BrowserStack Testing

Introduction & Overview

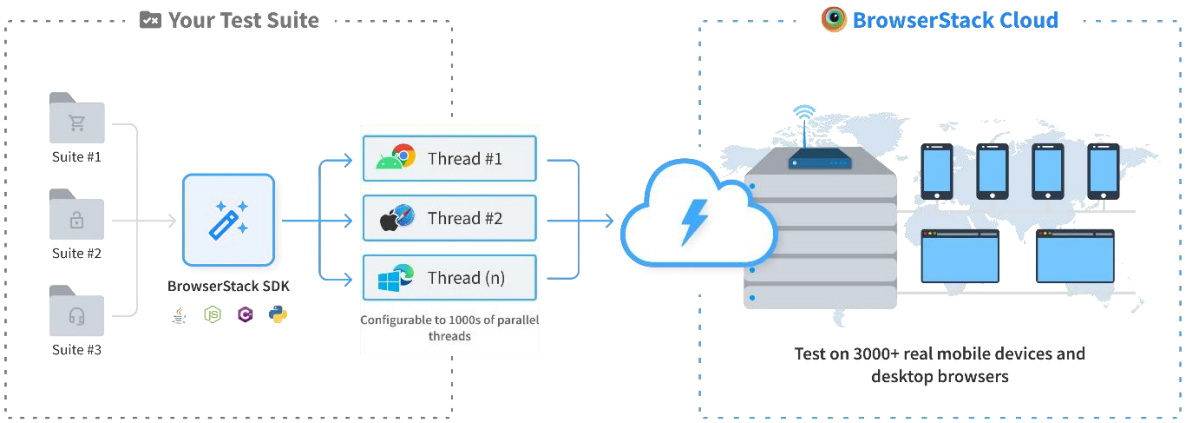
What is BrowserStack?

BrowserStack is a cloud-based testing platform that allows developers and Quality Assurance (QA) engineers to test their websites and mobile applications on real devices and browsers. Instead of buying dozens of different phones or installing multiple operating systems on your own computer, you can simply access BrowserStack's cloud to test how your software performs in the real world.
It eliminates the need for maintaining a physical device lab, providing instant access to thousands of real mobile devices (like iPhones, Samsungs) and desktop browsers (like Chrome, Safari, Edge). This ensures that applications work perfectly for every user, regardless of the device they are using.

Key Benefits of BrowserStack

Based on my testing experience, here are the main advantages of using this platform:

- **Cross-Browser Testing:** You can test web applications on a wide range of browsers (Chrome, Firefox, Safari, Edge) and their different versions without needing to install them locally.
- **Real Device Testing:** Unlike emulators, BrowserStack provides access to real physical devices (Android and iOS), which gives accurate results on how an app behaves on an actual phone screen.
- **No Local Setup Required:** Since it is completely cloud-based, there is no need to set up complex environments or install heavy software on your personal laptop. You just need a browser and an internet connection.
- **Automation Support:** It integrates seamlessly with popular automation frameworks like Selenium and Appium, allowing us to run automated test scripts to save time and effort.
- **Speed and Efficiency:** It allows for "Parallel Testing," meaning you can run multiple tests at the same time to speed up the process.

