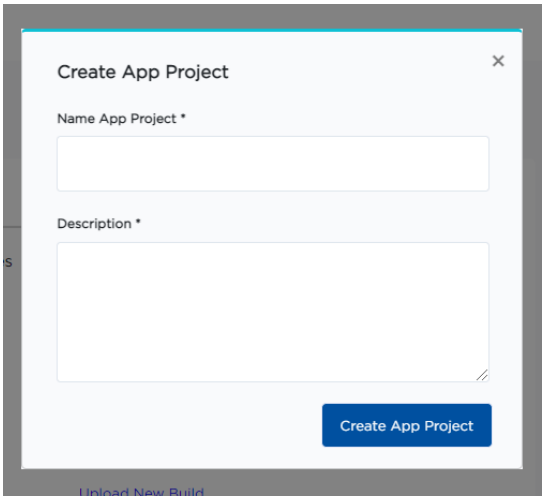

Figure 04: Create New App Project

3. Select **App Project** and you will see your newly created app project in the dropdown as shown in Figure 05.

(b) Select App Project

If there are projects associated with your account, then select App Project as shown in Figure 05.

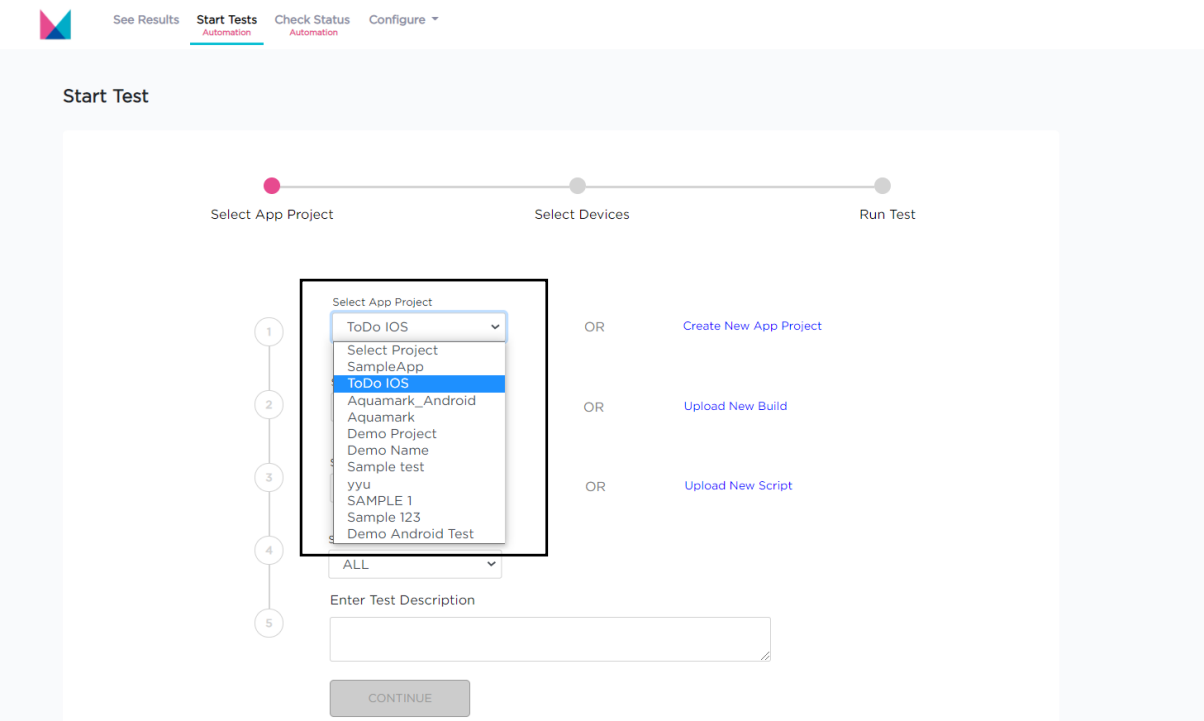

Figure 05: Select App Project

2.3 Configure Builds

Now that you have an app project, the next step is to upload a new build or select a default build or an existing build.

(a)Upload New Build

If you are using App Functional for the first time, you will have to upload a build by clicking on Upload New Build or by going to Configure in Figure 06.

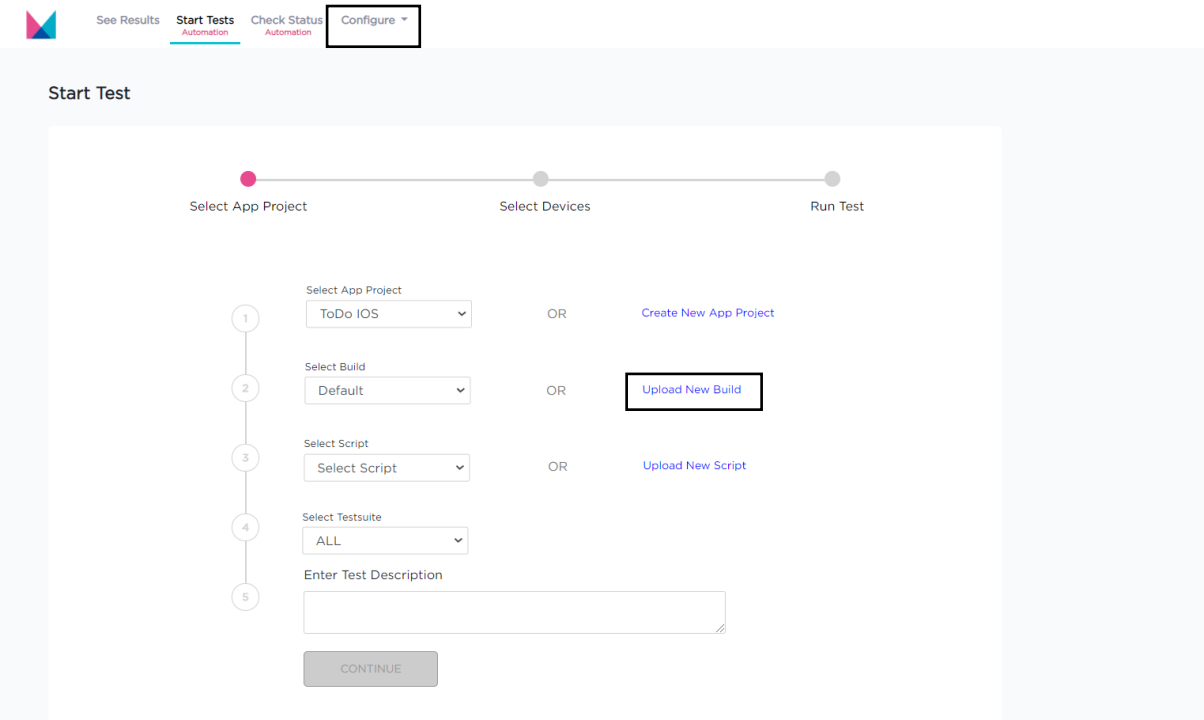

Figure 06: Upload New Build or Configure

1. In Upload New Build, select an app project from the dropdown (for example, MyDemoProject) and attachment(.apk or .ipa) from your system in Figure 07. Both **Select App Project** and **Attachment** are mandatory fields.

2. Click **Upload Build** to upload new build for an app project in Figure 07.

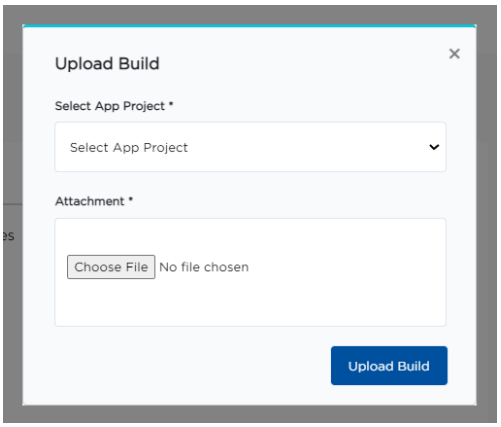

Figure 07: Upload Build

(b)Select a Build

If the builds are already available in the test devices, then select a Build.

The pre-requisite to select a default build as shown in Figure 08 is to ensure that the builds are available in the test devices. Otherwise, it will result into an error.

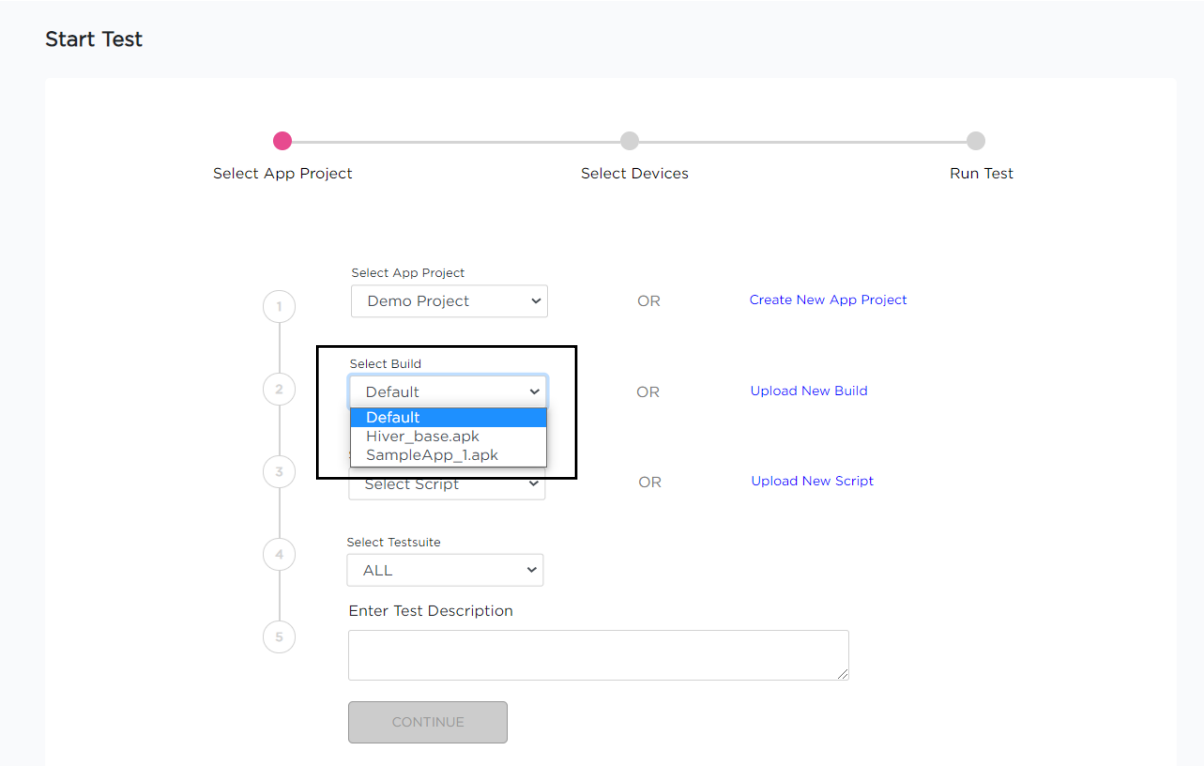

Figure 08: Select Build - Default

(c) Select a Build

If there are builds associated with your projects, then select Build as shown in Figure 09.

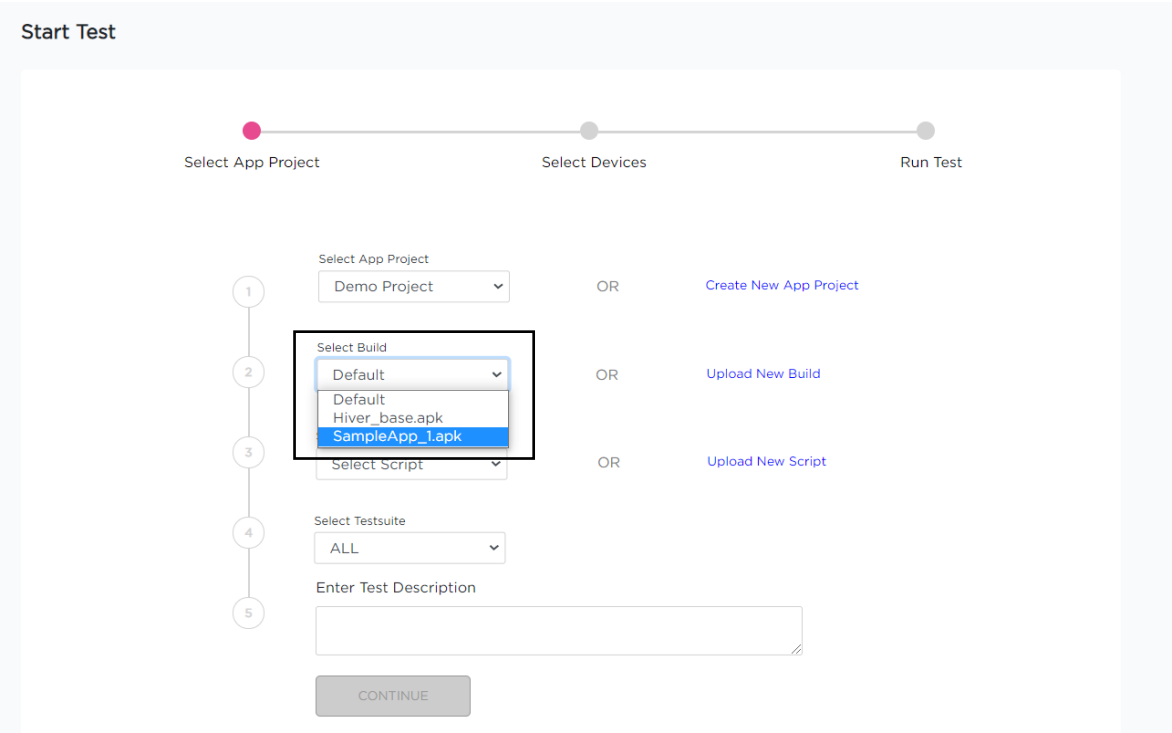

Figure 09: Select Build

2.4 Configure Scripts

Now that you have select a build, the next step is to upload a script or select a an existing script that can automate the user journey you want to test your mobile app or mobile website for.

(a) Upload New Script

If you are using App Functional for the first time, you will have to upload a script by clicking on Upload New Script or by going to Configure in Figure 10.

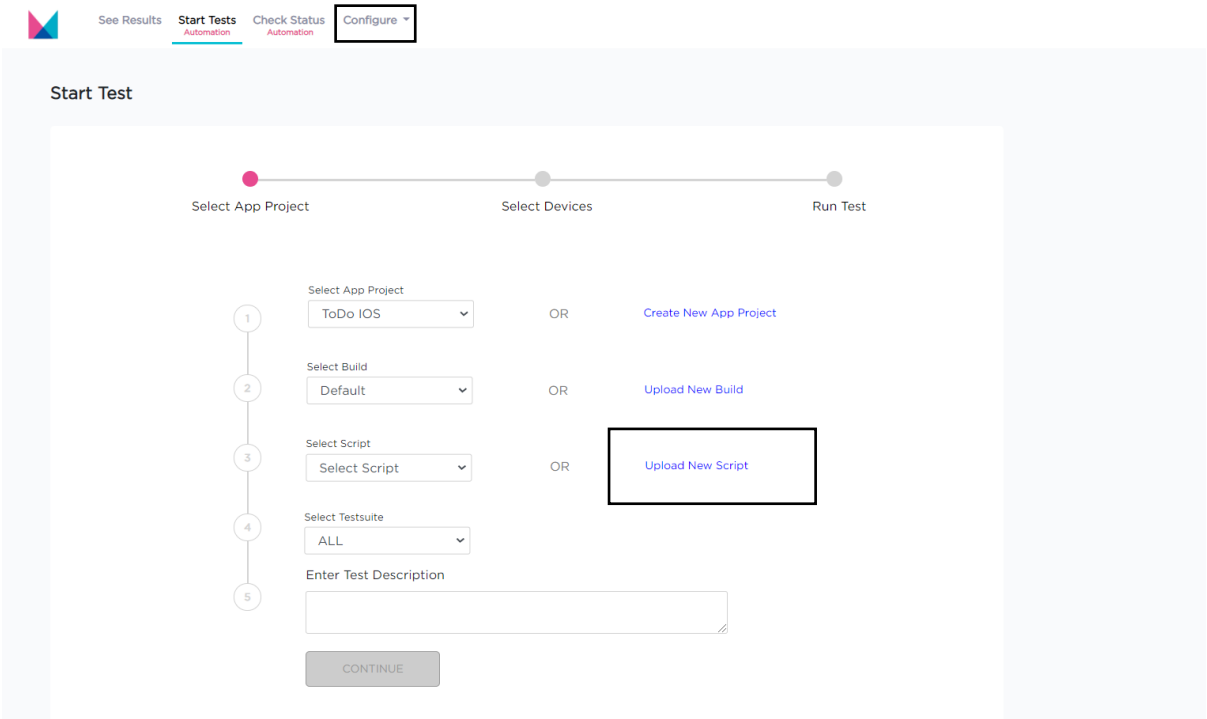

Figure 10: Upload New Script or Configure

1. In Upload Journey Script, enter a name for journey script, description, features tested, select framework, language, OS, project from the dropdown (for example, MyDemoProject) and attachment from your system in Figure 11. All the fields except selecting an OS are mandatory fields.

