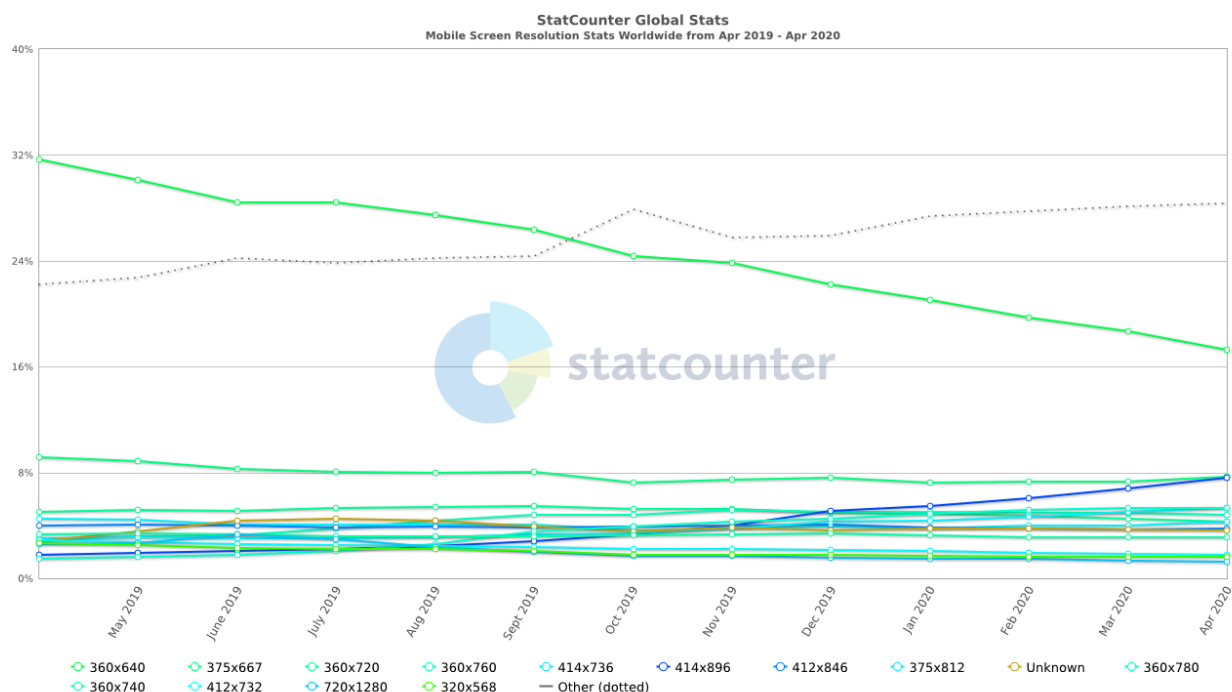


# How to perform web testing on Android devices

Gone are the days when only desktops were used for accessing websites. The processing power of smartphones is increasing with each passing day and consumers are increasingly using devices such as smartphones, tablets, etc. to access web content.

It is essential that your website runs properly on different device resolutions along with being fluid & responsive on different types of mobile devices. Mobile-friendly websites also have an edge over non-mobile friendly websites as popular search-engine like Google gives more preference to mobile-friendly websites.



There are numerous challenges that you might face when testing on Android devices:

1. Every month (if not every week), a new Android device is released in the market. Procurement of a couple of devices is practically feasible but if the 'cycle of procurement' ends up in an infinite loop, the intent of testing on actual devices completely falters out.

2. Testing on cloud-device farms is one option but it might not be an affordable option for many i.e. early-stage companies, freelancers, startups, and more.

3. Device Fragmentation is another major issue for the Android operating system. There are a number of OEMs (Original Equipment Manufacturer) manufacturing Android phones and the question to ask is whether testing on couple of a selected few Android devices is enough or not?

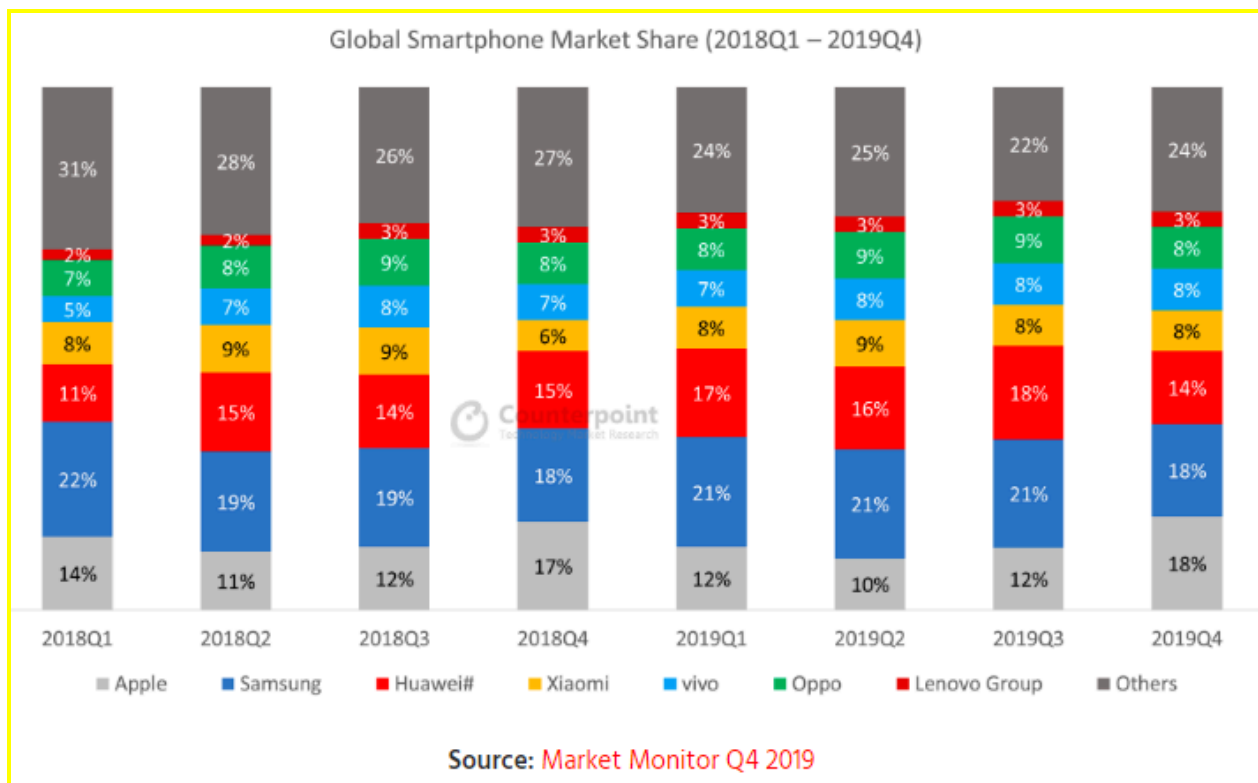


Figure 1 [Source](#)

4. There are varieties of testing tools available in the market and choosing the right tool that can suffice your requirements and budget can be a daunting task.

Apart from these high-level challenges, there are a number of micro-level challenges that makes web testing on Android devices an up-hill task.

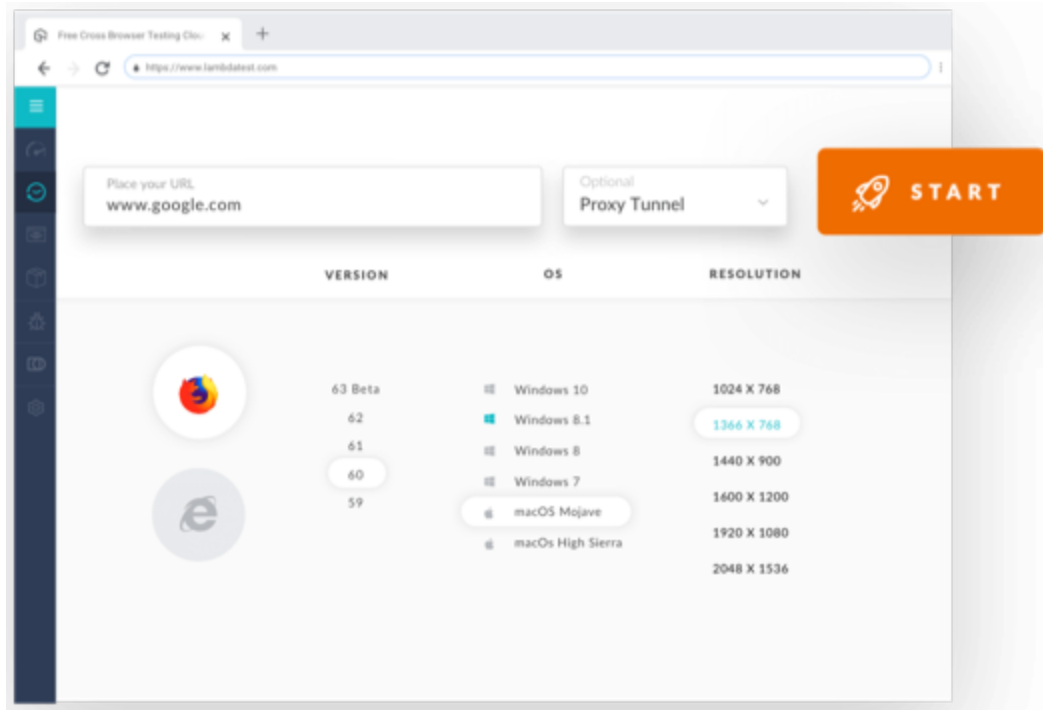
A feasible option is to choose a testing approach (or solution) that aids you in performing automation testing in a smart manner. It should be useful in accelerating testing of your web app on Android devices. Along with test coverage, you also need to focus on browser coverage so that the overall experience is same across different browsers i.e. cross browser testing.