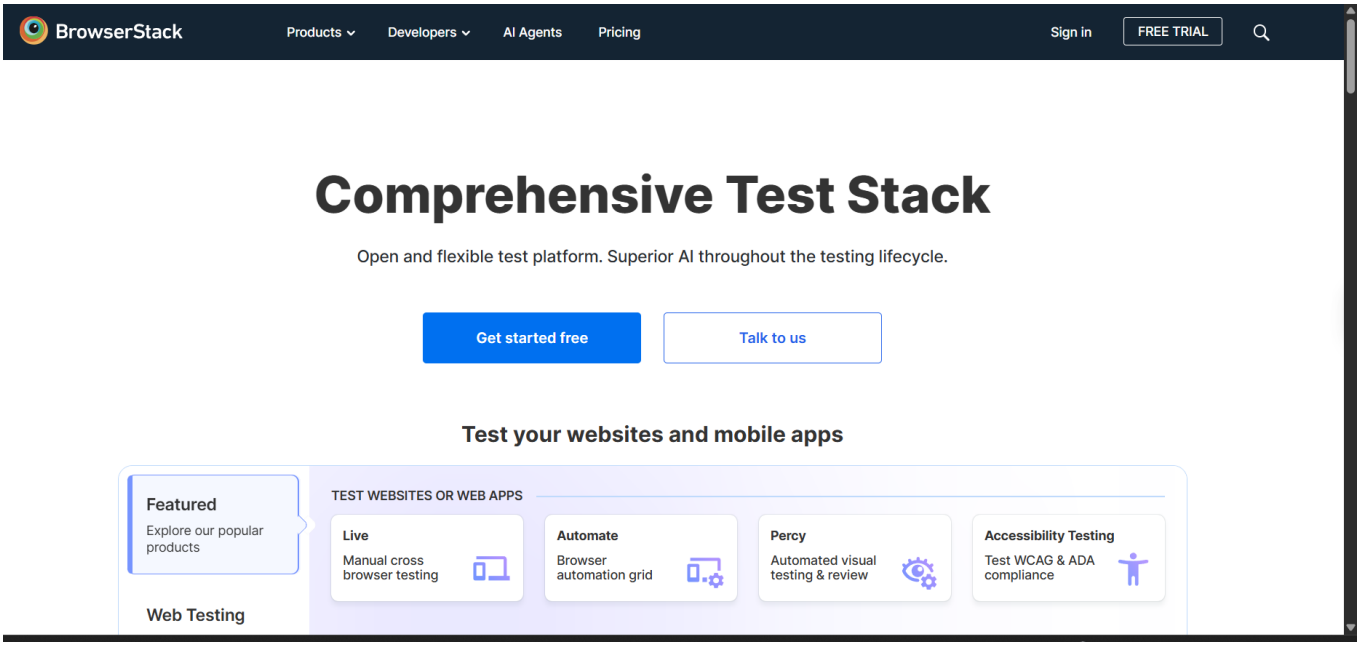


Account Setup

To begin the testing process, I first needed to set up a personal environment on the BrowserStack cloud platform. This involved creating an account and retrieving the necessary security credentials for the automation steps later.

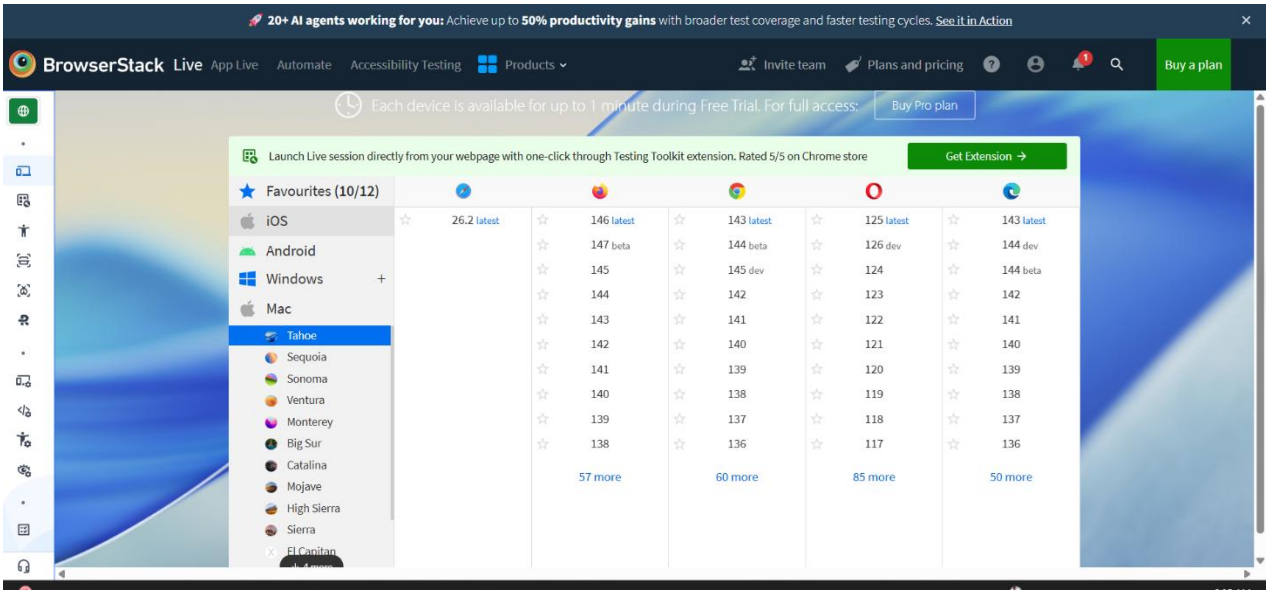
Step 1: Creating the Account

I visited the official BrowserStack website (https://www.browserstack.com) and signed up for a free trial. I used my Google account for a quick and secure registration process.