2. Click **Upload Journey Script** to upload new script for an app project in Figure 11.

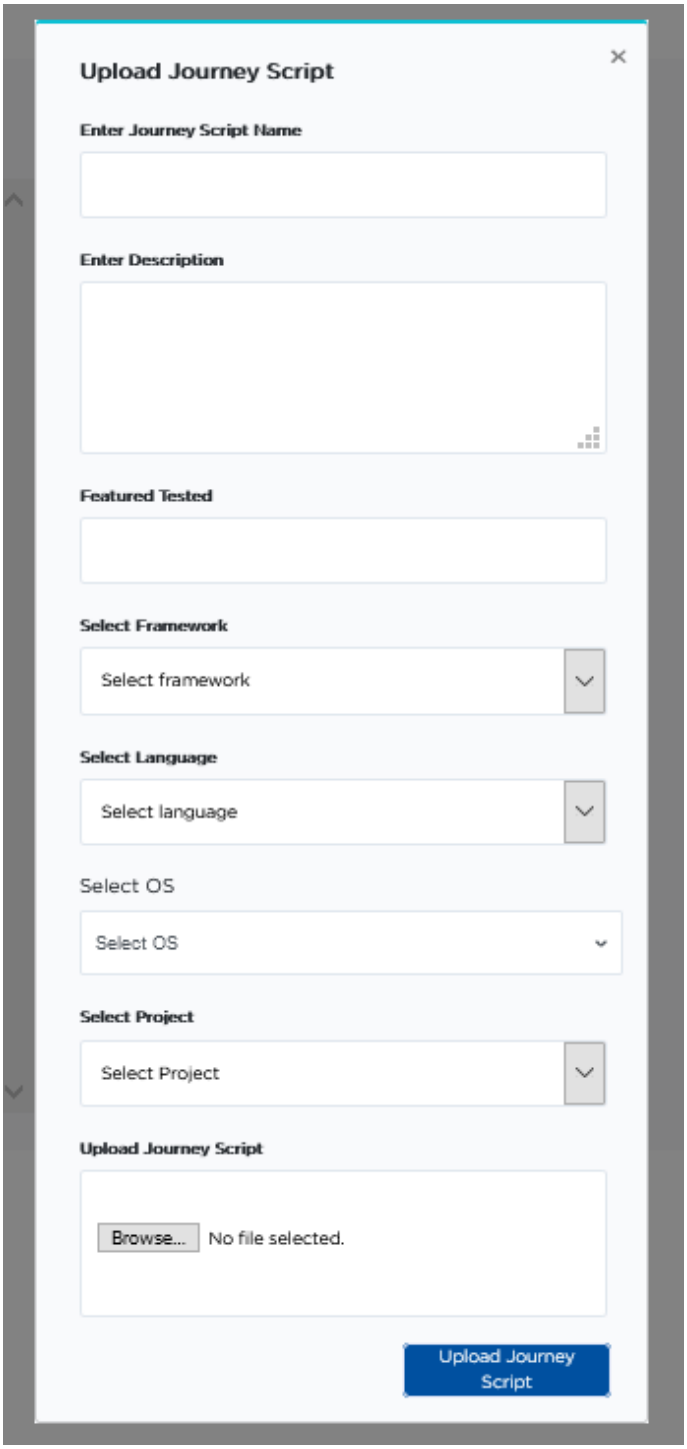

Figure 11: Upload Journey Script

(b) Select a Script

If there are scripts associated with your projects, then select a script as shown in Figure 12.

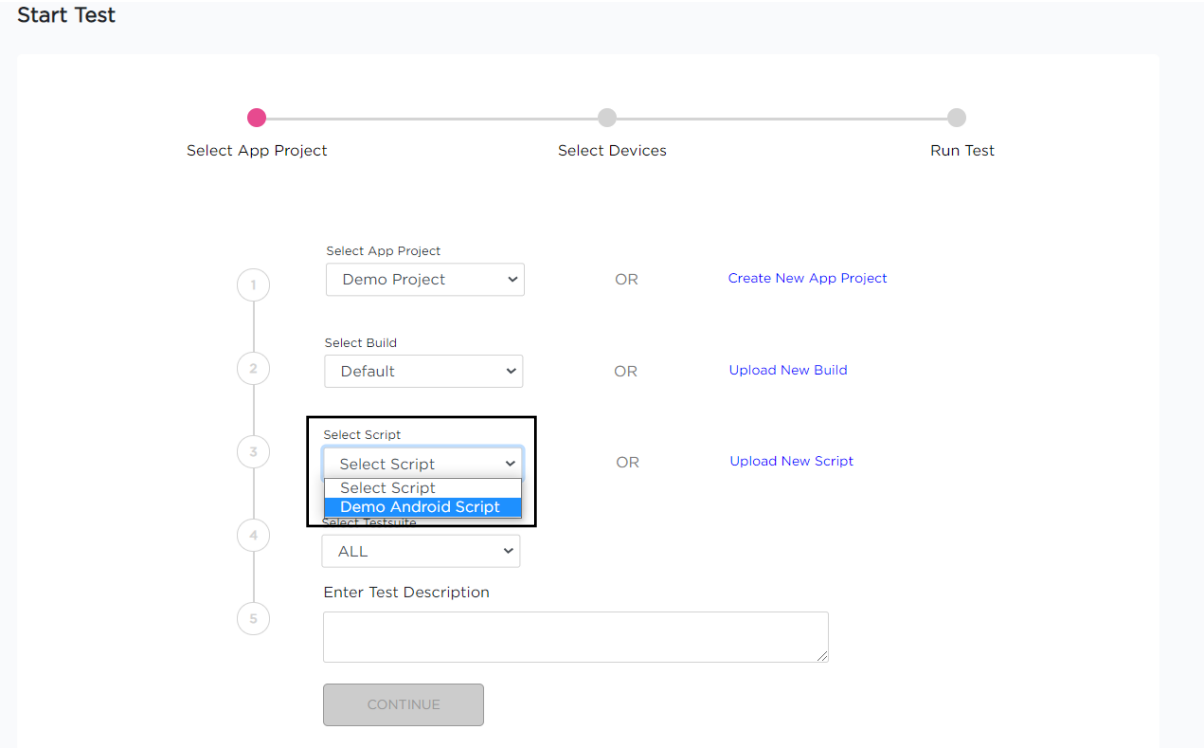

Figure 12: Select Script

2.5 Select TestSuites

If there are test suites associated with your projects, then select a test suite or **ALL** test suites for running them on test devices as shown in Figure 13. TestSuites are not available for Appium with Java TestNG scripts.

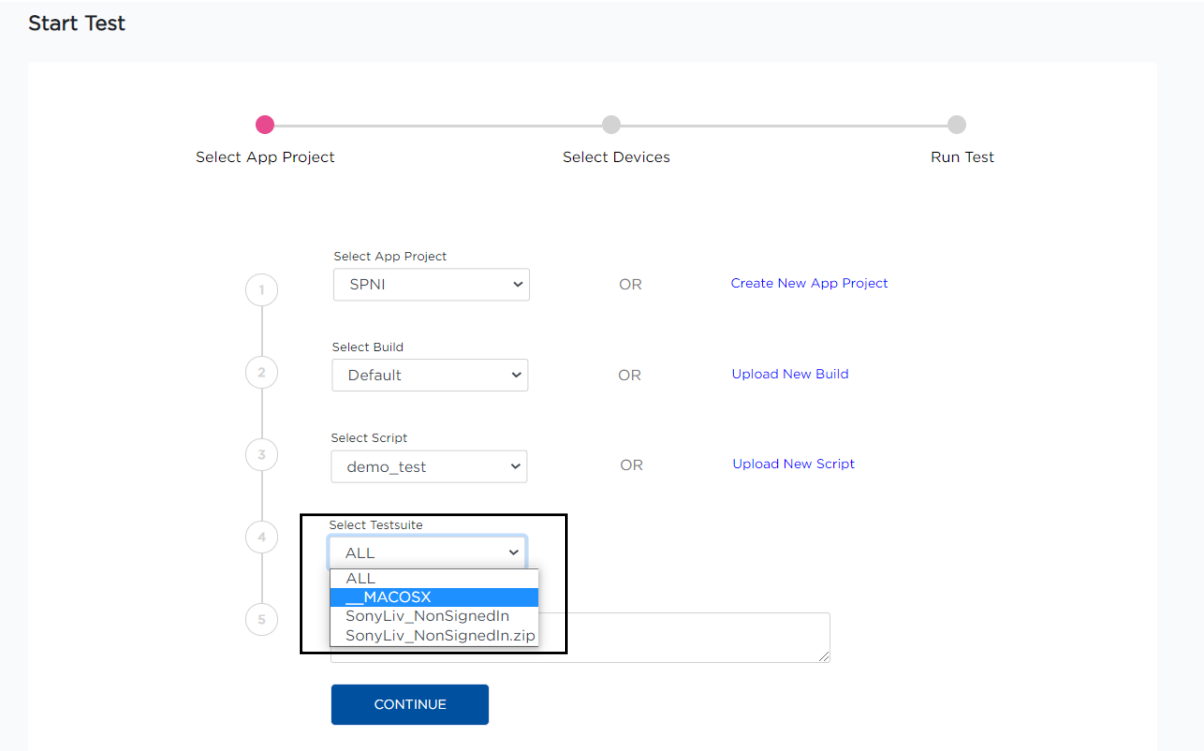

Figure 13: Select Testsuite

2.6 Enter Test Description

If you want to provide a name for this test run, then write a test description as shown in Figure 14. It is not a mandatory field.

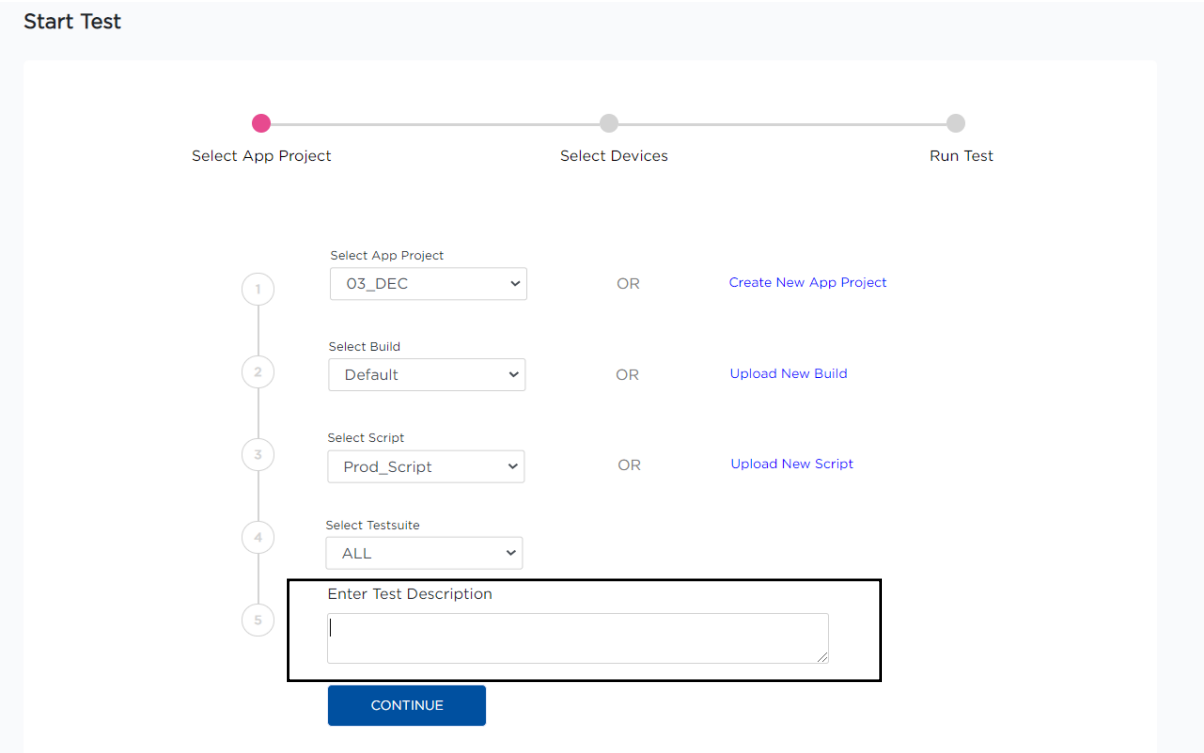

Figure 14: Enter Test Description

2.7 Select Devices

Now that you have selected project, build, script, test suite, test description, the next step is to choose devices to test your mobile app or mobile website.

Click on Continue as given in Figure 15.

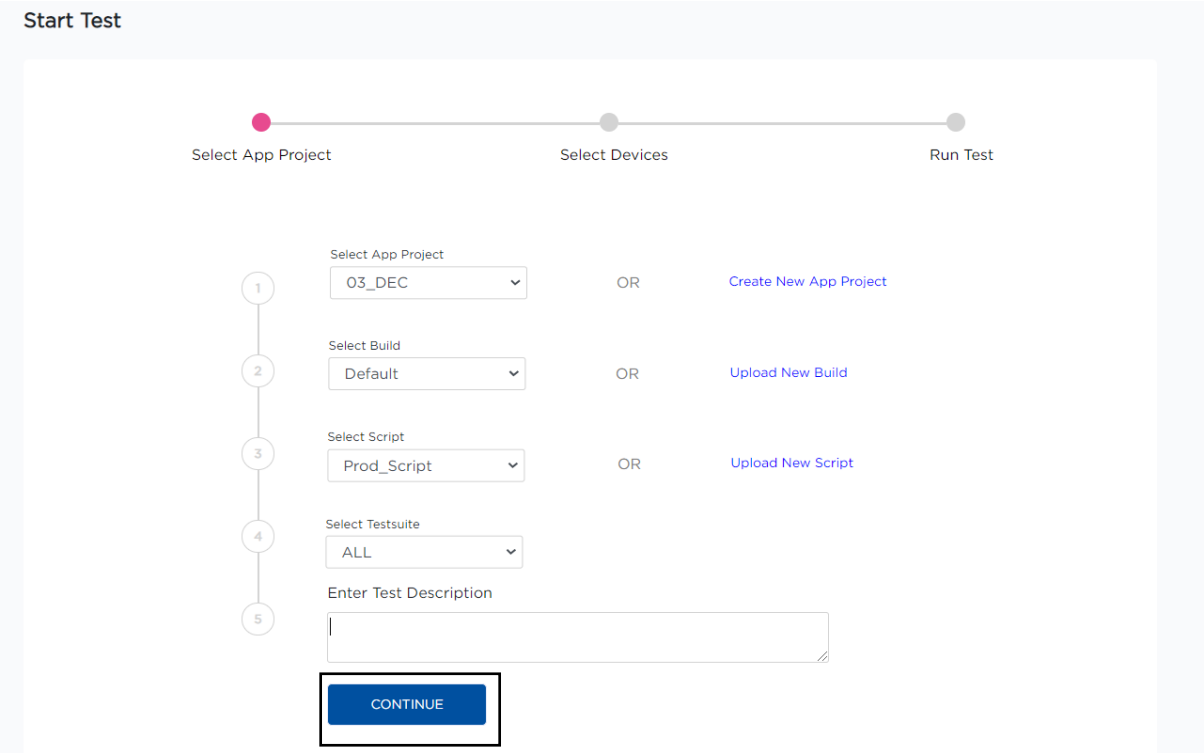

Figure 15: Click on Continue

By clicking on Continue, you will see Select Devices page as shown in Figure 16.

You can select devices in two ways –

(a) Select device group
(b) Select devices

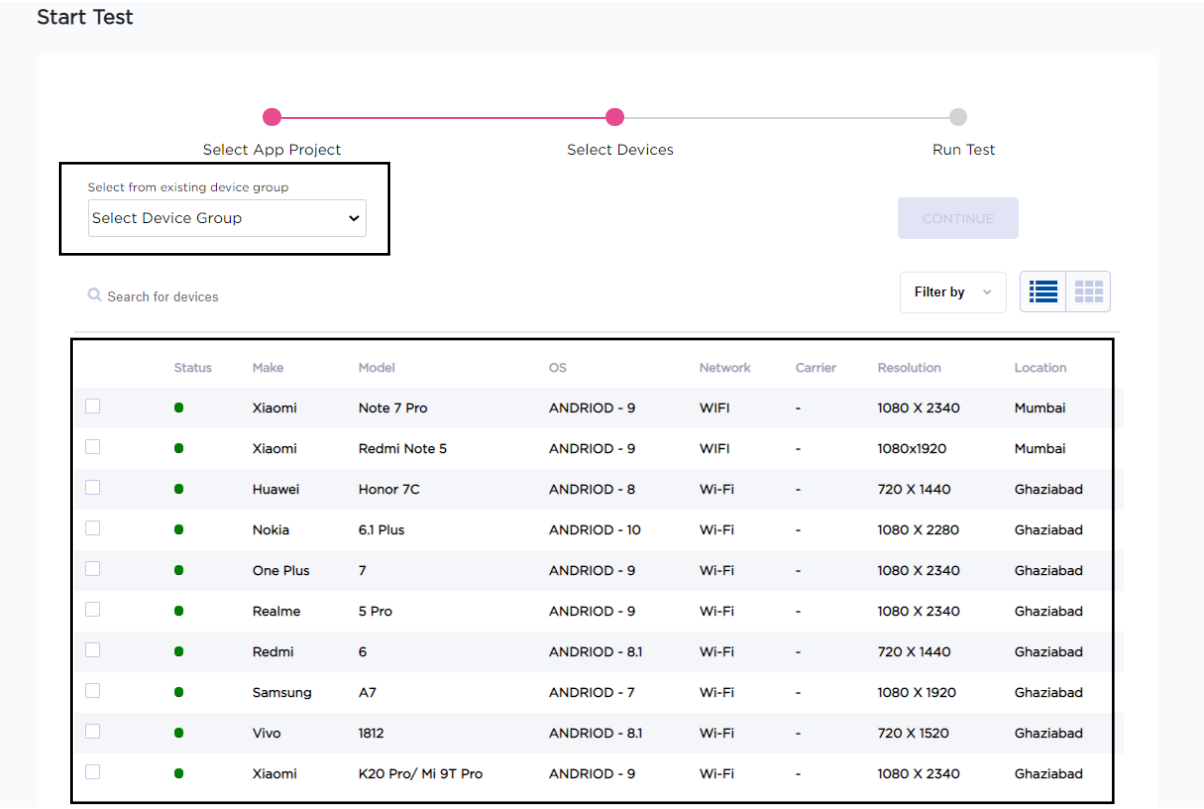

Figure 16: Select Devices

(a) Select from existing device group

Select a device group of choice by clicking on the dropdown as shown in Figure 17.