## **How cross browser testing is performed?**

There are two types of cross browser testing - manual browser testing and automated browser testing.

### **Manual cross browser testing**

As the name indicates, manual cross browser testing is performed by testing the web app across different combinations of browsers and Android emulators (and simulators). In the manual approach, testers have to identify the test combinations and manually run the cross browser tests for observing behavior of the web app along with locating bugs in it.

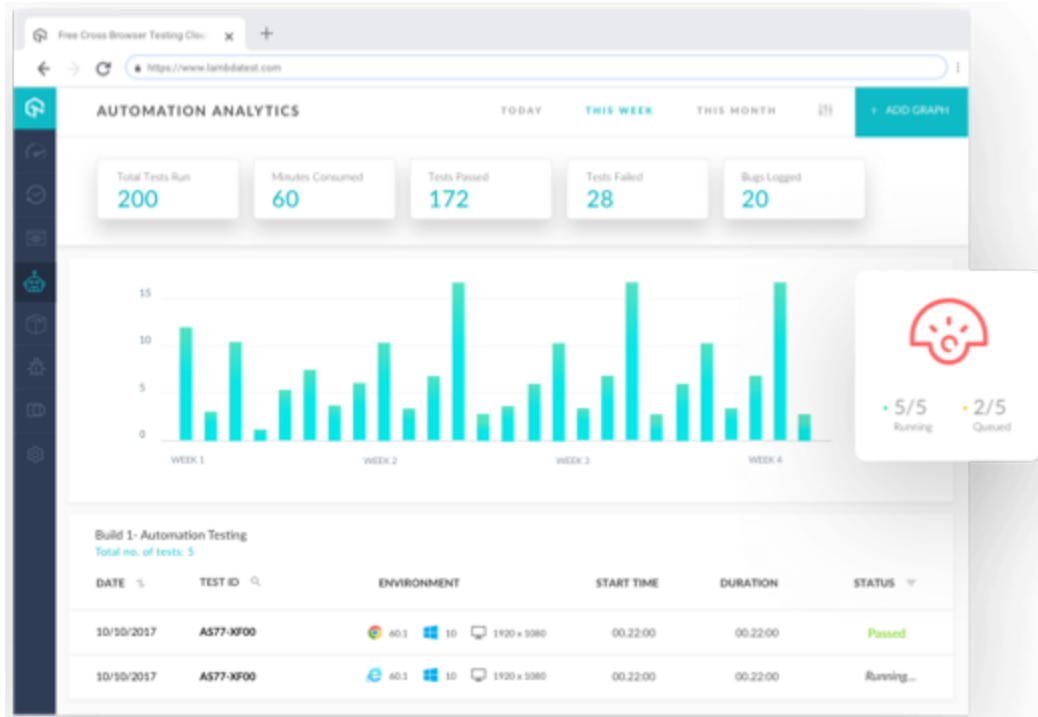


Below are some of the major challenges of manual cross browser testing:

- Time taken for execution of cross browser tests will exponentially increase with the complexity of the web app and the number of test combinations.
- Test coverage might be limited as manual cross browser testing will require significant amount of time, manual resources, as well as, computing resources.

## Automated cross browser testing

Automated browser testing is similar to manual browser testing with one big difference – cross browser tests are executed using an automation tool, thereby making the entire process manageable.



In automated cross browser testing, testers write automation scripts using appropriate test frameworks to perform testing on different browser & device combinations.

Below are some of the major advantages of automated approach to cross browser testing:

- The test process is less-time consuming as automation scripts (with minimal modifications) can be used for testing.
- It improves the test efficiency and effectiveness as test coverage can be improved by executing the scripts across different test combinations.

Though there are a large number of test frameworks available in the market, Selenium is the most widely-used for automated cross browser testing as it supports different programming languages, has excellent developer community, and reduces the script-maintenance cost. Selenium framework comprises of four major tools - Selenium IDE, Selenium RC, Selenium WebDriver, and Selenium Grid. Selenium Grid is very

popular as it enables developers & testers to execute automated cross browser tests across different browsers, browser versions, and devices in parallel.

Setting up in-house Selenium Grid infrastructure is not very challenging but it has severe shortcomings as mentioned below:

- Significant investment requirement for setting up the test infrastructure, which may exponentially rise with growing number of browser & device combinations.
- Additional resource overhead for maintaining the infrastructure.
- Scaling up is difficult, as well as, expensive as having a local setup for every (browser + Android OS version) combination is highly unfeasible.
- Achieving higher test coverage is not a simple task with a local (or in-house) Selenium Grid.

A more practical, in-expensive, and scalable approach is using leveraging cross browser testing on the cloud for performing web testing on Android devices.

[LambdaTest](#) is a popular online Selenium Grid using which web testing can be performed on best Android browser emulators online.

## **LambdaTest - Test on best Android Browsers Emulator Online**

LambdaTest is a cloud-based cross browser testing platform that helps you perform cross browser tests for web applications across 2000+ browsers, operating systems, and devices. It is easy to get started with LambdaTest, all you have to do is create an [account on the LambdaTest platform](#) and choose the appropriate [pricing plan](#) that suits your requirements.

LambdaTest has an awesome set of features when compared to other testing tools and these features can be leveraged to expedite the cross browser testing activity. The

platform provides a host of options (or features) that lets you automate and optimize the process.

## **Salient features of LambdaTest**

Here are some of the best features of LambdaTest that aids in web testing for the Android platform:

### **1. Real-time testing**

As the name indicates, real time testing in LambdaTest is useful when you want to check the status of a page (or complete website) on a certain device configuration. For example - If the client has raised the issue on a particular device (or browser version), the same test scenario can be replicated on LambdaTest by simply entering the required device and browser information.

Real-time testing is useful for interacting with the application live on different Android emulators.

As the details have to be selected manually, real-time testing is particularly useful on narrowing down the problem to selected devices (or browser versions). To perform real-time testing, enter the test URL and select the viewing specifications (i.e. device & browser configuration).

Below are some of the options that you can choose from:

- Platform - Android, iOS
- Web browser - Chrome, Firefox, Opera, Chromium, and others
- Device Manufacturer - Samsung, LG, One Plus, and others

- Device Type & Platform version - Galaxy S10, One Plus 6T, Google Pixel, and others

Here is the screenshot of configuration selected for testing our website in Google Chrome on One Plus 7 (Android 9.0)

Select your configuration to start Real time testing!

Place your URL  
<http://fortech.org/>

Optional  
Select Tunnel

START

	BRAND	DEVICE/OS	BROWSER
	Samsung	One plus 7 pro 9.0	Google Chrome
	Google	One plus 7 9.0	Firefox
	LG	One Plus 6T 9.0	Opera
	One Plus	One Plus 3T 5.1	
	HTC		
	Amazon		
	Motorola		
	Sony		
	Xiaomi		
	Huawei		
	Gionee		
	Oppo		
	Vivo		