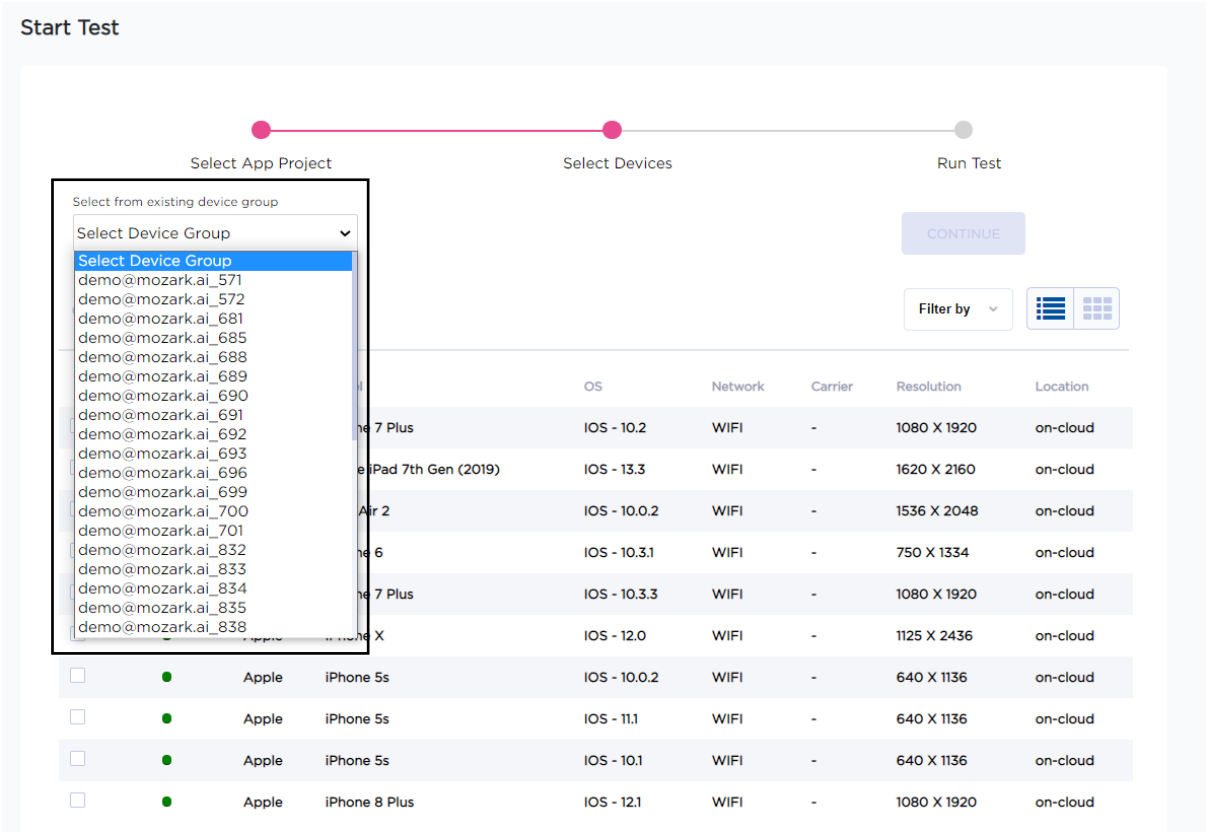

Figure 17: Select Device Group

(b) Select devices

You can also select devices by viewing each device in either List or Grid view, by going to different pages, by putting filters.

a. View

i. List

- Click on  to enable the List view as given in Figure 18.

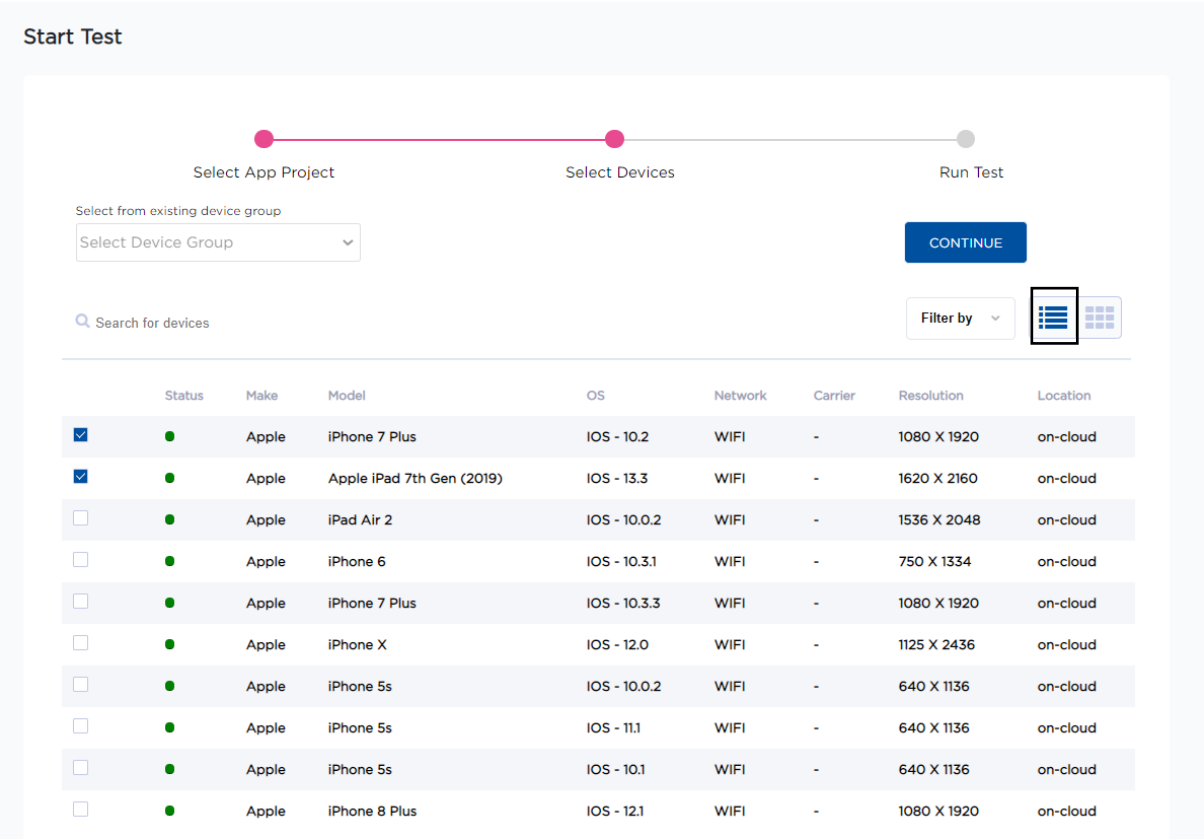

Figure 18: Select Devices in List View

- Select by clicking on the check box in the list view as given in Figure 18

ii. Grid

- Click on to enable the Grid view as given in Figure 19.
- By hovering over the card, you will be able to see the device details as given in Figure 20
- Select by clicking on the check box in the grid view as given in Figure 21

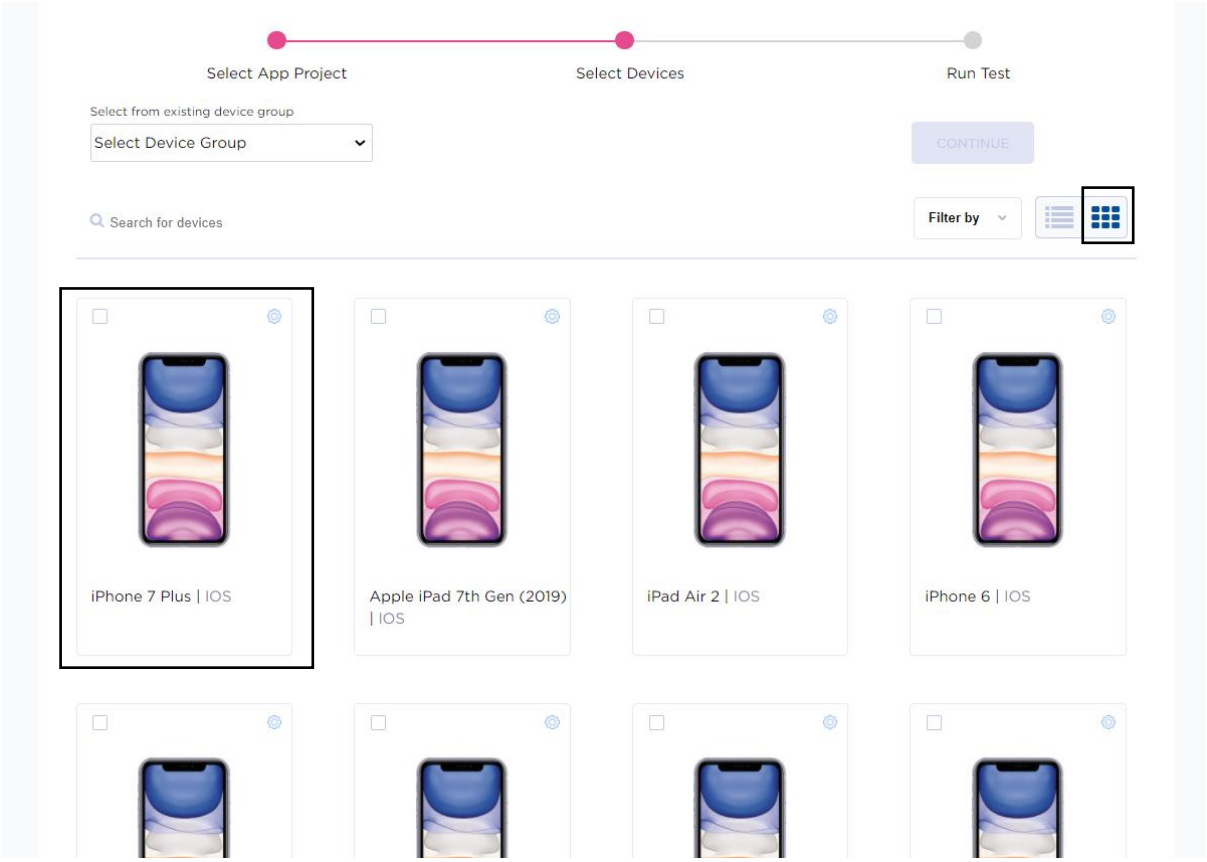

Figure 19: Select Devices in Grid View

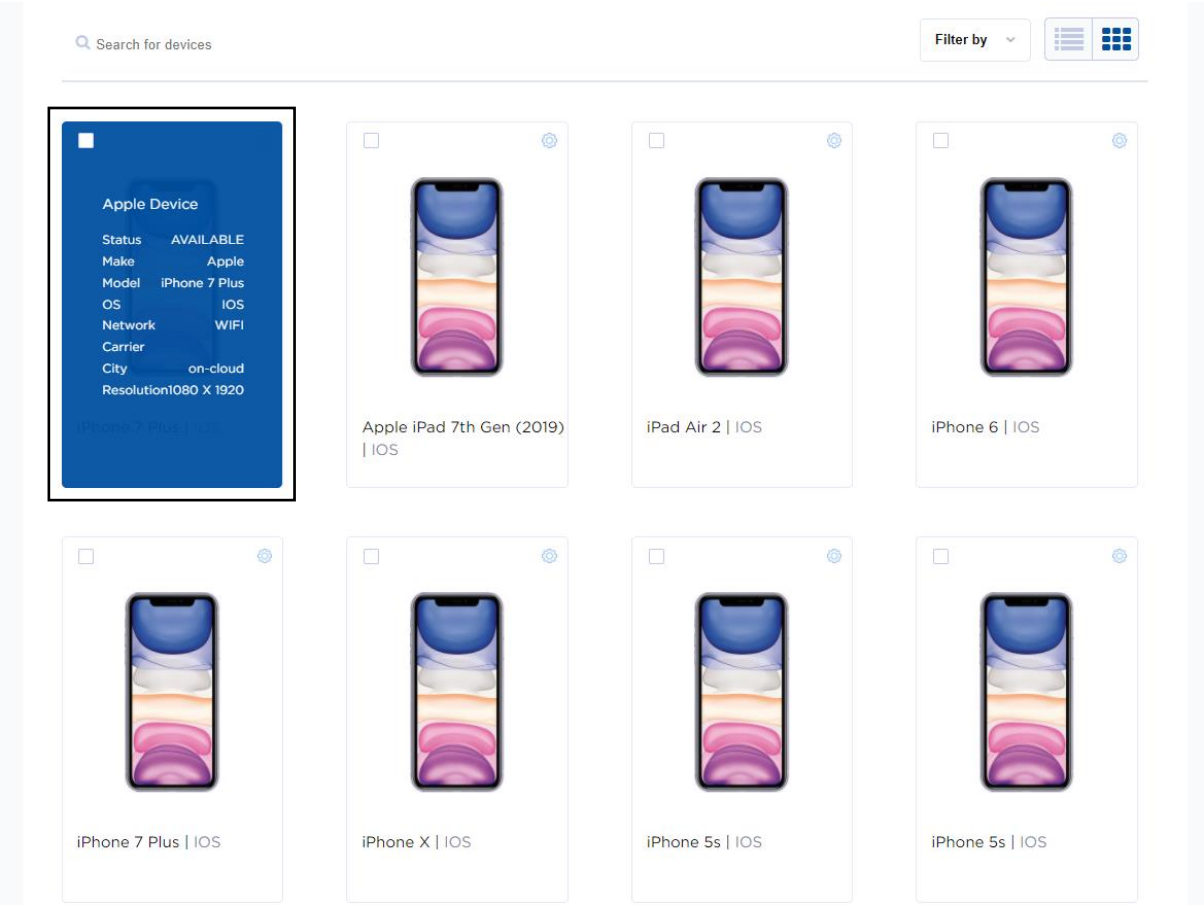

Figure 20: Hover over the device card

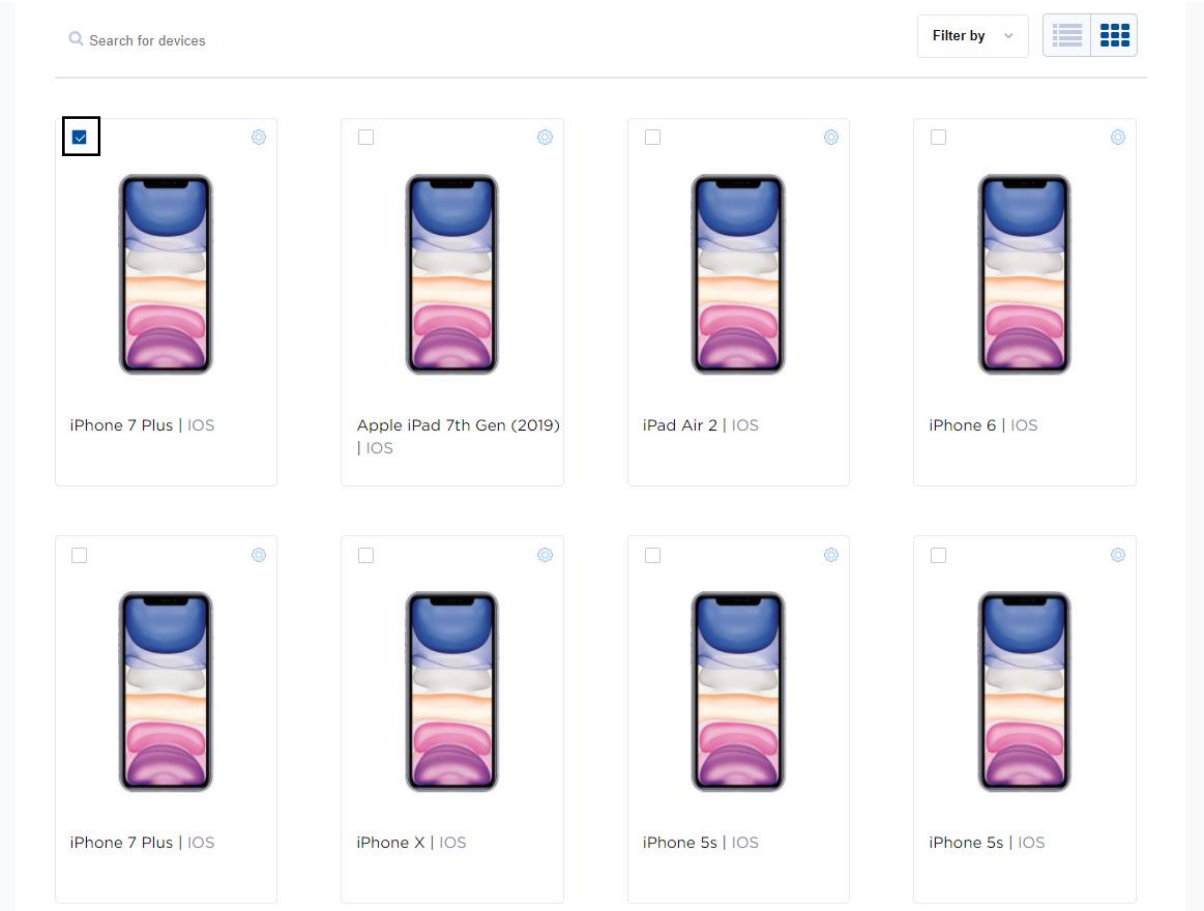

Figure 21: Select Devices in Grid View

b. Pagination

You can browse and select devices by going to different pages by clicking < and > as given in Figures 22 and 23.