app.lambdatest.com/viewer/TES100963501589108074861351



#### Hold Tight, Setting up your Browser

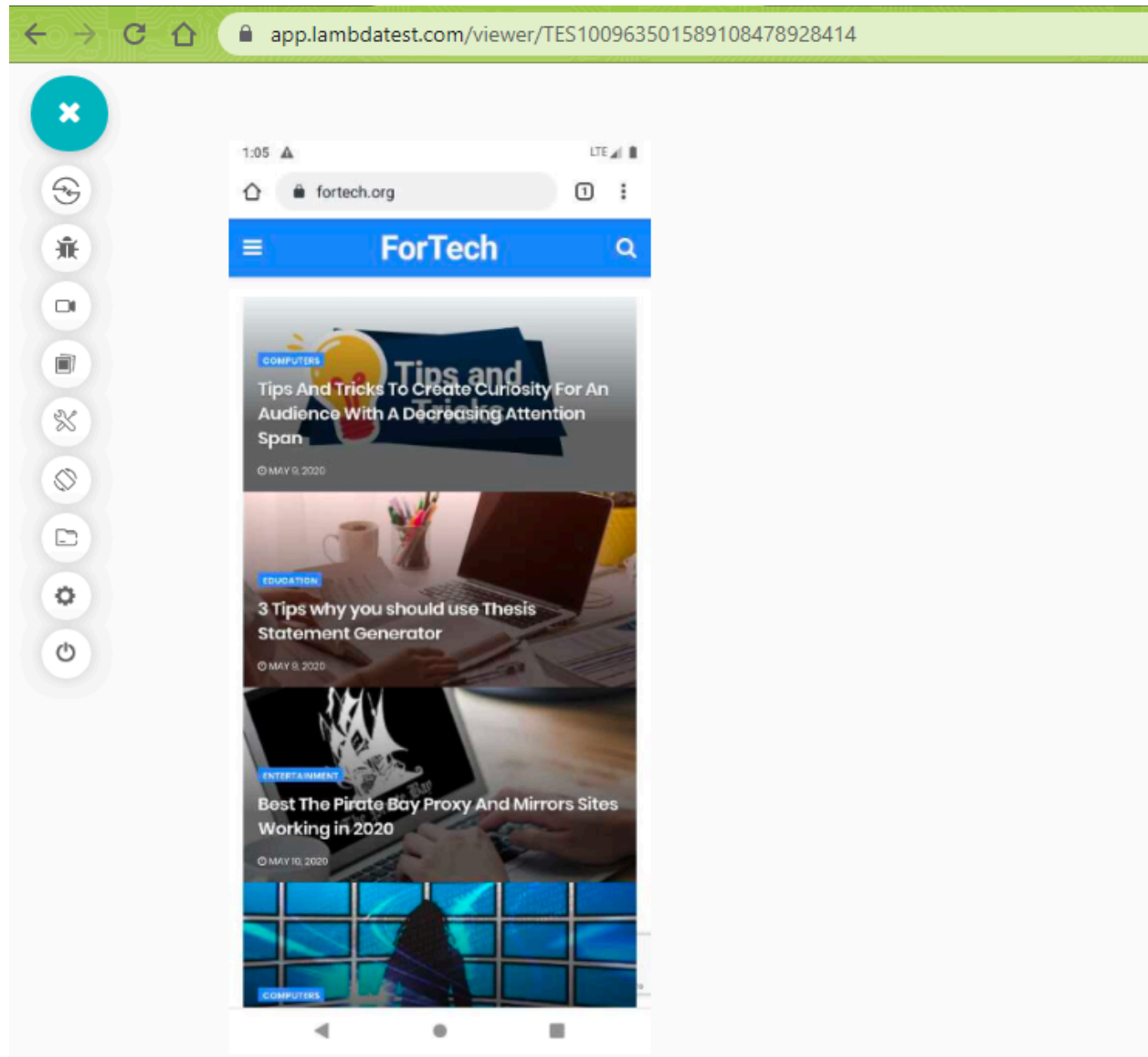
- ✓ Allocating a cloud machine
- ✓ Configuring the operating system
- Preparing a clean browser profile
- We are ready to take off!

Cancel

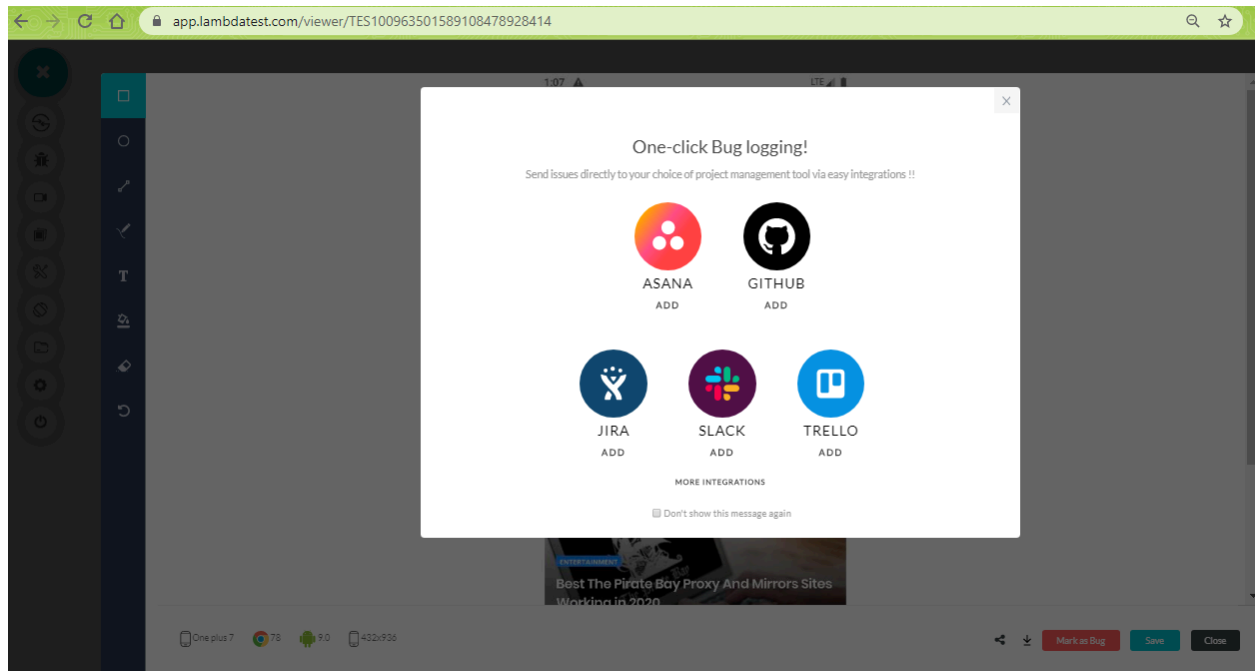


Once the testing begins, you also have the flexibility to choose a different test configuration from the real-time test viewer itself. This avoids the frequent back & forth to the test console for selecting different configurations. Apart from this, the sidebar on the viewer for real-time testing provides features:

- To mark a particular area in the app as bug & assign the same to a developer
- Enable device rotation and test functionalities of the web app
- Record video of the activity being performed inside the web app
- Use 'developer tools' for real-time debugging



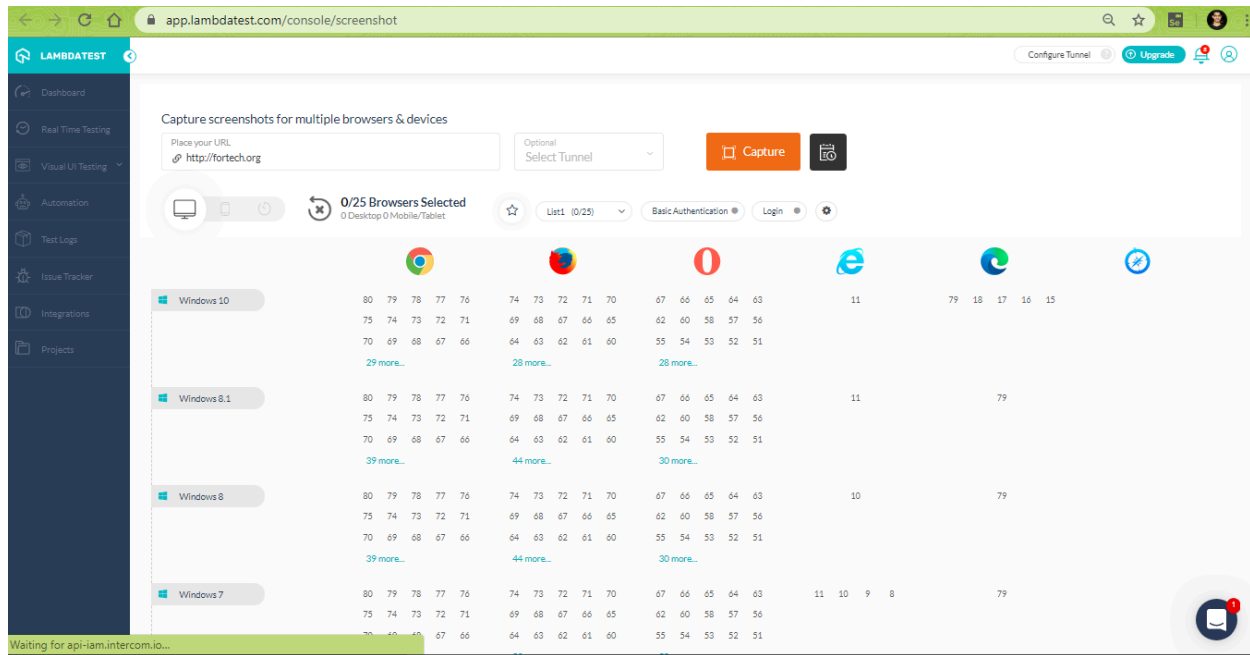
LambdaTest provides one-time bug logging through which issues can be sent directly to the choice of project management tools (e.g. Asana, GitHub, Trello, Jira, etc.) via easy integrations.



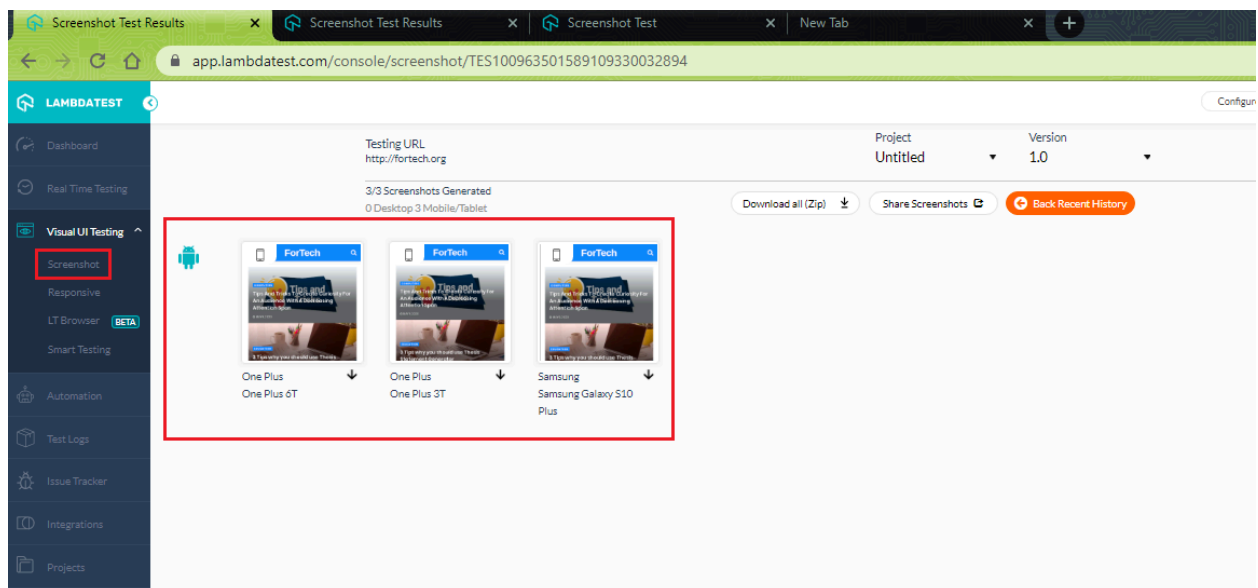
## 2. Screenshot Testing

Take a scenario where the user interface of a particular web page is breaking on a select few devices. You already have the device information and the web information which means that all you are required to do is test how the page looks like on those selected few devices. This is what Screenshot testing, a part of the Visual UI testing section in LambdaTest is meant to do.

In LambdaTest, navigate to the [Screenshot Testing](#) in Visual UI Testing menu. Enter the test URL and select the appropriate *device configuration*.



Once the required information is keyed in, click on the Capture button to execute the screenshot test on the target URL. Shown below is the snapshot of the screenshot test executed for our website on One Plus 3T, One Plus 6T, and Samsung Galaxy S10 Plus.



Once the screenshots are ready, the platform sends across an email indicating the readiness of the images for download. The screenshots can be organized as per the

project and its version. The platform eases the task of maintaining the images by using a pre-defined nomenclature for naming of the screenshot images.

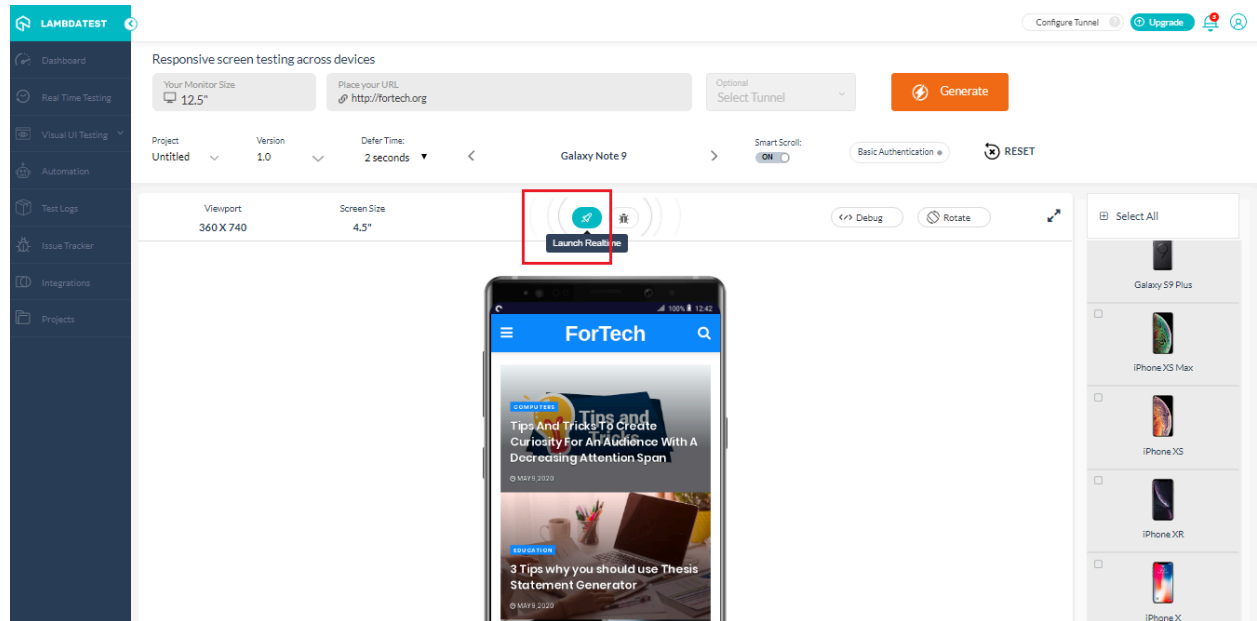
Screenshots captured via Screenshot testing can be useful when performing regression testing on the web app (or website).

### **3. Responsive Testing**

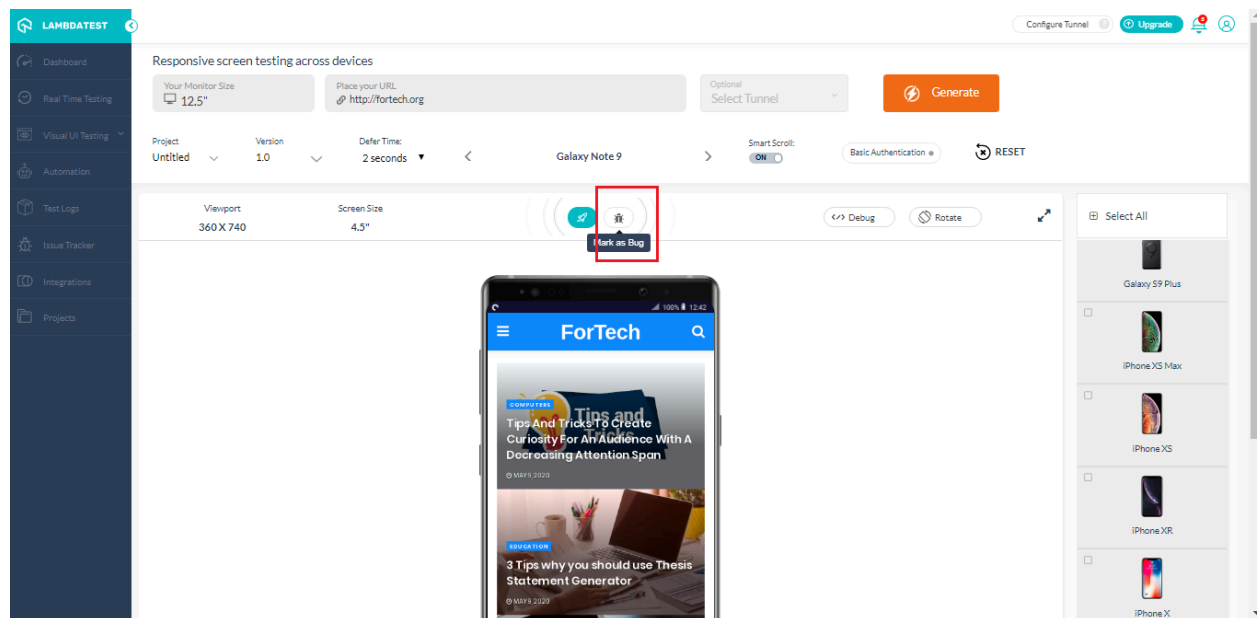
Responsive testing is primarily used to check whether the web app is designed by keeping the mobile-first principles in place. This feature in LambdaTest lets you interact with the web app in real-time. All you need to do is go to [responsive testing](#) option (inside Visual testing) and select the test-device(s) along with the target URL on which the testing has to be performed.

It is particularly useful for reviewing the web app's design and interactivity in different browsers, along with reviewing the site's responsiveness when it comes to changing device orientations.

There is also an option to launch real-time test from the same window.



Like real-time testing, you can also raise bugs right from the responsive testing window and assign it to the appropriate developer(s).



## 4. Automated Browser Testing

The features that we have seen so far can only be used if the cross browser testing has to be done on a smaller scale. We are mentioning 'smaller scale' as details need to be manually entered for starting those tests. Hence, there are inherent limitations with those testing mechanisms.

To perform cross browser testing at scale, LambdaTest provides an option of [automation testing](#) with an online Grid for [Selenium test automation](#). Porting an existing automated test implementation that uses a local Selenium Grid to LambdaTest's remote Selenium Grid is very easy as only infrastructural-related changes are required.