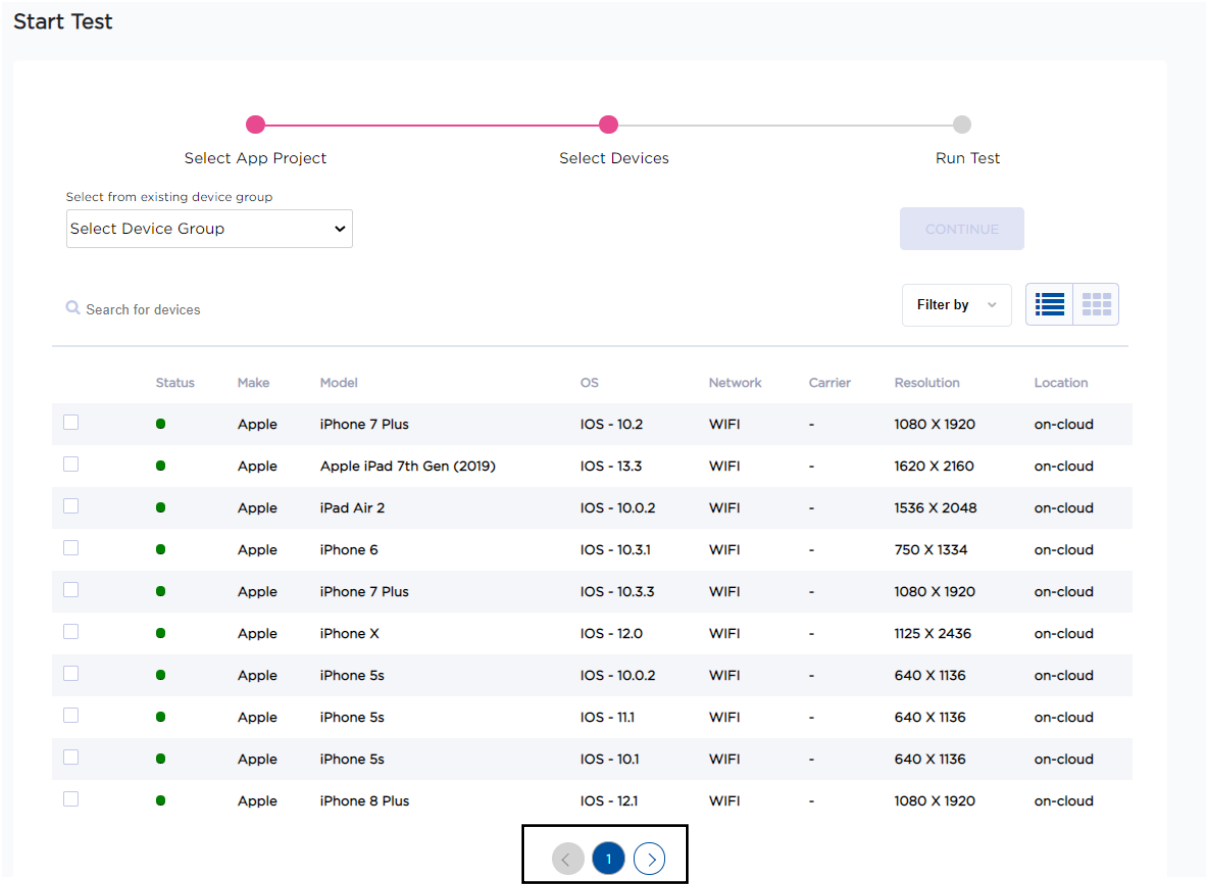

Figure 22: Pagination

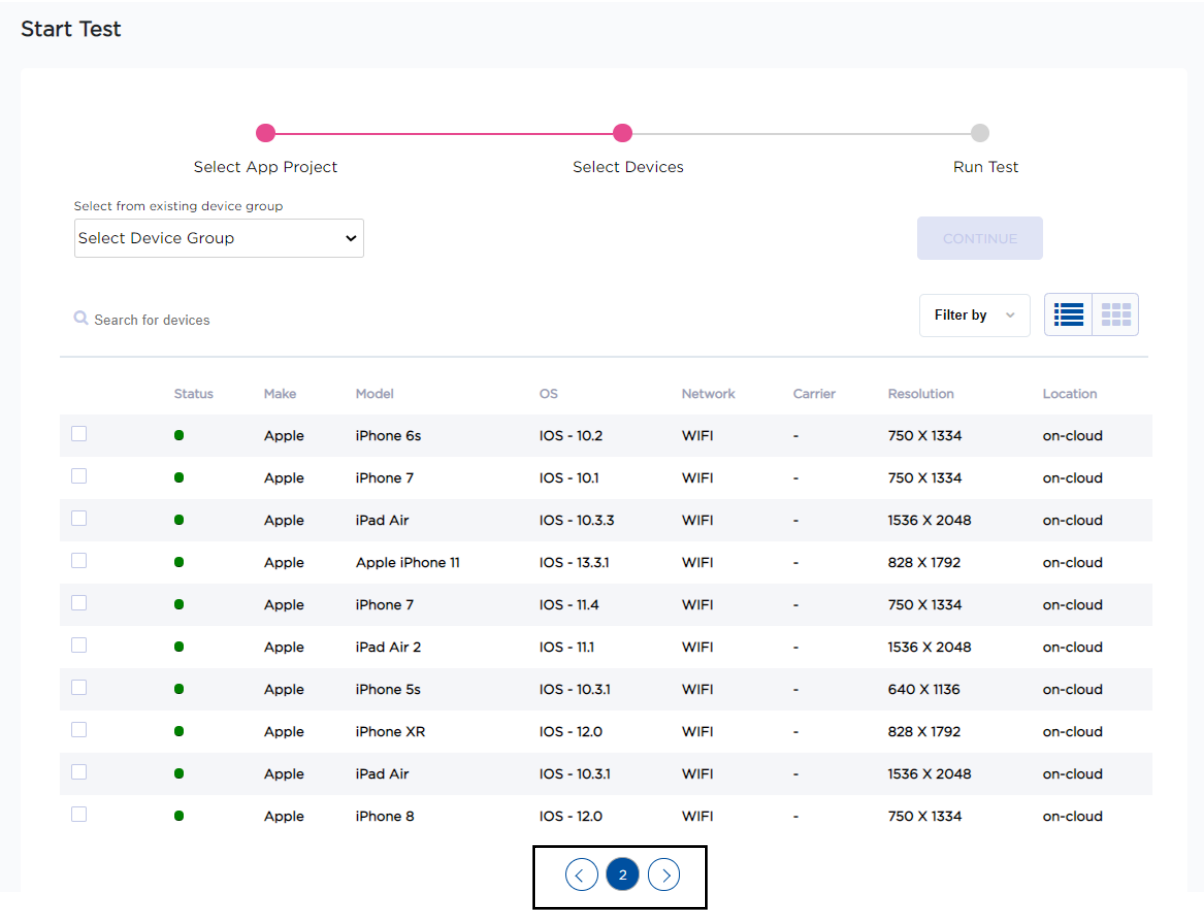

Figure 23: Pagination

c. Filter by

You can select devices further based on filters such as OS, Make, Model, Network, Resolution, Location as shown in Figures 24.

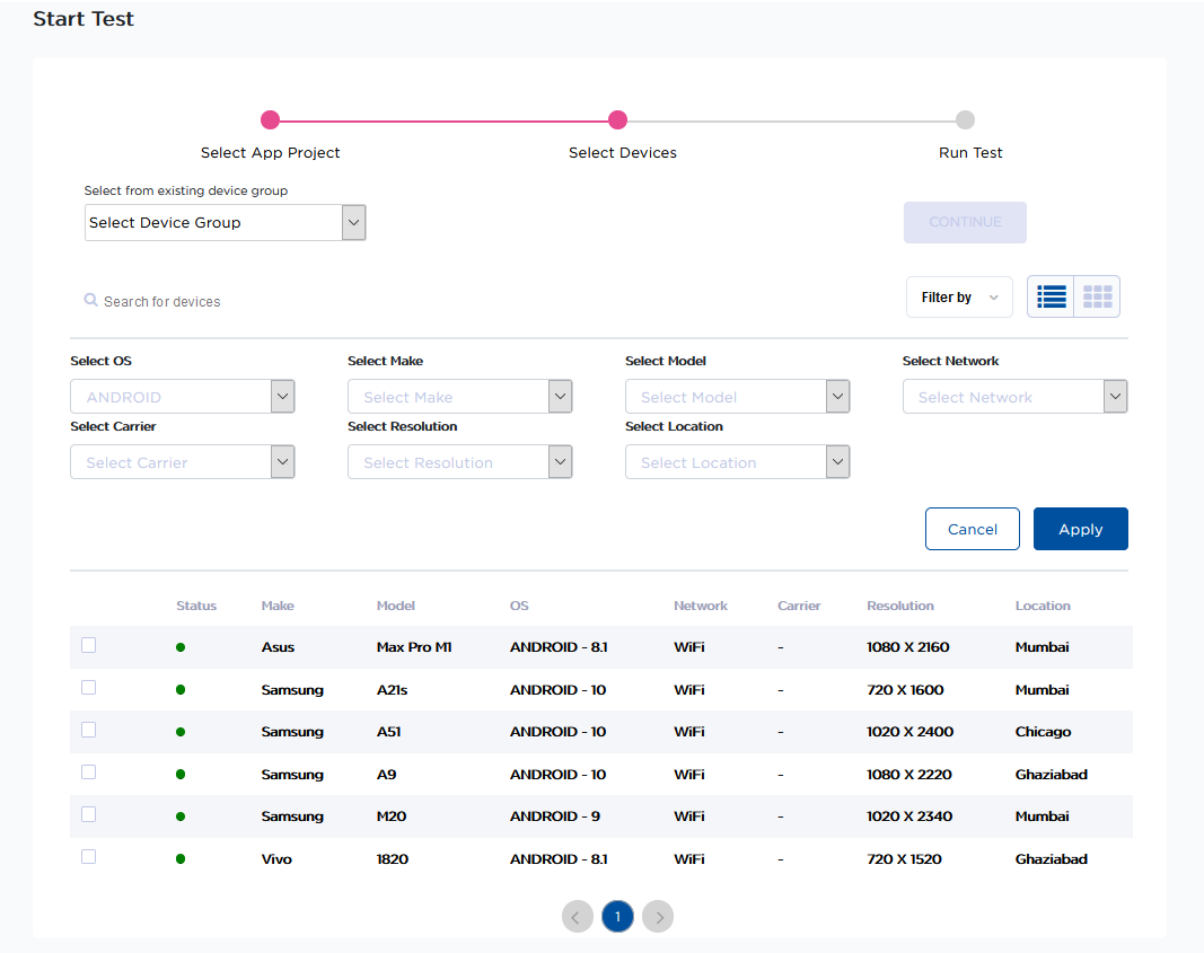

Figure 24: Filter By

d. Remove a selected device

By clicking on **X**, one can remove a selected device before beginning any test run as given in Figure 25.

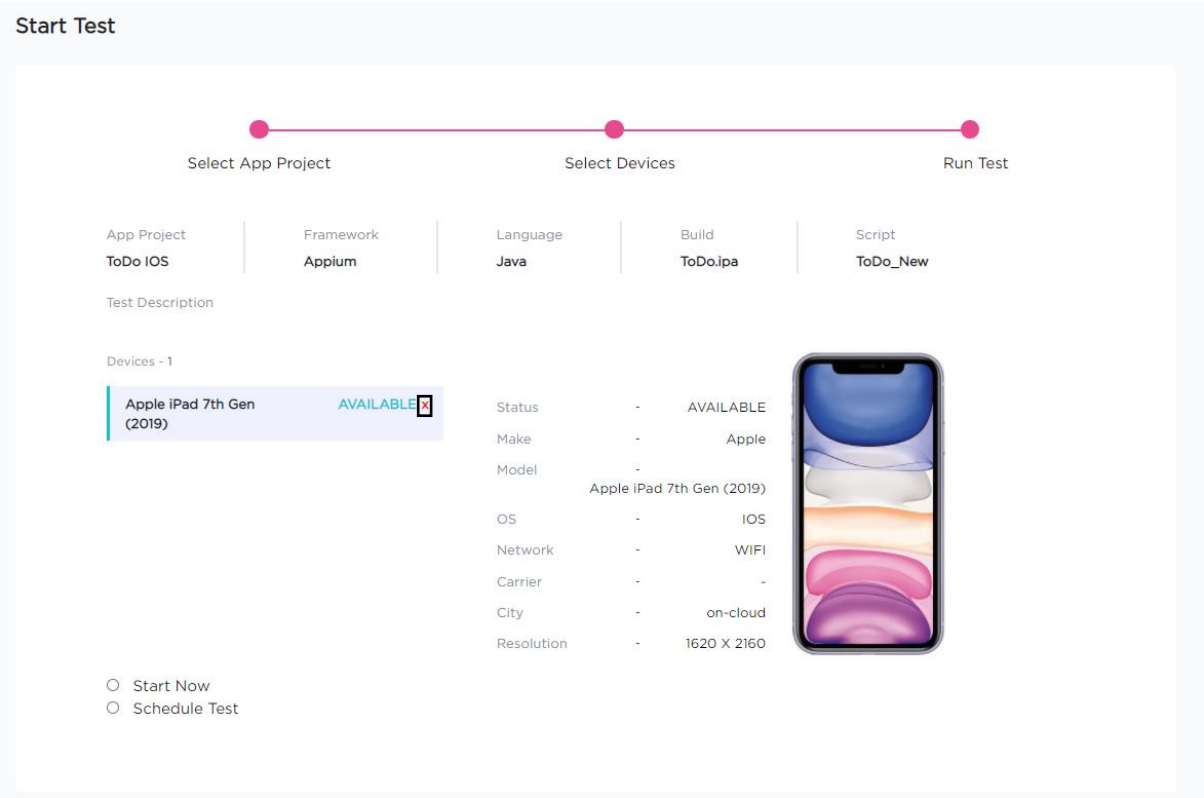

Figure 25: Remove a selected a device

2.8 Run Test

After selecting a project, build, script, devices, you can now run tests. There are two options to run tests. One is Start Now, where you can execute the test now and other one is Schedule Test where you can run continuous tests as shown in Figure 26.