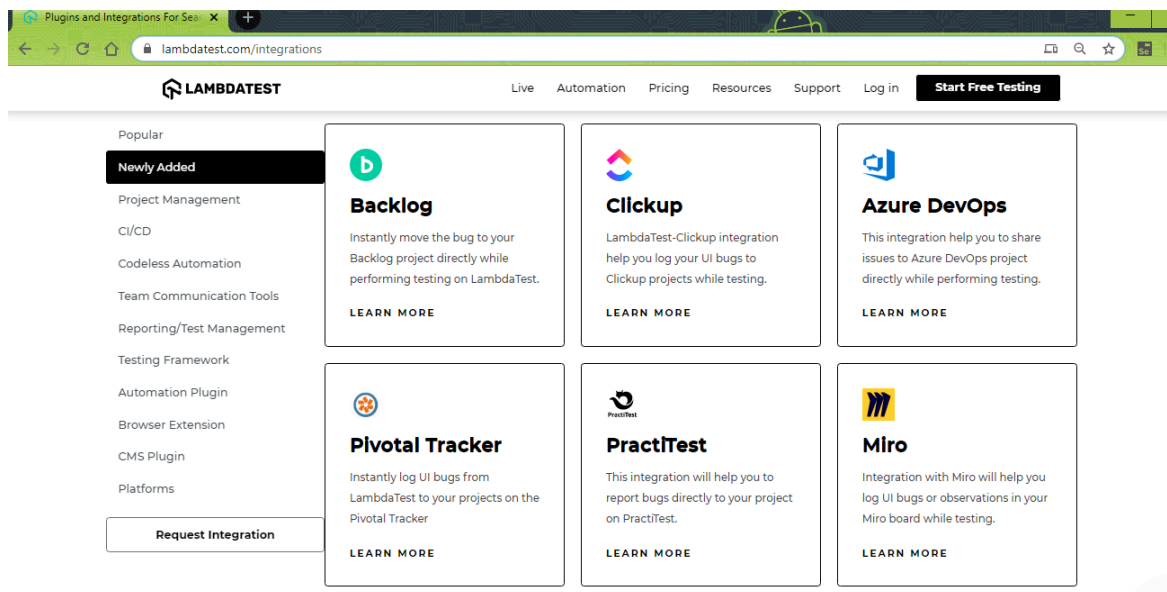
The [LambdaTest Capabilities Generator](#) is used to generate (browser + OS combinations) for Selenium supported programming languages such as Java, C#, PHP, Python, Ruby, JavaScript, and more. Shown below is the example capabilities generated for Galaxy S10 (Android 9.0). Appium; the open-source, cross-platform automation testing tool is used for performing web testing on Android devices.

The screenshot displays the LambdaTest Capabilities Generator web application. The browser address bar shows 'lambdatest.com/capabilities-generator/'. The page features a navigation bar with links for Automation, Live, Blog, Pricing, Support, and a 'Go to Dashboard' button. Below the navigation bar, there are tabs for Selenium 4, Selenium, and Appium (which is currently selected). To the right of these tabs are language selection buttons for Java, C#, PHP, Ruby, Javascript, and Python, along with a 'Copy to clipboard' button. The main configuration area is divided into several sections: 'USERNAME AND ACCESS KEYS', 'OS/BROWSER CONFIGURATION', 'GEO LOCATIONS', and 'ADVANCED CONFIGURATION'. In the 'OS/BROWSER CONFIGURATION' section, 'Select OS' is set to 'Android', 'Select Device' is set to 'Galaxy S10', 'Version' is set to '9', and 'Appium Version' is set to '1.12.1'. On the right side of the configuration area, a code editor displays the generated Appium capabilities in Java: 

```
DesiredCapabilities capabilities = new DesiredCapabilities();
capabilities.setCapability("build", "your build name");
capabilities.setCapability("name", "your test name");
capabilities.setCapability("platformName", "Android");
capabilities.setCapability("deviceName", "Galaxy S10");
capabilities.setCapability("platformVersion", "9");
```

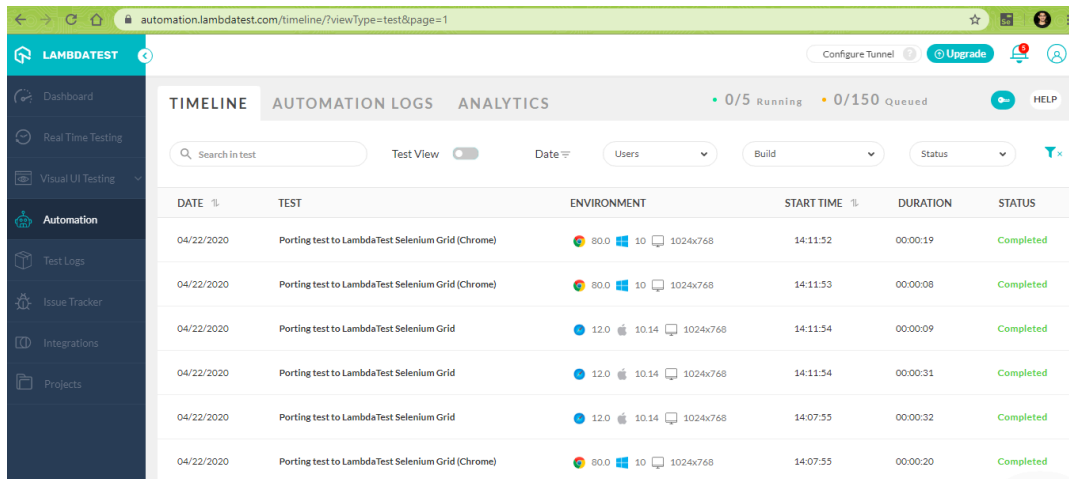
## Advantages of Automated Browser Testing on LambdaTest

1. The platform supports performing parallel testing using Selenium and Appium which reduces the overall time spent in executing the test scenarios. The blog titled [How Parallel Testing Instantly Improves Your Workflow](#) throws more light on the advantages of parallel testing.
2. When compared to manual testing, automated browser testing using LambdaTest helps in increasing the browser coverage, making the implementation fool-proof and robust.
3. Porting effort is minimal as implementation that uses local Selenium Grid can be ported to LambdaTest's cloud-based Selenium Grid with minimal changes.
4. You can use CI/CD, bug tracking tools, project management tools, etc. as the platform provides [seamless integration with popular third-party apps](#). This feature contributes a lot in acceleration of automated browser testing.



5. It helps in tracking the progress of the testing activity as test results (and reports) are available in the [Automation Timeline](#).





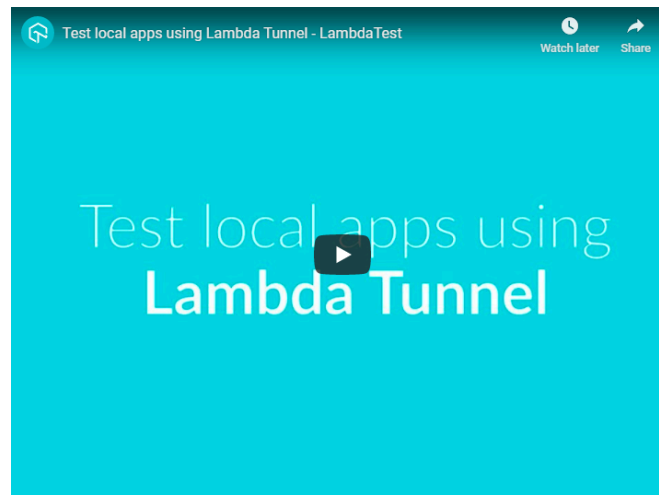
The screenshot shows the LambdaTest Automation Logs page. The left sidebar contains navigation links: Dashboard, Real Time Testing, Visual UI Testing, Automation (selected), Test Logs, Issue Tracker, Integrations, and Projects. The main content area has tabs for TIMELINE, AUTOMATION LOGS, and ANALYTICS. The TIMELINE tab is active, displaying a table of test runs. The table has columns for DATE, TEST, ENVIRONMENT, START TIME, DURATION, and STATUS. All tests shown are 'Porting test to LambdaTest Selenium Grid (Chrome)' and have a status of 'Completed'.

DATE	TEST	ENVIRONMENT	START TIME	DURATION	STATUS
04/22/2020	Porting test to LambdaTest Selenium Grid (Chrome)	80.0 10 1024x768	14:11:52	00:00:19	Completed
04/22/2020	Porting test to LambdaTest Selenium Grid (Chrome)	80.0 10 1024x768	14:11:53	00:00:08	Completed
04/22/2020	Porting test to LambdaTest Selenium Grid	12.0 10 1024x768	14:11:54	00:00:09	Completed
04/22/2020	Porting test to LambdaTest Selenium Grid	12.0 10 1024x768	14:11:54	00:00:31	Completed
04/22/2020	Porting test to LambdaTest Selenium Grid	12.0 10 1024x768	14:07:55	00:00:32	Completed
04/22/2020	Porting test to LambdaTest Selenium Grid (Chrome)	80.0 10 1024x768	14:07:55	00:00:20	Completed

## Testing Locally Hosted Pages using Lambda Tunnel

LambdaTest also offers an app called Lambda Tunnel that is instrumental in testing locally hosted pages and privately hosted pages in their [Selenium test automation](#) platform. It allows you to connect your local system with LambdaTest servers via SSH based integration tunnel.

Lambda Tunnel is especially useful for developers & testers who love to get things done via the terminal. Here is the small video that can help you get started with Lambda Tunnel:



Here is some information that will help in setting up Lambda Tunnel for your choice of host platform:

- [How to get started with Lambda Tunnel for Windows](#)
- [How to connect Lambda Tunnel For Mac](#)
- [How to connect Lambda Tunnel for Linux](#)
- [How to share Lambda Tunnel](#)
- [Lambda Tunnel to bypass Proxy Settings and Corporate Firewalls](#)

The key advantage of using Automated web testing on LambdaTest is it expedites the entire testing process as tests can be executed on different web browsers & Android emulators in parallel.

## 5. Smart Testing

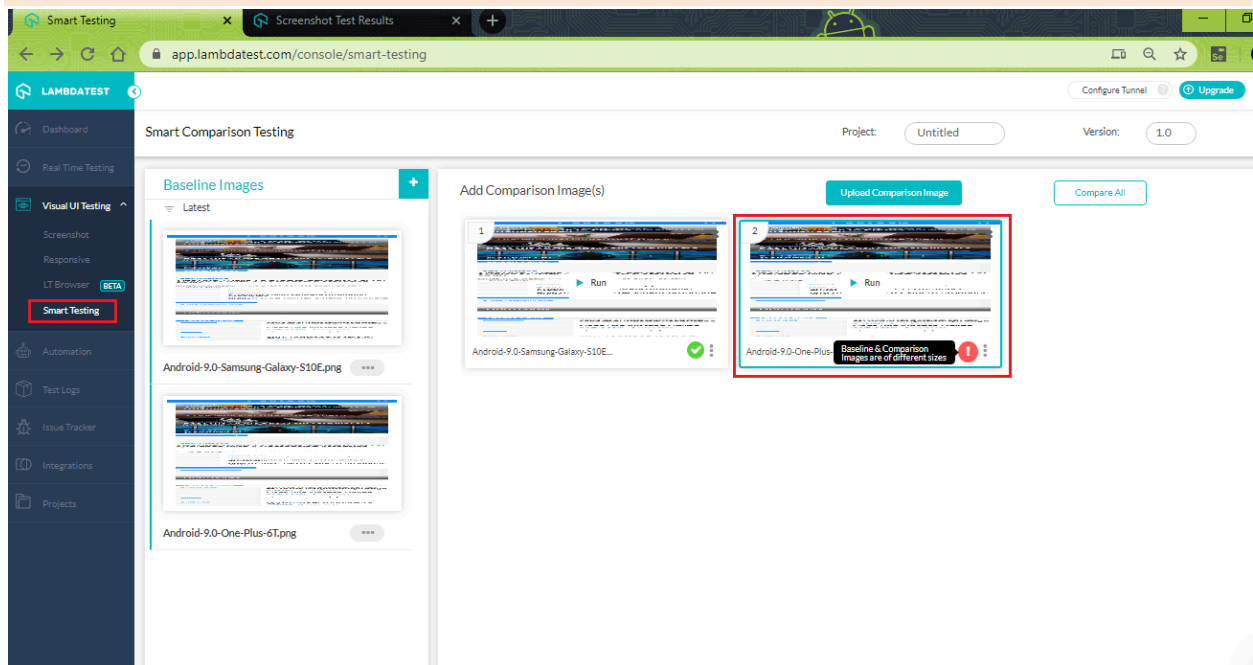
As the app development advances, developers need to ensure that new implementation does not cause any side-effects or do not hamper an already working functionality. This is where Smart Testing can be useful as it is one of the go-features on the platform for regression testing.

Once testers raise an issue for particular screen(s), smart testing is used to perform visual comparison of the two pages (impacted page where issue is seen and earlier version of the page where the issue did not exist). This is where [Screenshot testing](#) comes handy as the screen grabs (or page grabs) for smart testing were captured using screenshot testing.

For using [smart testing on LambdaTest](#), just upload the baseline image (i.e. screenshot of the impacted page) and the corresponding comparison image. Once the upload is

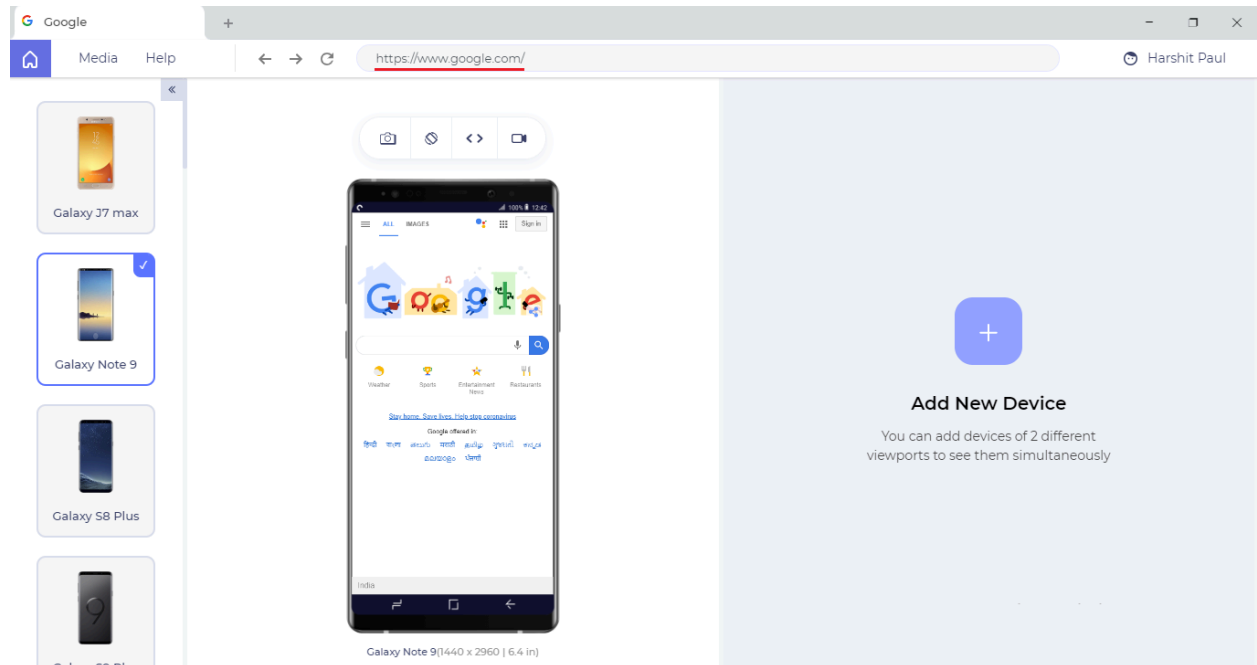
complete, press the Compare All button to trigger the visual comparison engine in smart testing.

The fascinating part about smart testing is that it first performs size comparison (of images) before comparing the image contents. This ensures that images being compared belong to the same device viewports which is the essential step for visual comparison.



## Bonus Tip - LambdaTest (LT) Browser

LambdaTest has developed a browser named LT browser which is currently in the Beta stage.



LT browser allows you to test the responsiveness of your website (or web app) over a wide variety of devices and view ports.

## How to download LT Browser

To start using LT browser, you need to download the executable file for your host platform:

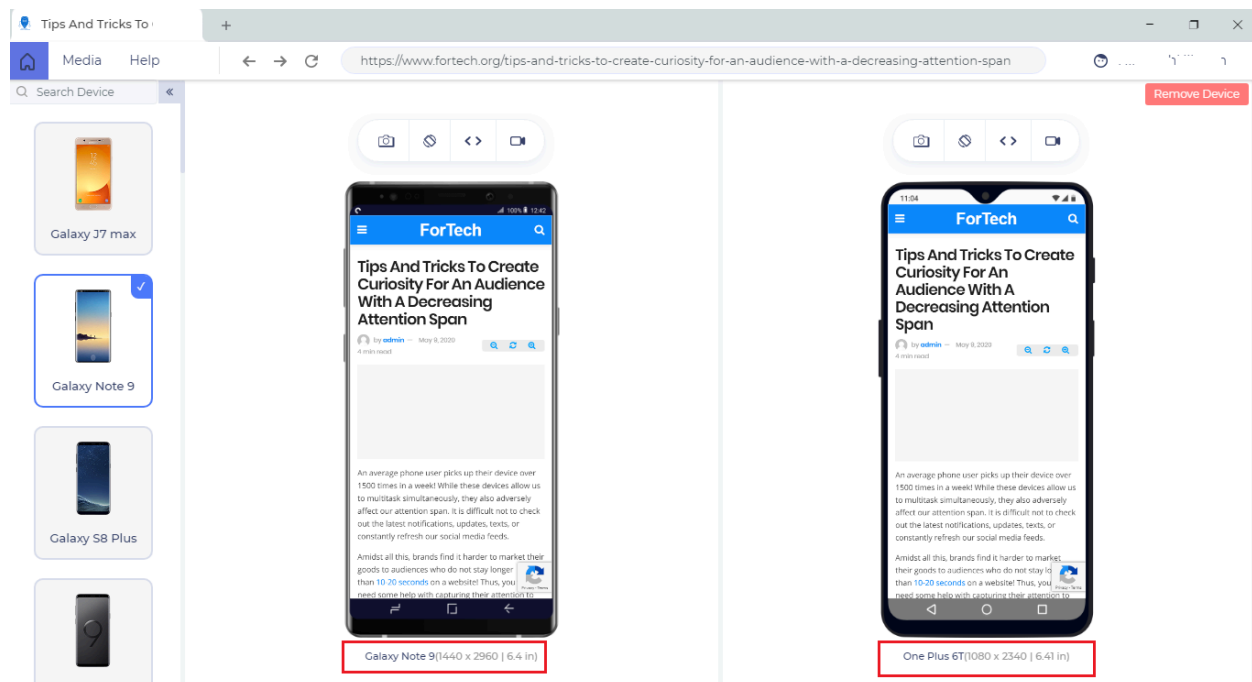
- [LT Browser for Windows](#)
- [LT Browser for macOS](#)
- [LT Browser for Linux](#)