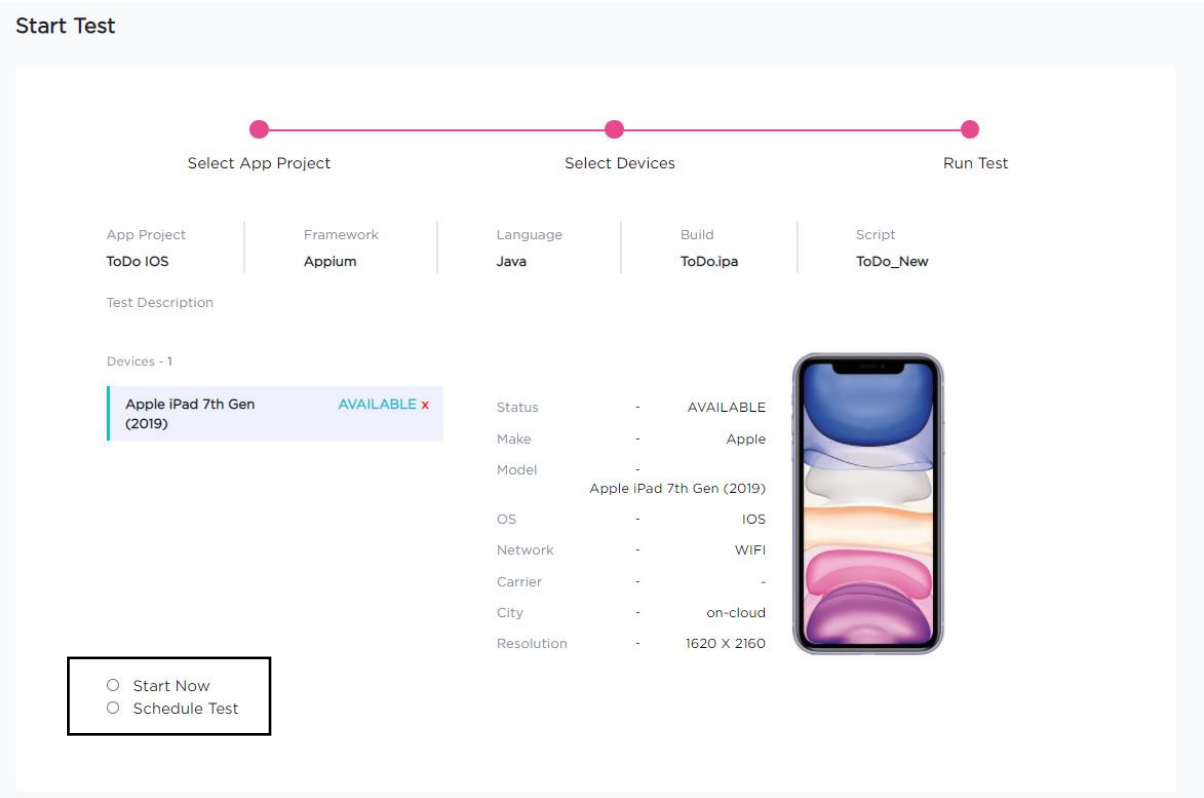

Figure 26: Run Tests - Start Now, Schedule Test

(a) Start Now

When 'Start Now' is selected, then the test starts instantaneously.

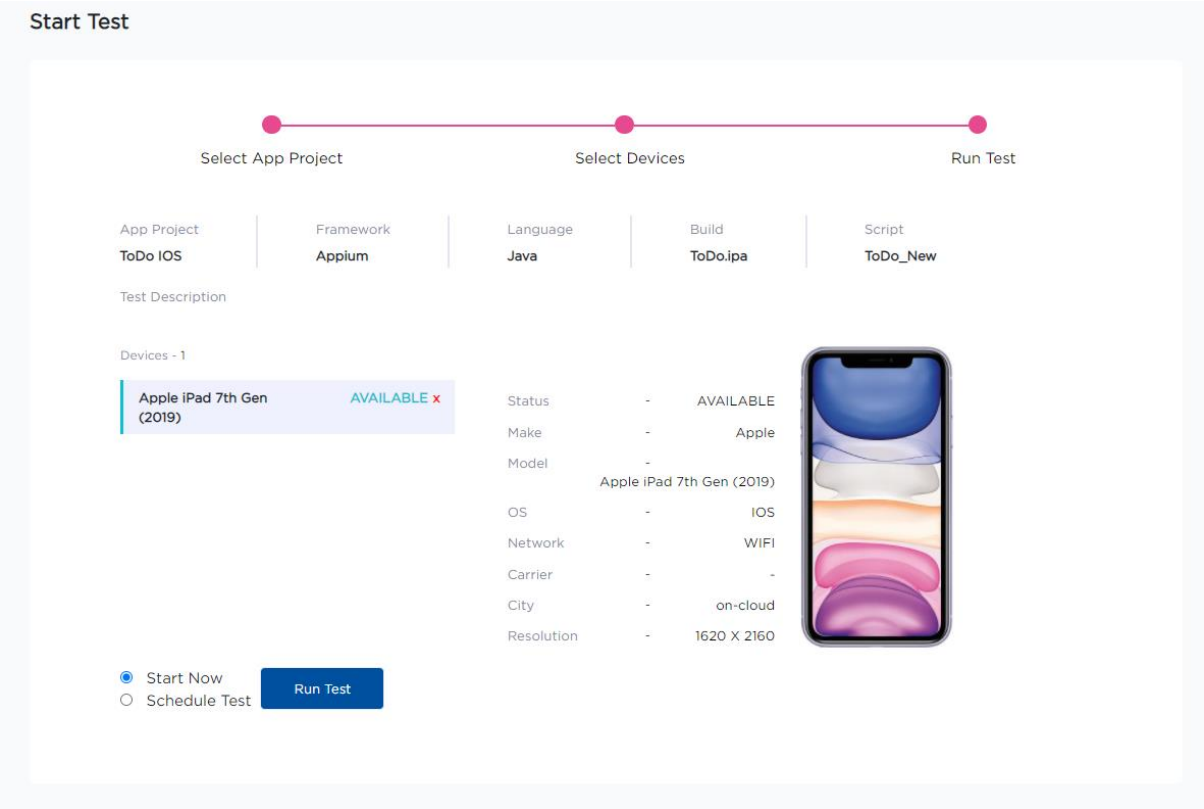

Figure 27: Start Now

(b) Schedule Test

When 'Schedule Test' is selected, then the following fields are required to be populated as shown in Figure 28:

(i) Start Date and Time

(ii) End Date and Time

(iii) Time interval (in minutes)
Put a sufficient interval between two test runs based on test execution time.

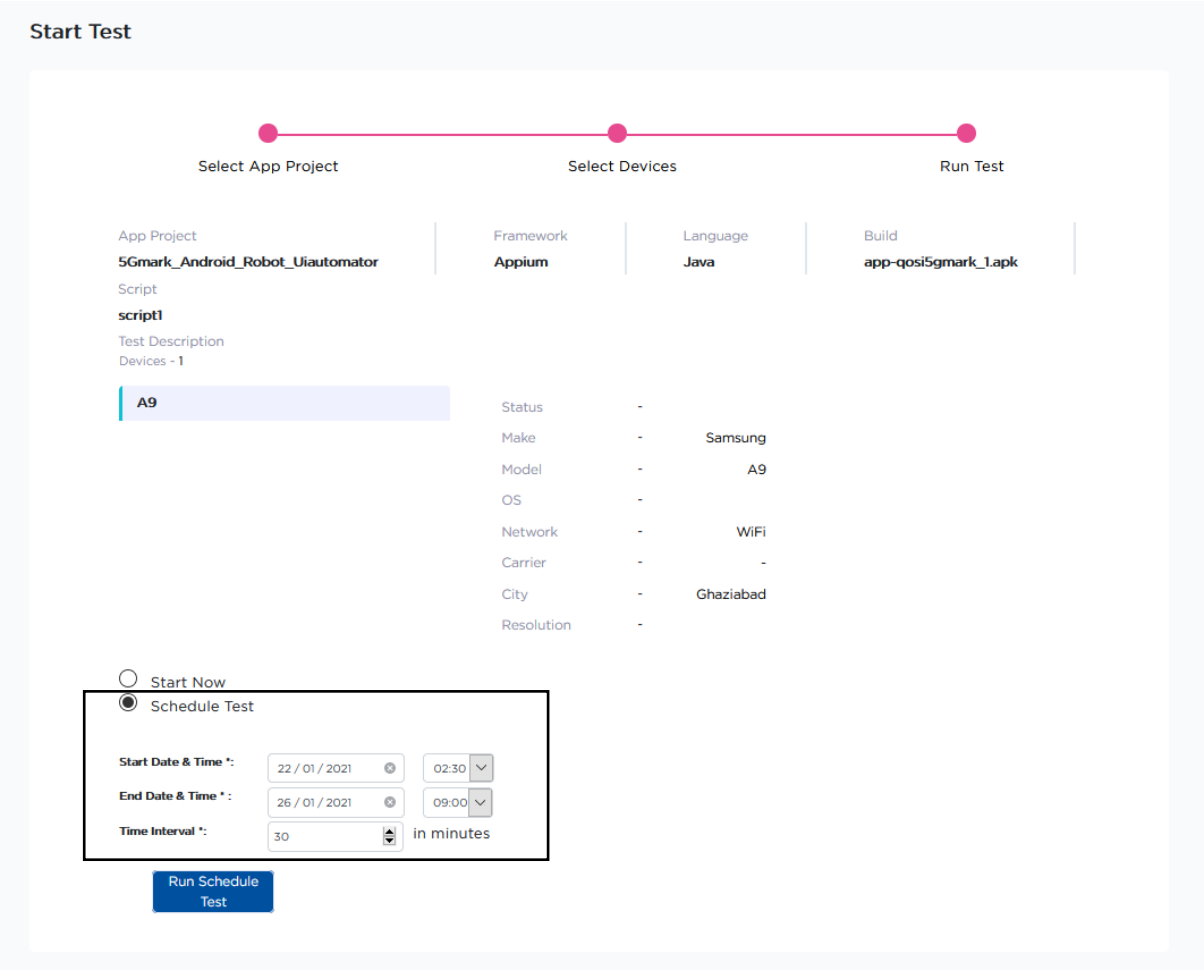

Figure 28: Schedule Test

3. Reviewing Test Results

In this section, we will see on how to check the test results for different set of devices. One can navigate to 'Check Status' and 'See Results' sections to know more about the test results.

3.1 Check Status

3.1.1 Ongoing Test

Ongoing test displays test details such as the status, date & time, device, app project, build, script of the tests as given in Figure 29.

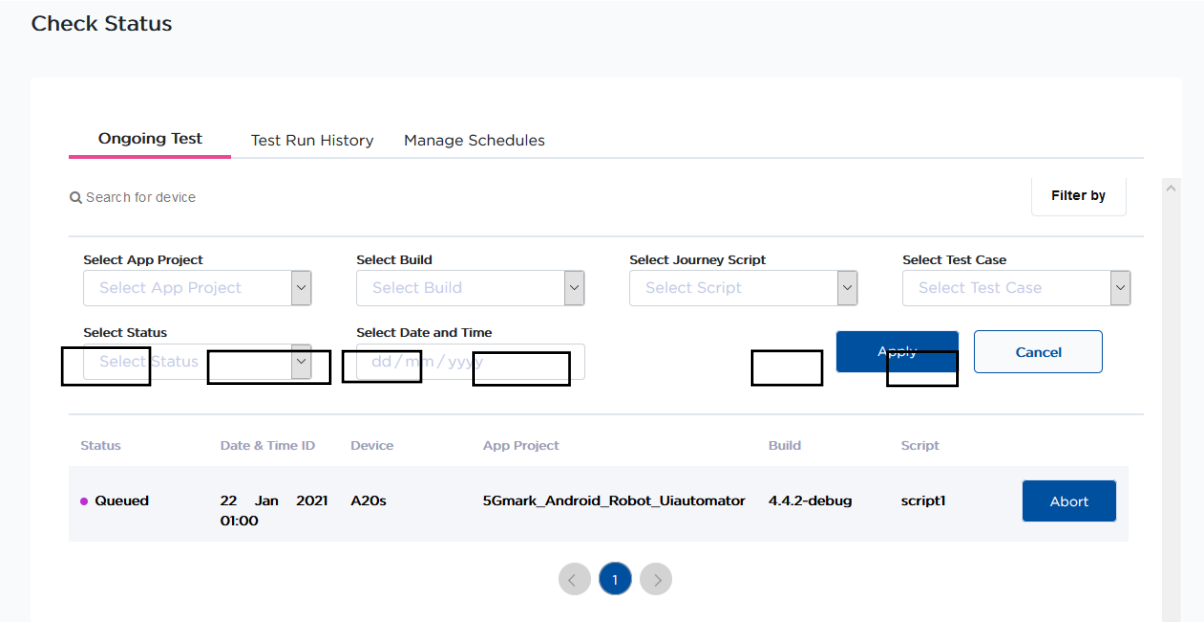

Figure 29: Ongoing Tests

In 'Ongoing Test', you can check for tests by searching for devices, by applying filters such as 'app project', 'build', 'journey script', 'test case', 'status', 'date & time' as given in Figure 30. A 'Filter By' field can be selected by clicking on the dropdown, selecting the required value and clicking 'Apply'.

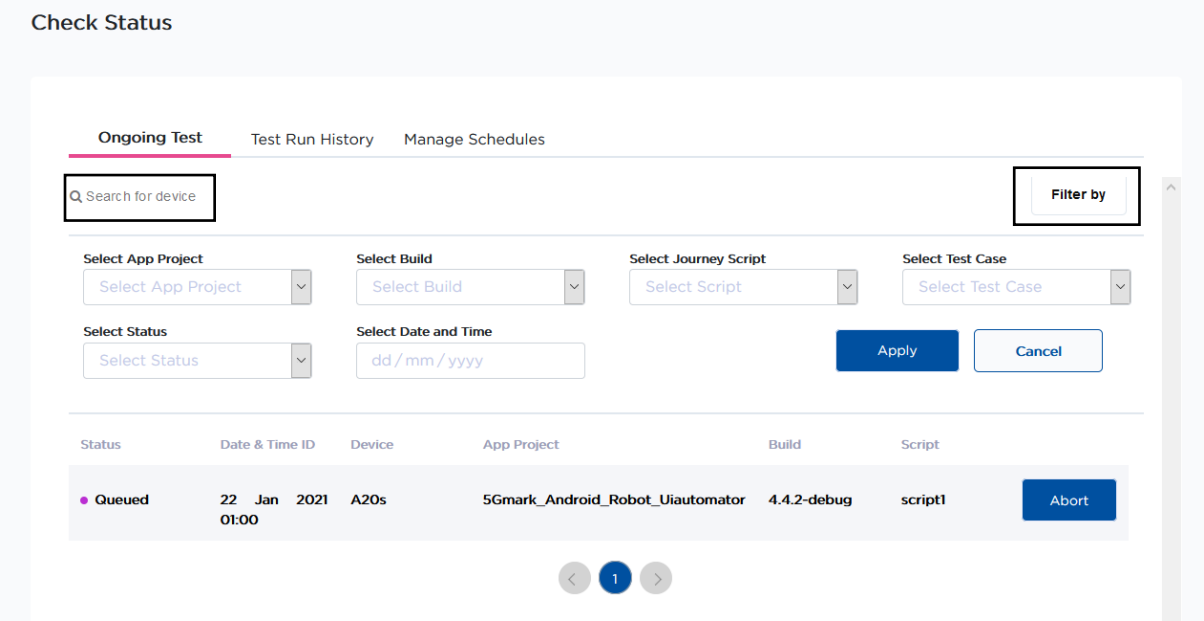

Figure 30: Ongoing Test: 'Search for device' and 'Filter By'

'Select Status' 'Filter' option has following status options:

- Queued – tests that are queued to execute on test devices
- Running – tests that are running on the test devices
- Completed – tests that are executed
- Interrupted – tests that did not complete but were aborted

Ongoing Test also facilitates the cancellation of any previous test runs based on their status by giving an option to Abort the test as given in Figure 31.

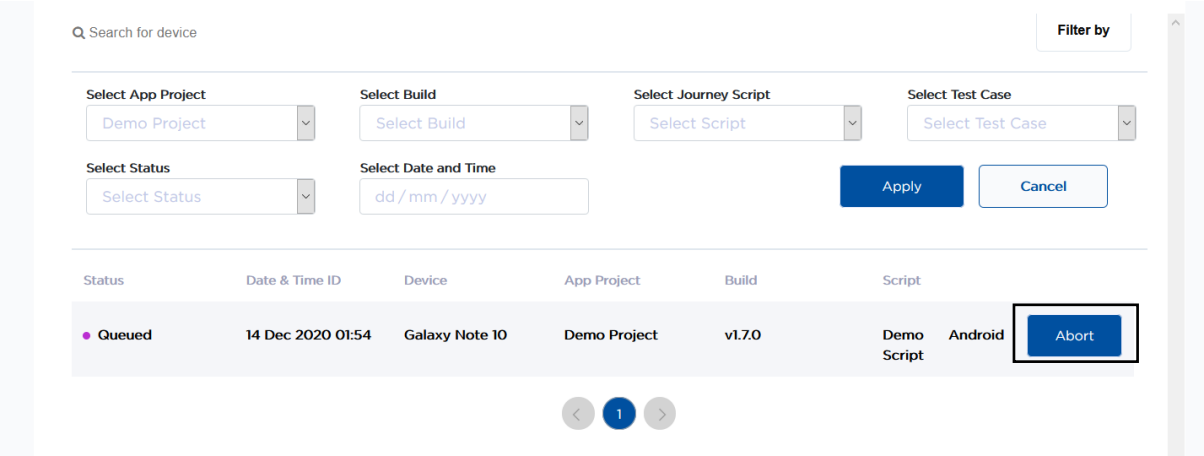

Figure 31: Abort

3.1.2 Test Run History

'Test Run History' displays test details such as date & time, device, app project, build, script and status as shown in Figure 32.