## Advantages of LT browser for web testing

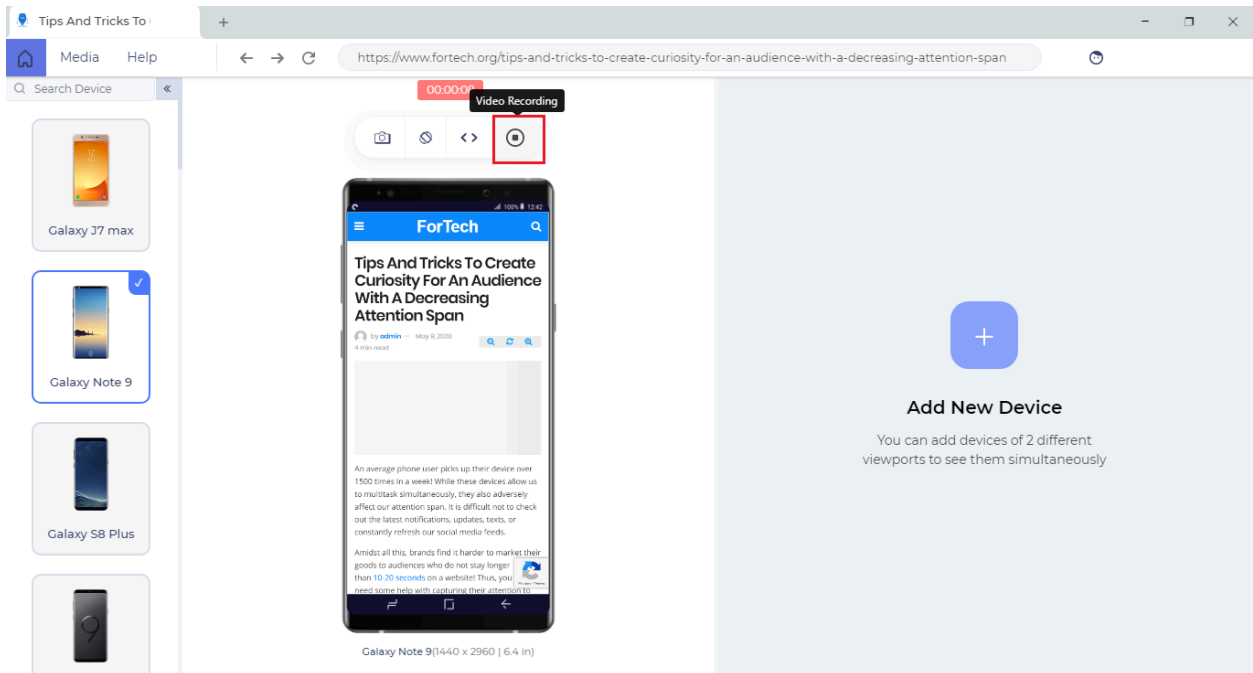
Here are some of the key advantages of using LT browser vis-à-vis the LambdaTest platform for web testing and cross browser testing:

1. **Debug on the Go in a responsive mode** - Using LT browser, you can verify how your web app will look like in different devices and viewports. Using Inspect Element tool in the browser, you can debug & fix issues on the Go.

2. **Compare layout locally and faster** - LT browser is useful for comparing web layouts across different devices and viewports. This feature is useful in identifying design issues that might occur on a select set of devices. For example, your web app might be rendering without any issues in Samsung Galaxy Note 9 but the same page may have issues in One Plus 6T. This helps in faster comparison and quicker resolution.

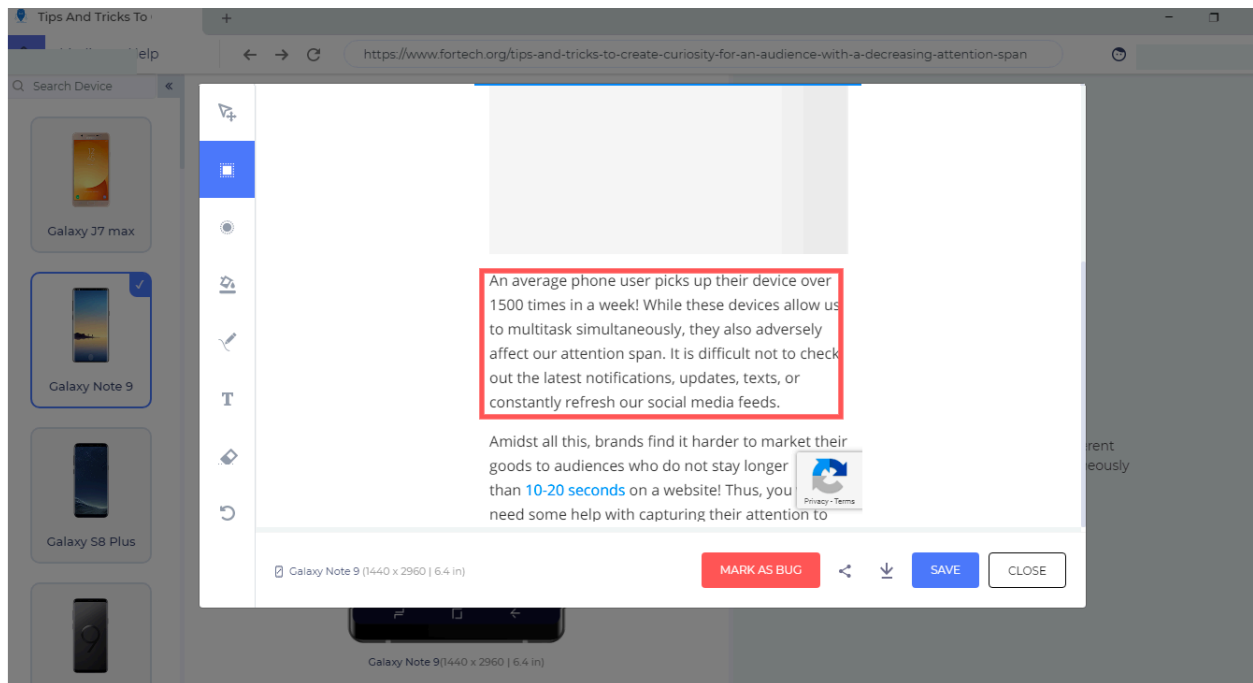


3. **Take Screenshots and Videos of bugs in responsive mode** - Akin to the features offered on the LambdaTest platform, LT browser also offers feature for taking screenshots and recording videos in a responsive mode. These images or videos can be downloaded to the local machine for further comparison.

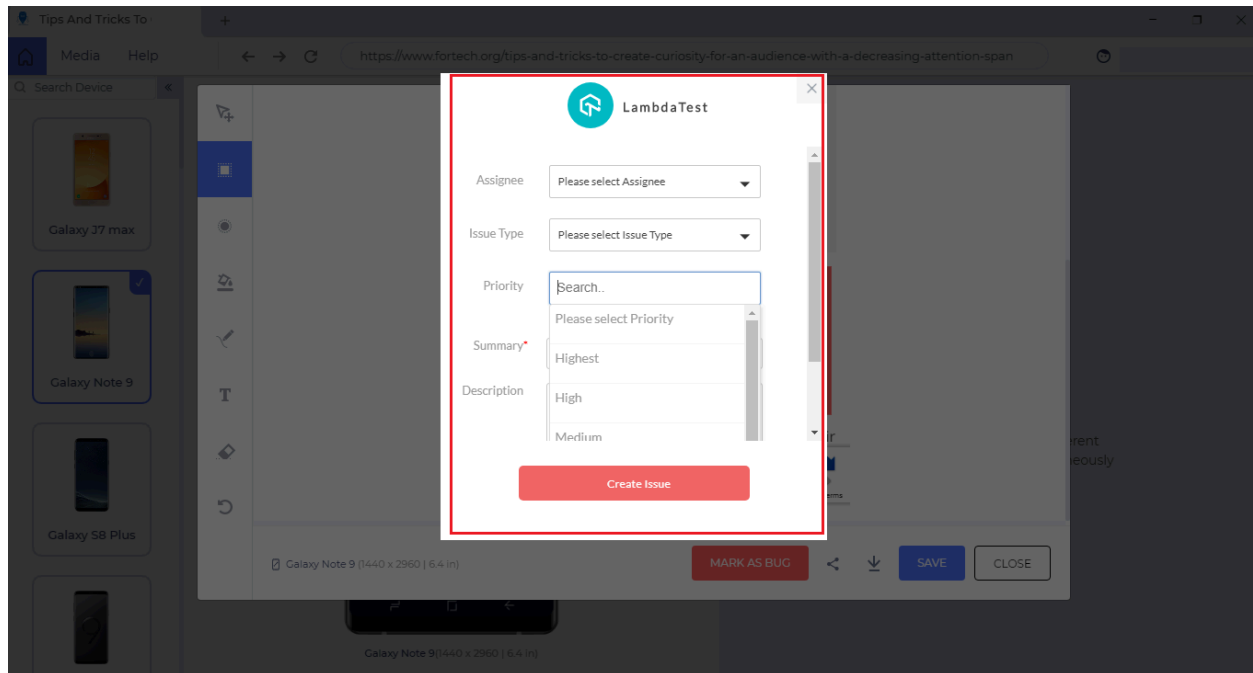


It would have been better if the link to the link would have been provided once the recording is complete. Since LT browser is still in the Beta stage, we expect number of feature advancements in the final version of the product.