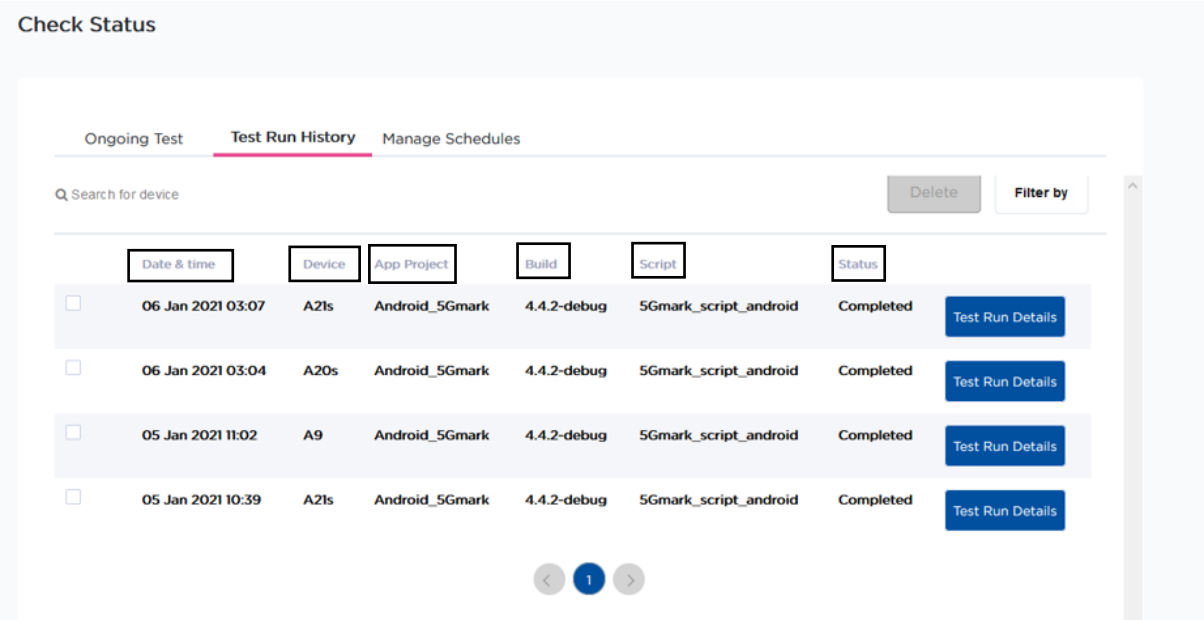

Figure 32: Test Run History

In 'Test Run History', you can check for tests by searching for devices, by applying filters such as 'date & time', 'device', 'app project', 'build', 'script', 'status' as given in Figure 33.

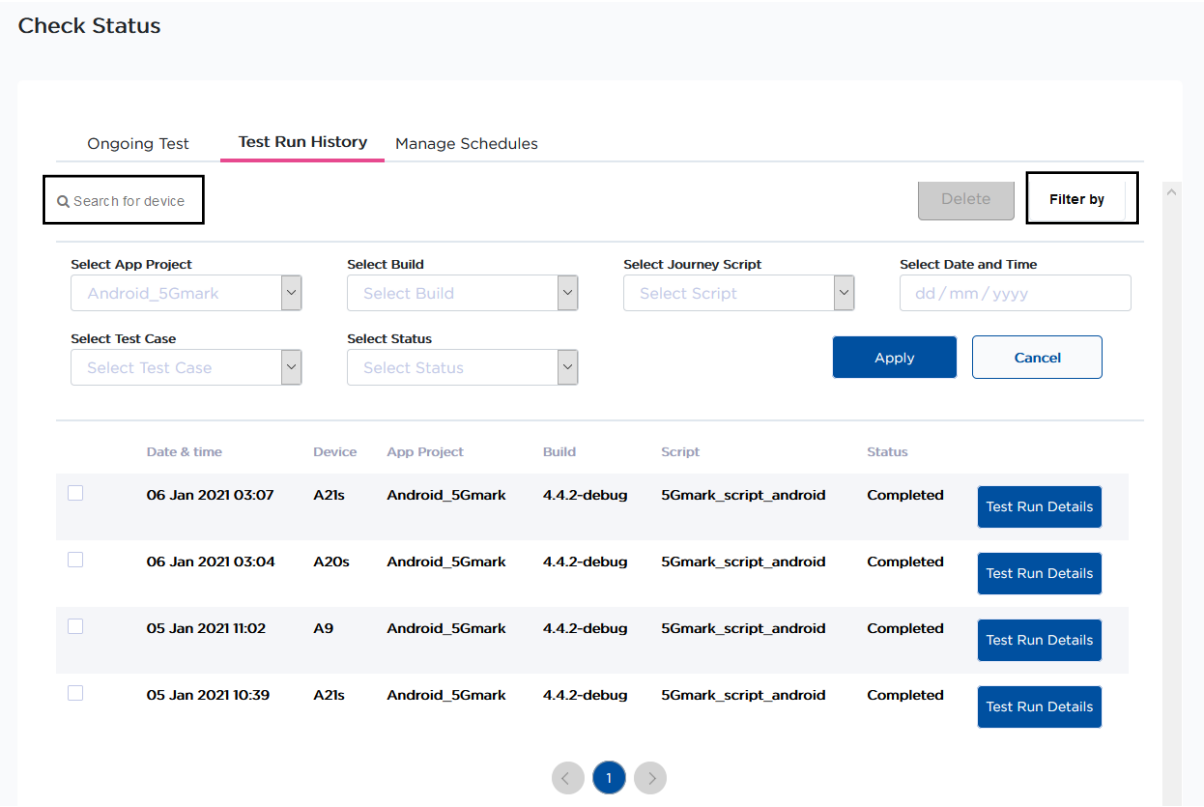

Figure 33: Search for device and Filter By

A 'Filter By' field can be selected by clicking on the dropdown, selecting the required value, and clicking 'Apply' as given in Figure 33.

'Test Run History' also facilitates the deletion of any test run history by selecting the check box and clicking on 'Delete' as given in Figure 34.

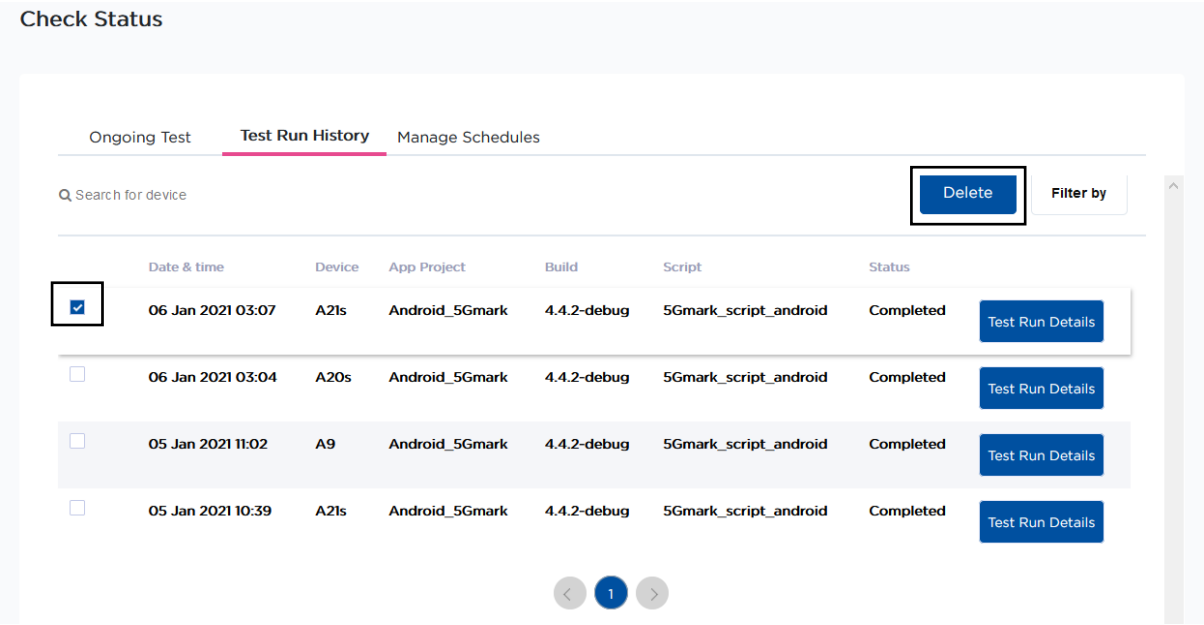

Figure 34: Test Run History – Delete

'Status' column in 'Test Run History' displays the following messages if the tests fail:

- BUILD NOT FOUND ON DEVICE – When the user chooses ‘default’ as the option for specifying build, but no build present in the device.

- DEVICE NOT AVAILABLE – When the device chosen by a user is not available for tests.

- DEVICE GETS DISCONNECTED WHILE TEST IS RUNNING – When the device gets disconnected while the script is executing.

- PLAN IS EXPIRED FOR THAT USER - When the plan purchased by the user is expired.

- MAX RERUN LIMIT REACHED – When the maximum test rerun limit is reached by the user.

By clicking on ‘Test Run Details’, you can get detailed test run information in the form of video recordings, screenshot recordings, test results, test logs, live logs as given in Figure 35.

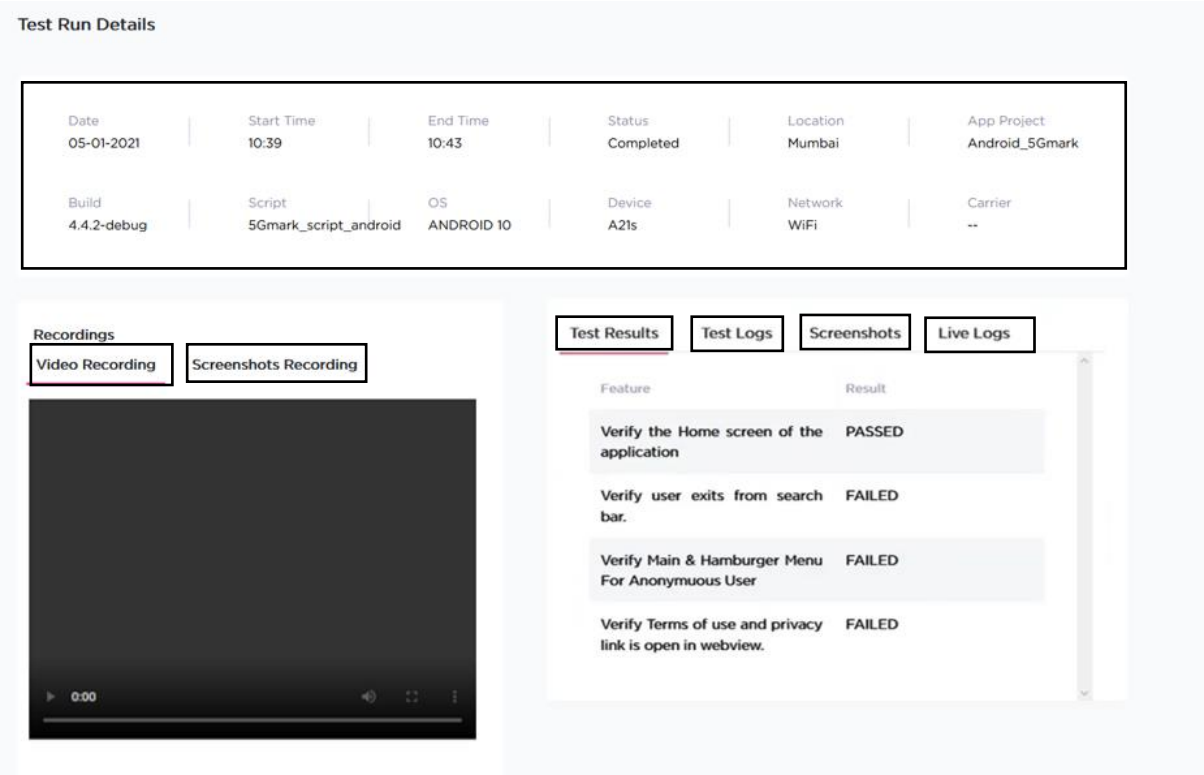

Figure 35: Test Run Details

You can view ‘Screenshot Recording’ by dragging the cursor or clicking on left and right arrows as given in Figure 36. The cursor shows the time stamp of a test run.

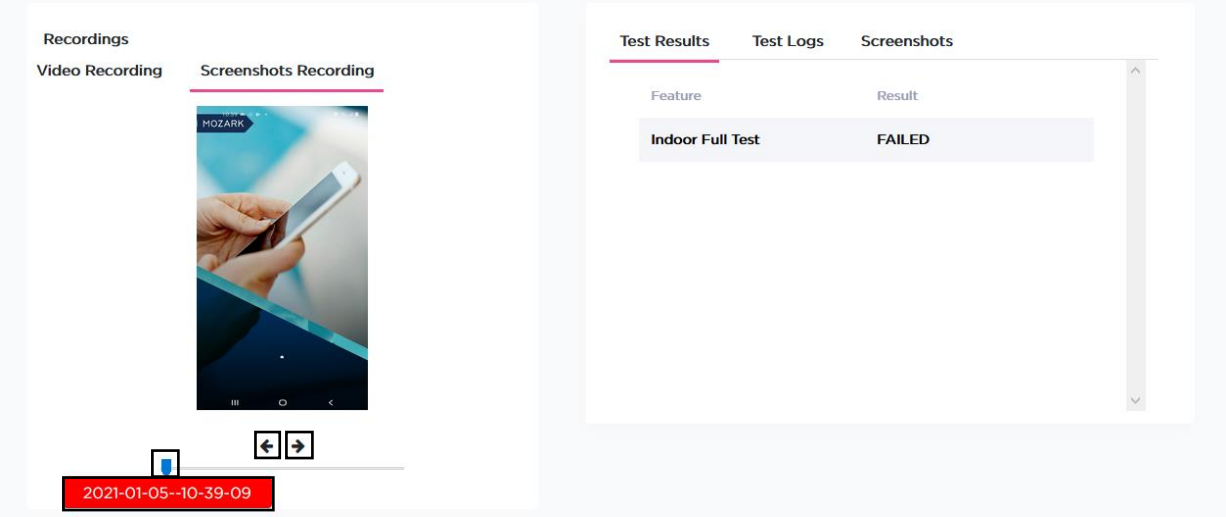

Figure 36: Screenshot Recording

'Test Run Details' show the following results:

- Test Run Details – These are the details about the test runs.
- Video Recording – Video of the tests taken as instructed by the script.
- Screenshot Recording – You can check the screenshot recording by clicking on the arrows given in Figure 36.
- Test Results – The test results such as passed or failed are displayed testcase wise.
- Test Logs – Various logs such as 'Download Logs', 'Download Memory Info Logs', 'Download CPU Info Logs' are generated.
- Screenshots – Snaps of the tests taken as instructed by the script.
- Live Logs – Logs are generated when tests are running.

Please note:

- To record video and capture screenshot, appropriate code should be added to the test script before uploading the script to App Functional. Get in touch with the team for further assistance by sending an email to enquiry@mozark.ai
- For Appium with Java TestNG scripts, you have to import the package org.testng.Assert and call method Assert.assertTrue(true) and Assert.assertTrue(false) to print pass and fail in the test results.

By clicking on 'Rerun Test', you can run the failed testcases as given in Figure 37.

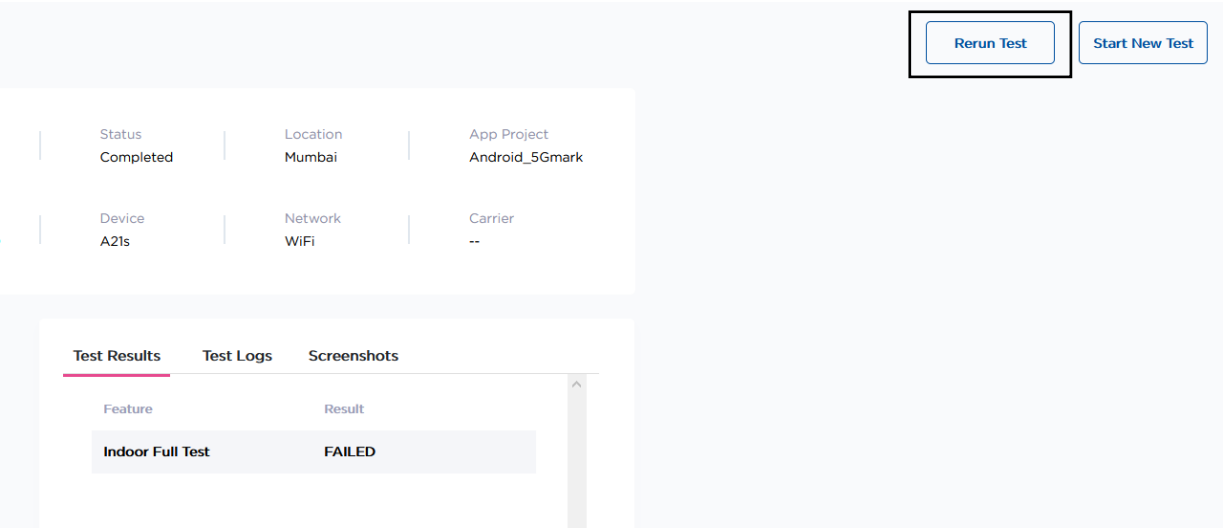

Figure 37: Rerun Test

3.1.3 Manage Schedules

In 'Manage Schedules', you can 'View Tray', 'Edit Schedule', 'Delete' continuous tests scheduled by you as given in Figure 38.

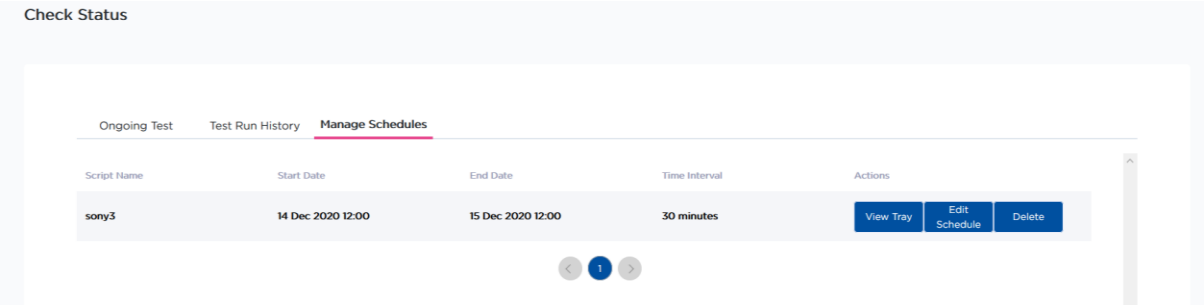

Figure 38: Manage Schedules

(i) View Tray

Click 'View Tray' in Figure 38 to view the devices in that tray as given in Figure 39.

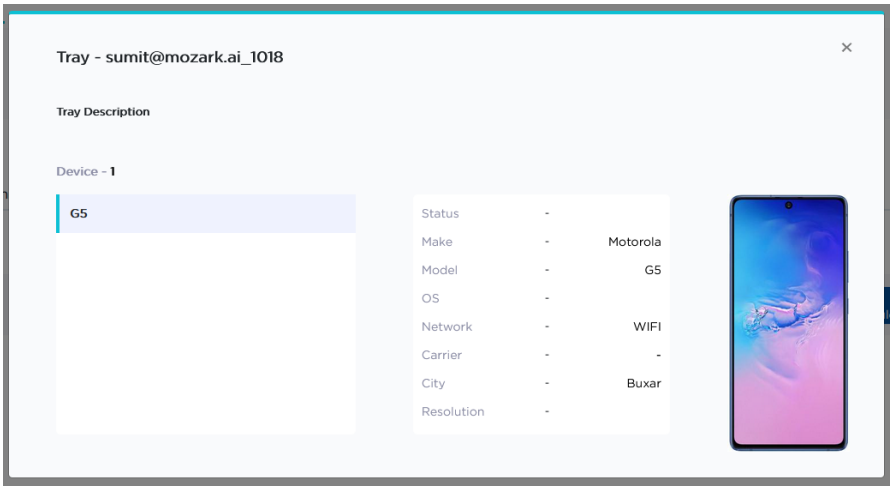

Figure 39: View Tray

(ii) Update Schedule Test

Click 'Update Schedule Test' in Figure 38 to update 'start date & time, 'end date & time', 'time interval' for your continuous tests as given in Figure 40.

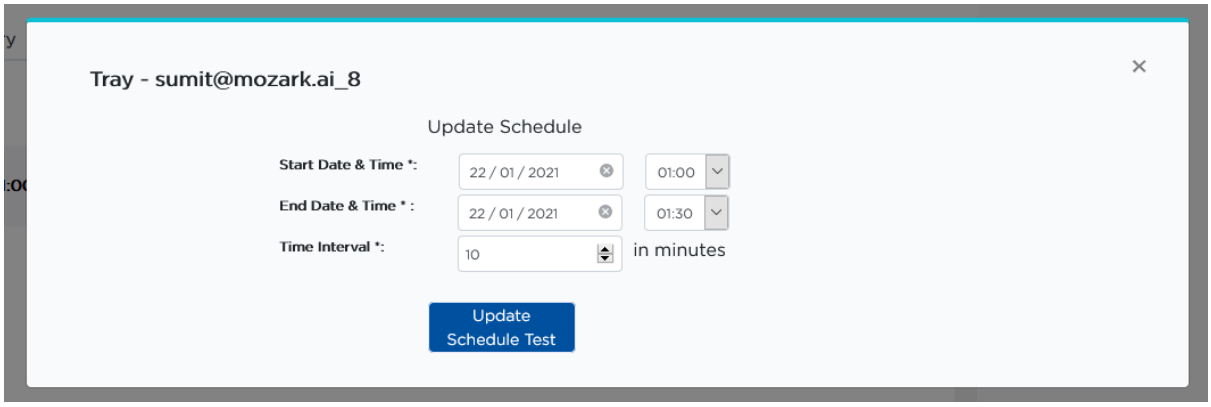

Figure 40: Update Schedule Test

(iii) Delete

Click 'Delete' in Figure 38 to delete a test schedule as given in Figure 41.

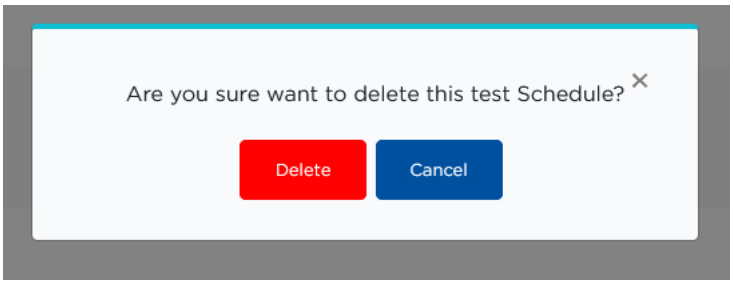

Figure 41: Delete

3.2 See Results

In ‘See Results’, you can get insights from the test runs by using ‘Set Filters’, ‘Choose X-Axis’, ‘Choose Chart Type’, and ‘Advanced Dashboard’ as given in Figure 42.

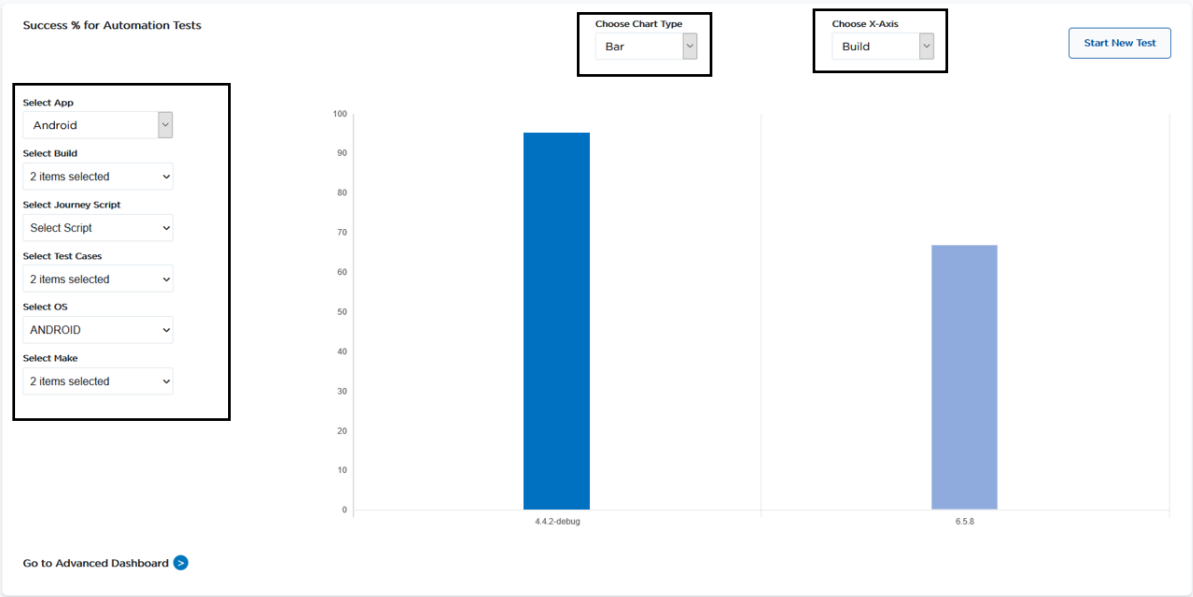

Figure 42: See Results

Set Filters such as ‘Select App’, ‘Select Build’, ‘Select Journey Script’, ‘Select Test Cases’ to customize the insights from the test runs. In Figure 43, different test cases are selected by clicking the check boxes to see results for the selected test cases.

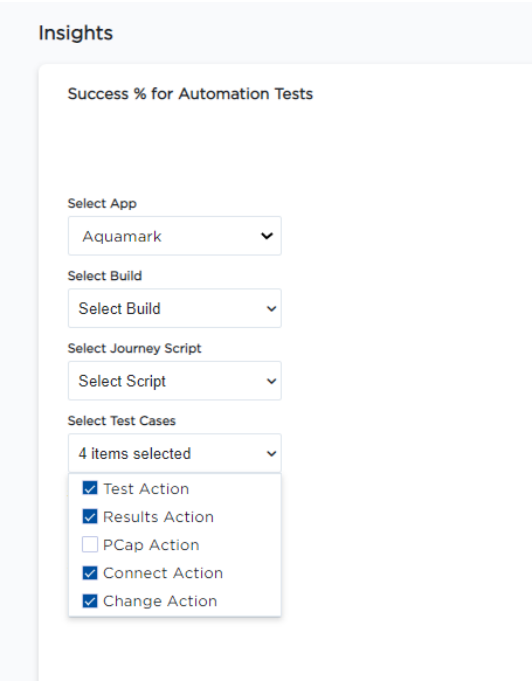

Figure 43: Set Filters

Click the dropdown given for ‘Choose X-Axis’ to select ‘Build’, ‘Script’, ‘Testcase’’, ‘OS’, ‘Make’ to customize the insights from the test runs. For example, a testcase is selected by clicking the dropdown to see results for the given testcases.

Click the dropdown given for ‘Choose Chart Type’ to select ‘Bar’, ‘Line’ to customize the insights from the test runs.

Click ‘Advanced Dashboard’ in Figure 44 to view test run details. In Figure 44, you can apply filters to check the test run details for filtered test runs.

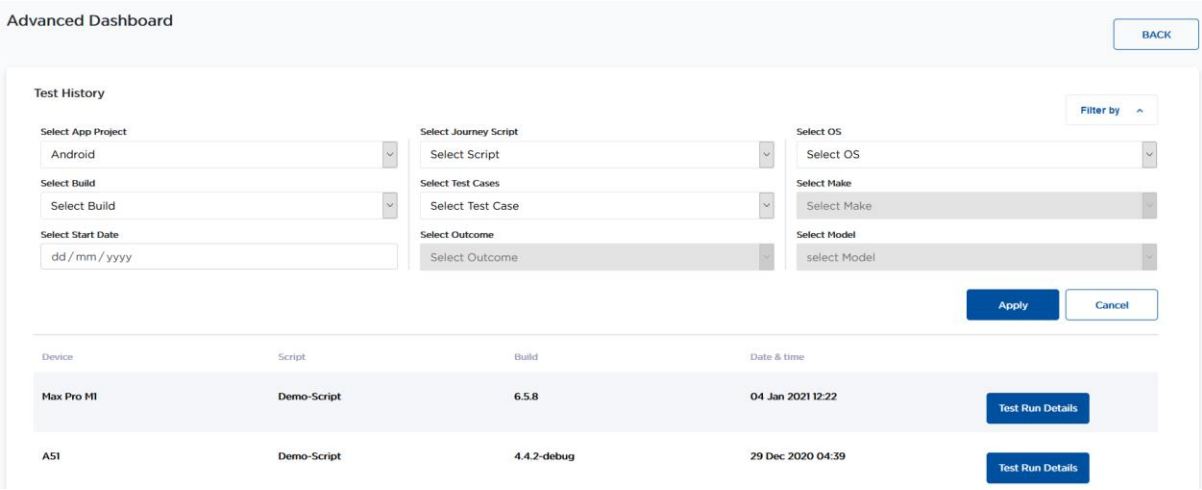

Figure 44: Advanced Dashboard

4. Scripting Guidelines

The scripting guideline is to help users write/modify their automation scripts to be compatible with MOZARK App Functional platform. The Language-Framework supported on the platform are:

1. Robot-Appium with Python

2. Robot-UIAutomator with Python

3. Appium with Java TestNG

4.1 Robot-Appium

This section describes how to configure, package, and upload your Robot-Appium tests to App Functional.

Please refer to our sample build and sample scripts for Robot-Appium with Python -

Android(Build): https://bitbucket.org/mozarkai/mozark-python-appium-robot-tests-android-5gmark/src/master/

iOS(Build): https://bitbucket.org/mozarkai/mozark-python-appium-robot-tests-ios-5gmark/src/master/

Android(Mobile Website): https://bitbucket.org/mozarkai/mozark-python-appium-robot-tests-android-mweb-5gmark/src/master/

iOS(Mobile Website): https://bitbucket.org/mozarkai/mozark-python-appium-robot-tests-ios-mweb-5gmark/src/master/

Use the below instructions to get started with Robot-Appium scripting on App Functional.

1. Ensure that the Test Package conforms to below folder structure

You will have to create a particular folder structure for your test package. Refer to our sample folder given in Figure 45. This folder structure will ensure that all the dependencies and packages are available.

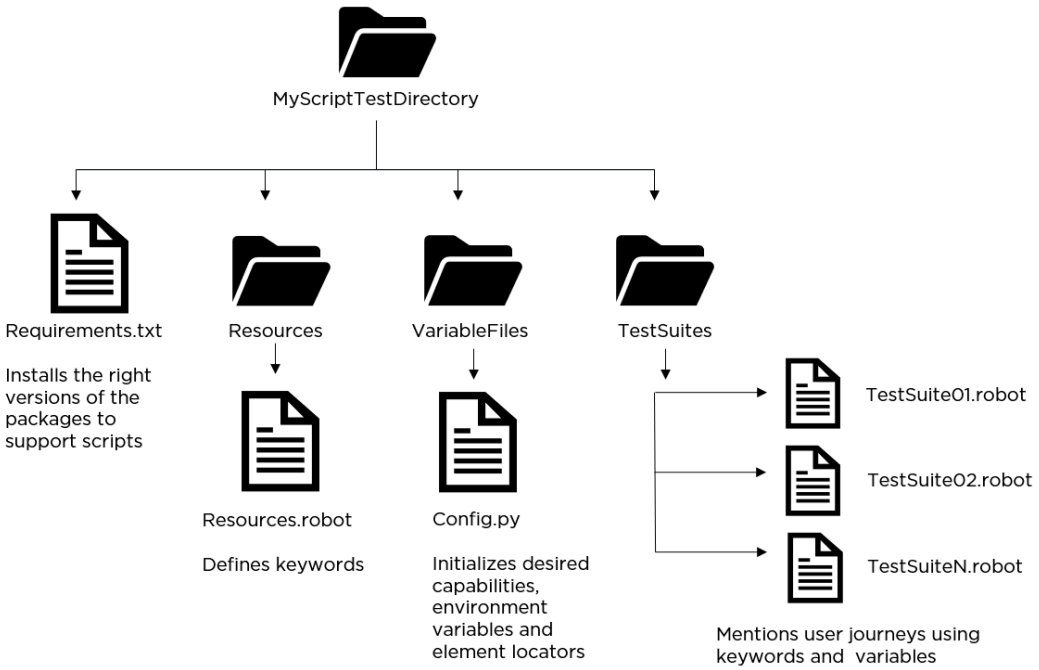

Figure 45: Folder Structure

Please note:

- To ensure the appropriate setup for your Robot-Appium scripts while uploading your test package on App Functional, generate requirements.txt and bundle it inside your test directory/package as given in Figure 45. You can run the below command to generate requirements.txt:

```
pip freeze > requirements.txt
```

The following code snippet is a glimpse of requirements.txt:

```
robotframework-appiumlibrary==1.5.0.7
  - Appium-Python-Client [required: >=0.28, installed: 1.0.2]
```

```
    - selenium [required: >=3.14.1,<4, installed: 3.141.0]
      - urllib3 [required: Any, installed: 1.26.2]
  - decorator [required: >=3.3.2, installed: 4.4.2]
  - docutils [required: >=0.8.1, installed: 0.16]
  - kitchen [required: >=1.2.4, installed: 1.2.6]
  - robotframework [required: >=2.6.0, installed: 3.2.2]
  - selenium [required: >=2.47.1, installed: 3.141.0]
    - urllib3 [required: Any, installed: 1.26.2]
  - six [required: >=1.10.0, installed: 1.15.0]
setuptools==49.2.1
```

2. Configure your test package

Replace all hard-coded variables in the config.py file, such as Device UDID, Appium Server IP etc with environment variables, to ensure compatibility. Environment variables represent values that are used by your automated tests. App Functional dynamically populates environment variables at runtime by giving the values to the required environment variables.

- Go to MyScriptTestDirectory/VariableFiles/Config.py
- Add code snippet- import os
- Update the below capabilities with the mentioned environment variables

Capabilities	Description	Values
Udid	Unique device identifier for the device under test	os.environ['DEVICE_SERIAL_ID']
platformVersion	Mobile OS version for the device under test	os.environ['DEVICE_OS_VERSION']
appiumServer	Appium server IP address	os.environ['APPIUM_SERVER']
chromedriverPort	Port number assigned to chrome browser application. This is only required for tests on the default browsers for mobile websites.	os.environ['CHROME_PORT']
systemPort	To connect with Appium server. This is only required for tests on the default browsers for mobile websites.	os.environ['SYSTEM_PORT']

- Add supported environment variables or desired capabilities(Supported variables are a set of desired capabilities that can be used while writing testcases.)