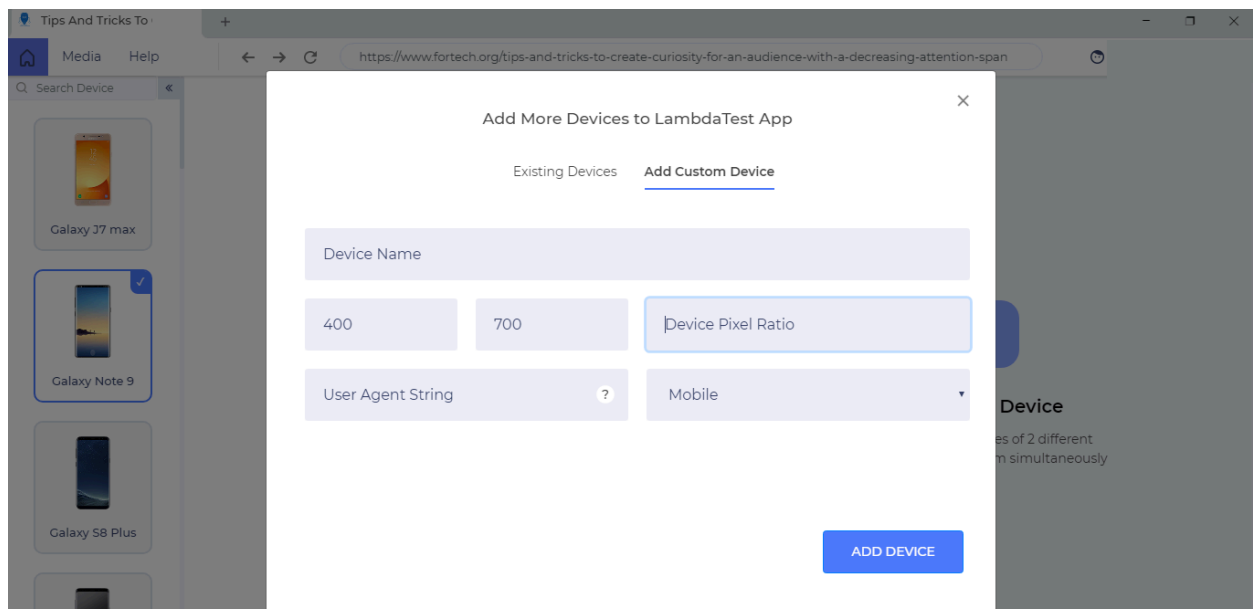
**4. Collaborate and bugs with your team** - Another major advantage of using the LT browser is that the tester can raise bugs for the web app from the LT browser context itself.



LT browser has a one-click bug logging that is useful for sending directly to your choice of project management and bug-tracking tools such as Asana, GitHub, Jira, Slack, Trello (and more) via easy integrations. Issue priority, description, summary and assignee can be entered from the LT browser; enhancing the speed of web testing and issue tracking.



5. **Perform web testing on custom devices** - In case, you are unable to find a test device with preferred view port or device pixel ratio, you add a custom device to your LT browser. For adding a custom device, click on the **add a custom device** tab.

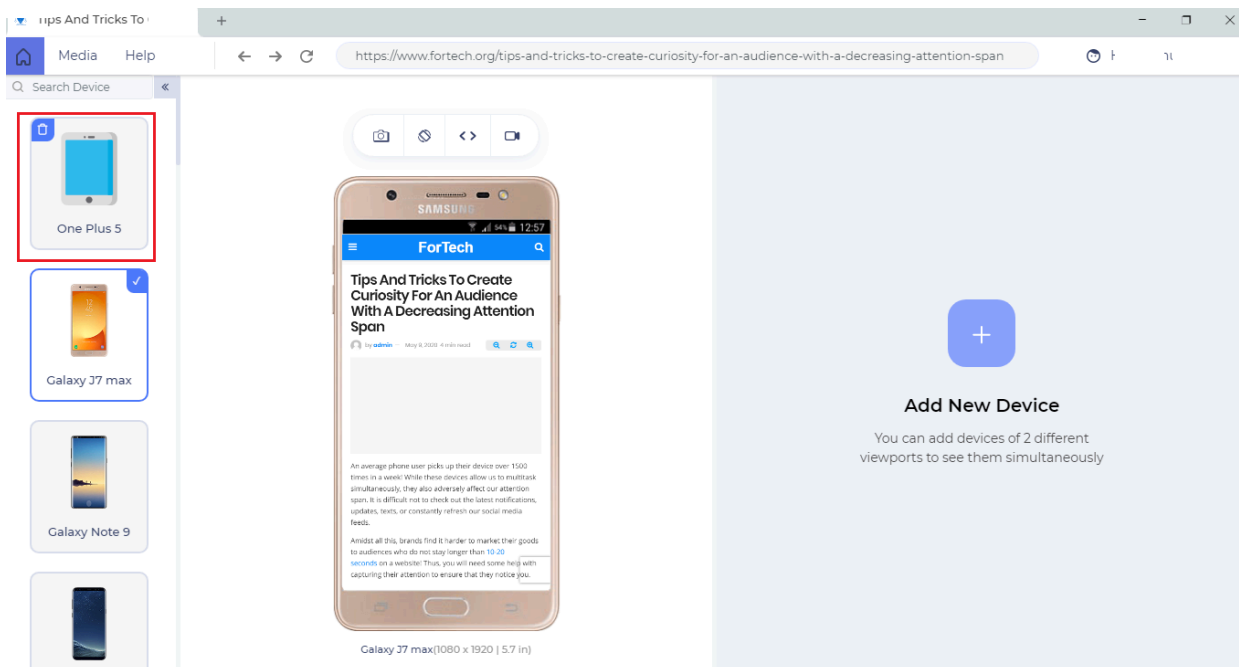


The following details have to be entered for adding the custom device:



- Device Name
- Device Pixel Ratio
- Device Platform
- View Port Specifications
- User Agent String

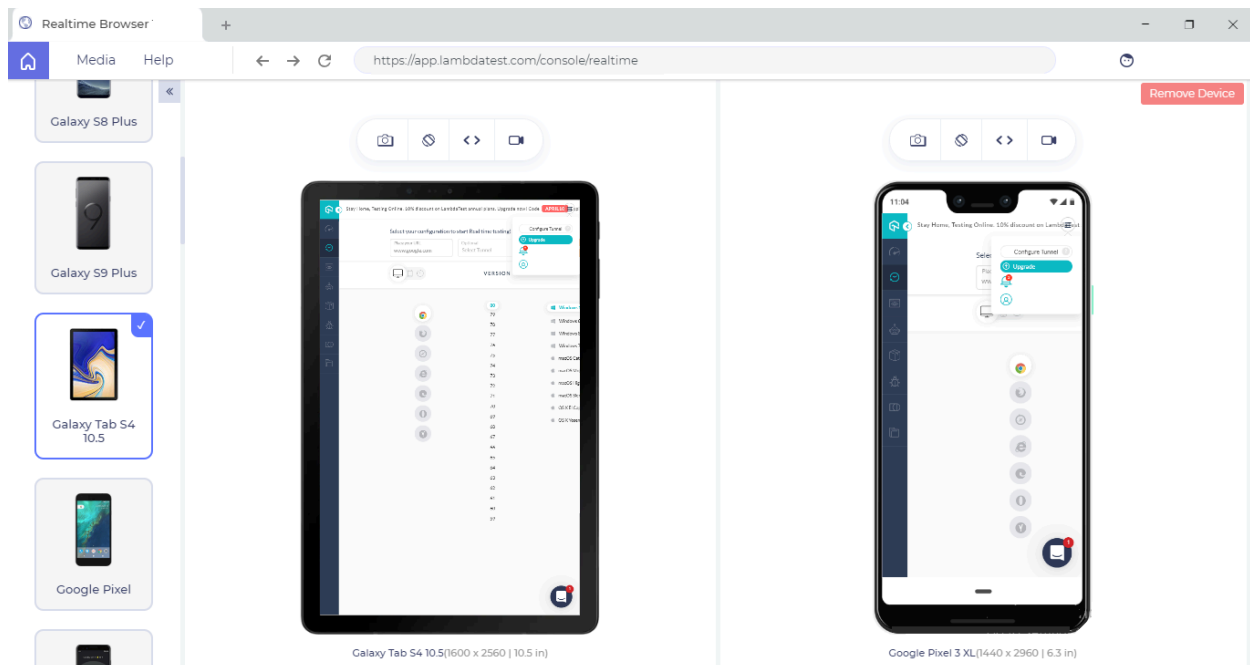
Here we have added the device specifications of One Plus 5. Once the device is added, you would find the same in the side-bar where other devices are listed.



**6. Test web apps behind login** - There are scenarios where features of the web app can only be tested after a successful login. This is typically the case with banking applications, e-commerce applications where few features such as money transfer, cart checkout, etc. requires a valid login.

LT browser provides responsive testing along with live-interactive testing experience. To test sections of web app behind login, just login with valid credentials and start testing the web pages.

This ensures that continuity of testing does not break even if the web app requires a login.



More details about LT browser are available [here](#).

## Wrapping Up

There are a number of web browsers & device configurations against which web apps (and websites) need to be tested. Web testing using a local setup is good to some extent but it is not scalable and does not help in improving the browser coverage. Instead cloud-based cross browser testing platform like LambdaTest should be used as it accelerates web testing, improves test coverage, and aids in locating errors in the web app.