Capabilities	Description	Values
deviceName	Model name of the device	os.environ['DEVICE_NAME']
phoneNumber	Phone number of the device. This may be used to verify OTP flows.	os.environ['DEVICE_PHONE_NUMBER']
mjpegPort	To take screenshots with the script.	os.environ['MJPEG_PORT']

Please note:

- Put username and password for GMAIL login, Facebook login as part of the script if there is any testcase that requires such credentials.
- Rectify all the build errors and missing dependencies before bundling and uploading your test package to App Functional.

3. Upload the test package(scripts) to App Functional

- Sign in to App Functional console: http://demo-appfunctional.mozark.ai/
- Create or select a project. Follow the instructions on how to create or select a project.
- Upload or select a build. Follow the instructions on how to create or select a project.
- Bundle your test package
- Upload the test package. Follow the instructions on how to upload a test script.

Please refer to our sample build and sample scripts for Robot-Appium with Python -

Android(Build): https://bitbucket.org/mozarkai/mozark-python-appium-robot-tests-android-5gmark/src/master/

iOS(Build): https://bitbucket.org/mozarkai/mozark-python-appium-robot-tests-ios-5gmark/src/master/

Android(Mobile Website): https://bitbucket.org/mozarkai/mozark-python-appium-robot-tests-android-mweb-5gmark/src/master/

iOS(Mobile Website): https://bitbucket.org/mozarkai/mozark-python-appium-robot-tests-ios-mweb-5gmark/src/master/

4.2 Robot-UIAutomator

This section describes how to configure, package, and upload your Robot-UIAutomator tests to App Functional. The UIAutomator testing framework provides a set of APIs to build user interface tests that perform interactions on user and system apps for Android.

Please refer to our sample build and sample scripts for Robot-UIAutomator with Python for Android(Build): https://bitbucket.org/mozarkai/mozark-python-uiautomator-robot-tests-android-5gmark/src/master/

Use the below instructions to get started with Robot-UIAutomator scripting on App Functional.

1. Ensure that the Test Package conforms to below folder structure

You will have to create a particular folder structure for your test package. Refer to our sample folder given in Figure 46. This folder structure will ensure that all the dependencies and packages are being taken care of.

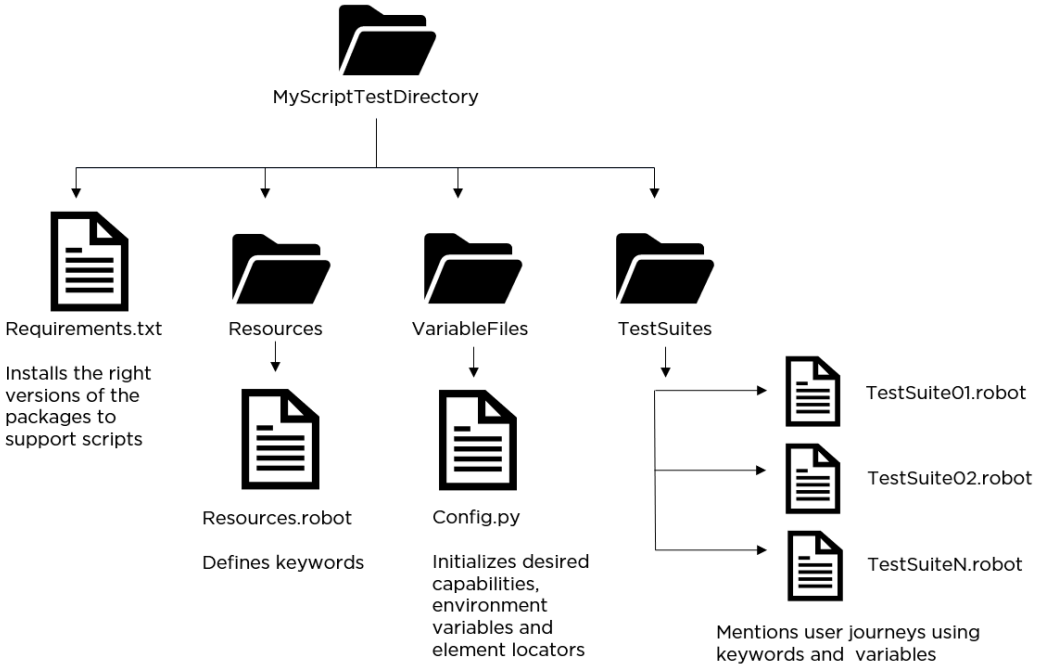

Figure 46: Folder Structure

Please note:

- To ensure the appropriate setup for your Robot-Appium scripts while uploading your test package on App Functional, generate requirements.txt and bundle it inside your test directory/package as given in Figure 46. You can run the below command:

```
pip freeze > requirements.txt
```

The following code snippet is a sample of requirements.txt:

```
robotframework-uiautomatorlibrary==0.4
  - uiautomator [required: >=0.1.30, installed: 1.0.2]
    - urllib3 [required: >=1.7.1, installed: 1.26.2]
setuptools==49.2.1
```

2. Configure your test package

Replace all hard-coded variables in the config.py file, such as Device UDID, Appium Server IP etc with environment variables, to ensure compatibility. Environment variables represent values that are used by your automated tests. App Functional dynamically populates environment variables at runtime by giving the values to the required environment variables.

- Go to MyScriptTestDirectory/VariableFiles/Config.py
- Add code snippet- `import os`
- Update the below capabilities with the mentioned environment variables

Capabilities	Description	Values

udid	Unique device identifier for the device under test	os.environ['DEVICE_SERIAL_ID']
platformVersion	Mobile OS version for the device under test	os.environ['DEVICE_OS_VERSION']

- Add supported environment variables or desired capabilities(Supported variables are a set of desired capabilities that can be used while writing testcases.)

Capabilities	Description	Values
deviceName	Model name of the device	os.environ['DEVICE_NAME']
phoneNumber	Phone number of the device. This may be used to verify OTP flows.	os.environ['DEVICE_PHONE_NUMBER']
mjpegPort	To take screenshots with the script.	os.environ['MJPEG_PORT']

Please note:

- Put username and password for GMAIL login, Facebook login as part of the script if there is any testcase that requires such credentials.
- Rectify all the build errors and missing dependencies before bundling and uploading your test package to App Functional.

3. Upload the test package(scripts) to App Functional

- Sign in to App Functional console: http://demo-appfunctional.mozark.ai/
- Create or select a project. Follow the instructions on how to create or select a project.
- Upload or select a build. Follow the instructions on how to create or select a project.
- Bundle your test package
- Upload the test package. Follow the instructions on how to upload a test script.

Please refer to our sample build and sample scripts for Robot-UIAutomator with Python for Android(Build): https://bitbucket.org/mozarkai/mozark-python-uiautomator-robot-tests-android-5gmark/src/master/

4.3 Appium Java TestNG

This section describes how to configure, package, and upload your Appium tests to App Functional. Appium is an open-source tool for automating native and mobile web applications. For more information, see Introduction to Appium on the Appium website.

Please refer to our sample build and sample scripts for Appium with Java TestNG

Android(Build): https://bitbucket.org/mozarkai/mozark-java-appium-testng-tests-android-5gmark/src/master/

iOS(Build): https://bitbucket.org/mozarkai/mozark-java-appium-testng-tests-ios-5gmark/src/master/

1. Ensure that the Test Package conforms to below folder structure

You will have to create a particular folder structure for your test package. Refer to our sample folder given in Figure 47. This folder structure will ensure that all the dependencies and packages are available.

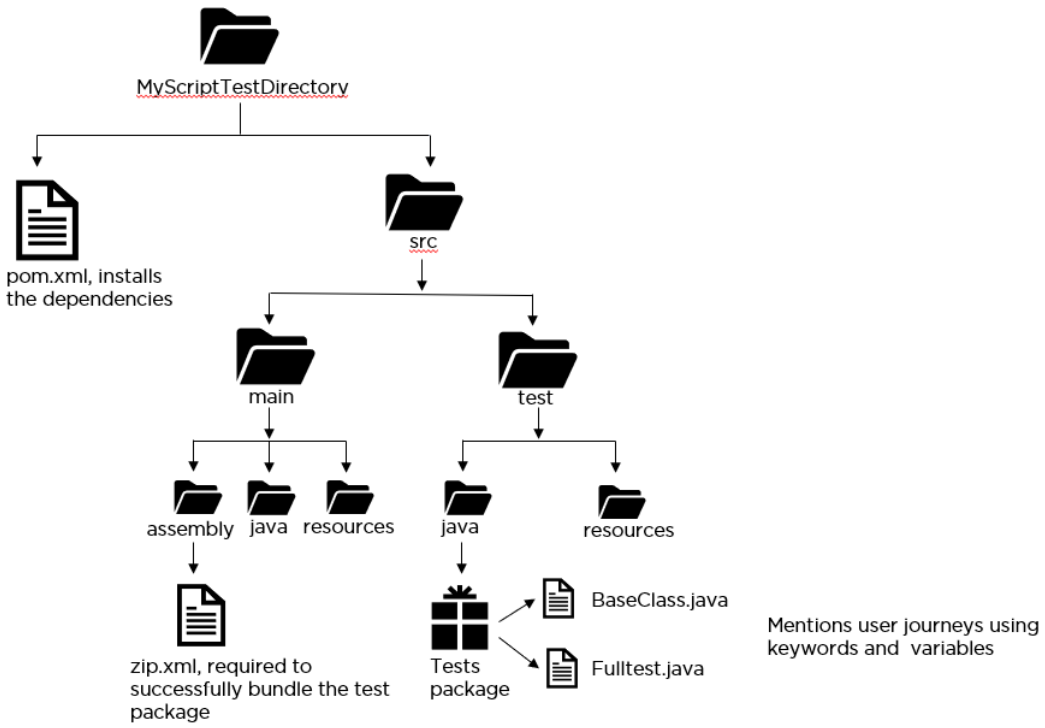

Figure 47: Folder Structure for Appium with Java TestNG Scripts

Please note:

- Add Maven dependencies and builds to pom.xml as given in the folder structure in Figure 47.

```
<dependencies>

<!-- https://mvnrepository.com/artifact/io.appium/java-client -->
<dependency>
        <groupId>io.appium</groupId>
        <artifactId>java-client</artifactId>
        <version>7.4.1</version>
</dependency>
```

```
<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java
-->
<dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>3.141.59</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->
<dependency>
        <groupId>org.testng</groupId>
        <artifactId>testng</artifactId>
        <version>7.3.0</version>
        <scope>test</scope>
</dependency>

<!-- https://mvnrepository.com/artifact/com.googlecode.json-simple/json-
simple -->
<dependency>
        <groupId> com.googlecode.json-simple</groupId>
        <artifactId>json-simple</artifactId>
        <version>1.1.1</version>
</dependency>

</dependencies>
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-jar-plugin</artifactId>
            <version>2.6</version>
            <executions>
                <execution>
                    <goals>
                        <goal>test-jar</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-dependency-plugin</artifactId>
            <executions>
                <execution>
                    <id>copy-dependencies</id>
                    <phase>package</phase>
                    <goals>
                        <goal>copy-dependencies</goal>
                    </goals>
                    <configuration>
                        <outputDirectory>${project.build.directory}/dependency-
jars/</outputDirectory>
                    </configuration>
                </execution>
            </executions>
```

```
        </plugin>
        <plugin>
          <artifactId>maven-assembly-plugin</artifactId>
          <executions>
            <execution>
              <phase>package</phase>
              <goals>
                <goal>single</goal>
              </goals>
              <configuration>
                <finalName>Aquamark</finalName>
                <appendAssemblyId>false</appendAssemblyId>
                <descriptors>
                  <descriptor>src/main/assembly/zip.xml</descriptor>
                </descriptors>
              </configuration>
            </execution>
          </executions>
        </plugin>
      </plugins>
    </build>
```

- Add dependencies to zip.xml as given in the folder structure in Figure 47.

```
<assembly
   xmlns="http://maven.apache.org/plugins/maven-assembly-
plugin/assembly/1.1.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/plugins/maven-assembly-
plugin/assembly/1.1.0"
   "http://maven.apache.org/xsd/assembly-1.1.0.xsd">
   <id>zip</id>
   <formats>
      <format>zip</format>
   </formats>
   <includeBaseDirectory>false</includeBaseDirectory>
   <fileSets>
      <fileSet>
         <directory>${project.build.directory}</directory>
<outputDirectory>./</outputDirectory>
         <includes>
            <include>*.jar</include>
         </includes>
      </fileSet>
      <fileSet>
         <directory>${project.build.directory}</directory>
         <outputDirectory>./</outputDirectory>
         <includes>
            <include>/dependency-jars/</include>
         </includes>
      </fileSet>
   </fileSets>
</assembly>
```

2. Configure your test package

Step 01: Write the environment variables or desired capabilities to MyTest.json

Environment variables represent values that are used by your automated tests. You can use these environment variables in your MyTest.json and test code. App Functional dynamically populates environment variables at runtime by giving the values to the required environment variables.

Capabilities	Description	Values
platformName	Mobile OS platform to use	Eg: iOS, Android, or FirefoxOS
platformVersion	Mobile OS version to use	Eg: 7.1, 4.4
deviceName	Kind of mobile device to use	Eg: Galaxy S4, iPhone 11 For Android devices, run adb devices For iOS devices, run instruments -s devices
appPackage	Java package of the app you want to run.	Run adb shell dumpsys window | grep -E 'mCurrentFocus' Output is of the format appPackage/appActivity and fetch appPackage s the value. Open the app on a real device connected via USB(with USB debugging on) before running adb shell command.
appActivity	Activity name for the Android activity you want to launch from your package.	Run adb shell dumpsys window | grep -E 'mCurrentFocus' Output is of the format appPackage/appActivity and fetch appActivity as the value. Open the app on a real device connected via USB(with USB debugging on) before running adb shell command.
appiumURL	Appium server IP address	Eg: http://127.0.0.1:4723/wd/hub

Step 02: Add the filepath of MyTest.json to an environment variable

1. Open terminal(MacOS/Linux) or command line(Windows)

2. Initialize “DESIRED_CAPABILITIES_FILE_PATH” with the filepath of MyTest.json by executing the below command:

For terminal(MacOS/Linux):

```
export DESIRED_CAPABILITIES_FILE_PATH=<filepath of MyTest.json>
```

For command line(Windows):

```
set DESIRED_CAPABILITIES_FILE_PATH=<filepath of MyTest.json>
```

3. Open Eclipse app from terminal(MacOS/Linux) or command line(Windows)

4. Use the below method to BaseClass.java to read the filepath of MyTest.json file from an environment variable in the folder structure:

```
System.getenv("DESIRED_CAPABILITIES_FILE_PATH");
```

Step 03: Create a zipped test package file

```
mvn clean package -DskipTests=true
```

The target directory file path will be:
java_testng_appium_android/target/java_testng_appium_android.zip

and the sample file name of .zip file will be(in the case of 5GMark):
java_testng_appium_android.zip

The file with .zip extension will be created as a result. This is your test package.

Please note:

- Put username and password for GMAIL login, Facebook login as part of the script if there is any testcase that requires such credentials.
- Rectify all the build errors and missing dependencies before bundling your test package.

Step 04: Bundle to create your zipped test package file for App Functional

Now, create a test bundle for App Functional by zipping MyTest.json and .zip file created in previous two steps as MyTestPackage.zip to upload your tests to App Functional.

3. Upload the test package(scripts) to App Functional

- Sign in to App Functional console: http://demo-appfunctional.mozark.ai/
- Create or select a project. Follow the instructions on how to create or select a project.
- Upload or select a build. Follow the instructions on how to create or select a project.
- Upload the MyTestPackage.zip as test script. Follow the instructions on how to upload a test script.

Please refer to our sample build and sample scripts for Appium with Java TestNG:

Android(Build): https://bitbucket.org/mozarkai/mozark-java-appium-testng-tests-android-5gmark/src/master/

iOS(Build): https://bitbucket.org/mozarkai/mozark-java-appium-testng-tests-ios-5gmark/src/master/

5. Limits

The following list describes current App Functional limits:

- The maximum file size of an app that you can upload is 4MB.
- The maximum file size of a test bundle that you can upload is 4MB.
- There is no limit to the number of projects that you can upload to test.
- There is no limit to the number of apps that you can upload to test.

- There is no limit to the number of test bundles that you can upload to test.
- There is no limit to the number of devices that you can include in a test run.
- There is no limit to the number of runs that you can schedule.
- There is no limit to the duration of an automated test run.
- You can retain the test results for one month.
- You can retain test logs for one month.
- You can retain screenshots and videos for one month.

6. Document History

S.No	Date	Version Number
1.	Dec 8, 2020	1.0

7. Resources

This section contains links to product video tutorials that can help you in getting started with App Functional.

Product Videos (https://www.youtube.com/channel/UCdcRIhyX-ZzYWRCvrSiIlHg):

1. How to run an automated test?
https://www.youtube.com/watch?v=fEkpNzDK5ic

2. How to write scripts and add log results?
https://www.youtube.com/watch?v=OcgqCPc1lvw

8. Support

For further assistance, you can reach out to enquiry@mozark.ai.