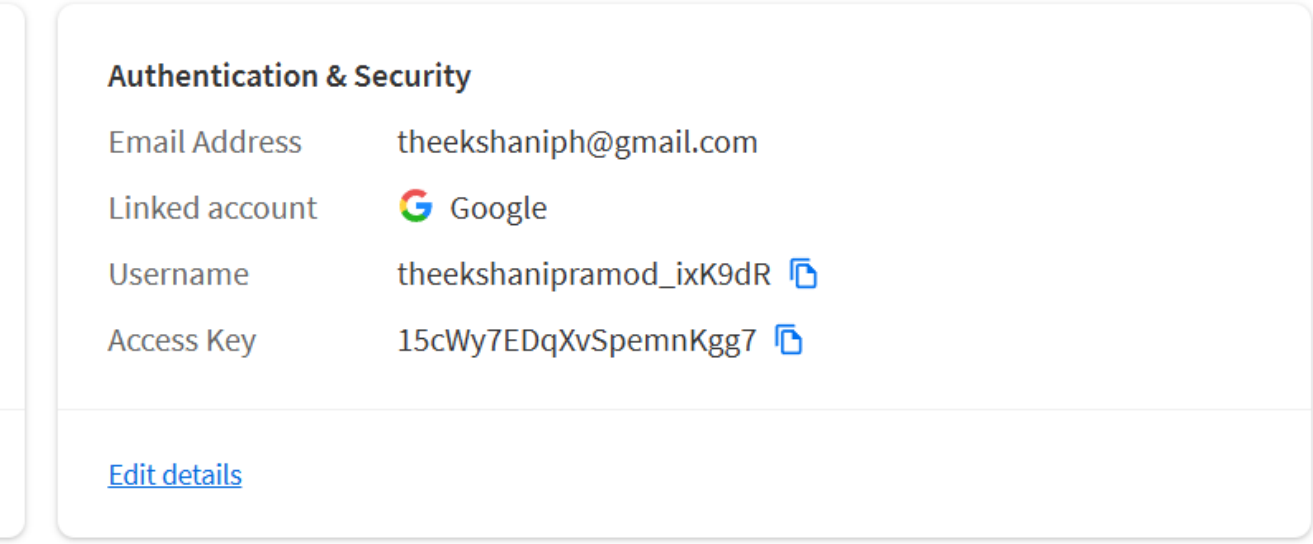
Step 2: Accessing the Dashboard

After successfully signing in, I was directed to the main Dashboard. This is the central hub where all testing tools—like "Live" manual testing and "Automate"—are located.



Step 3: Retrieving Access Credentials

For the automated testing part of this assignment (which I will cover in Section 5), I needed specific security keys to allow my Java code to talk to the BrowserStack servers. I navigated to **Profile > Settings & Usage** to find my "Username" and "Access Key."



Manual Web Testing (Live)

After setting up my account, I moved on to the first major task: **Manual Web Testing**. This feature is called "Live" on the BrowserStack dashboard.

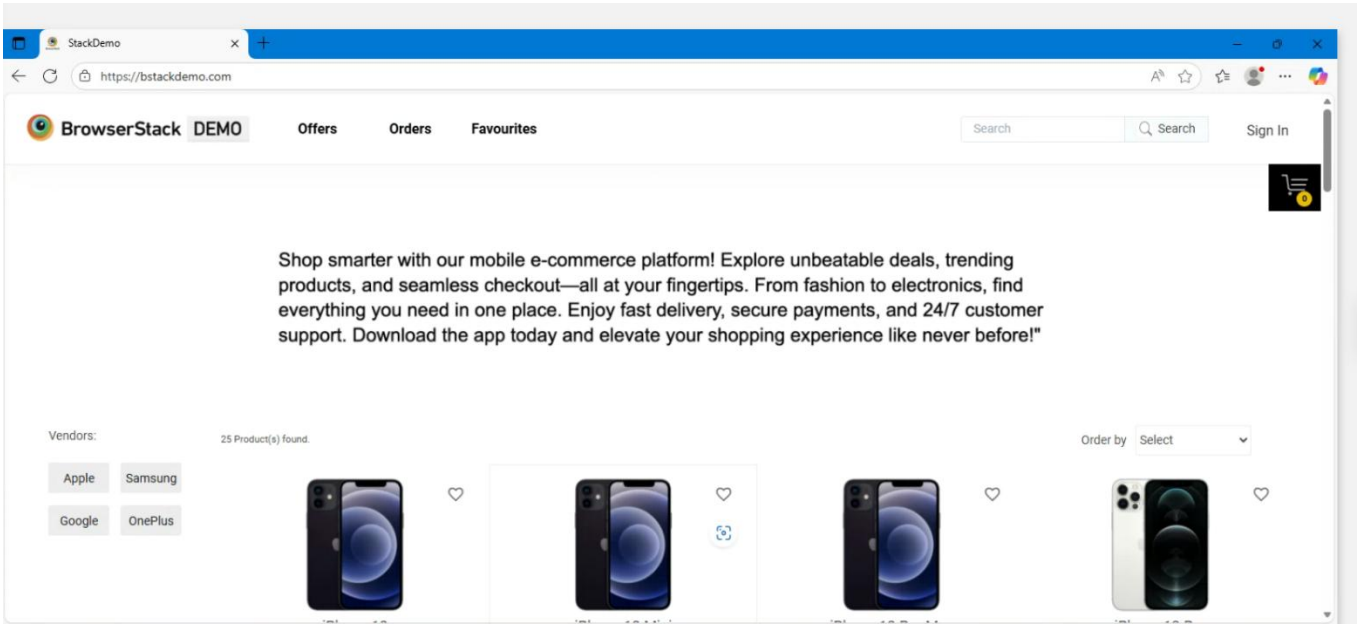
1. What is "Live" Testing?

The "Live" feature allows us to access a real computer located in the cloud. Instead of installing different browsers on my own laptop, I can simply select an Operating System (like Windows or macOS) and a Browser (like Chrome or Firefox) from the dashboard. BrowserStack then instantly opens a virtual machine with that exact configuration, allowing me to test websites as if I were using that specific computer.

For this assignment, I performed Cross-Browser Testing on a **Windows 11** environment using three different web browsers to ensure the test website (bstackdemo.com) works correctly on all of them.

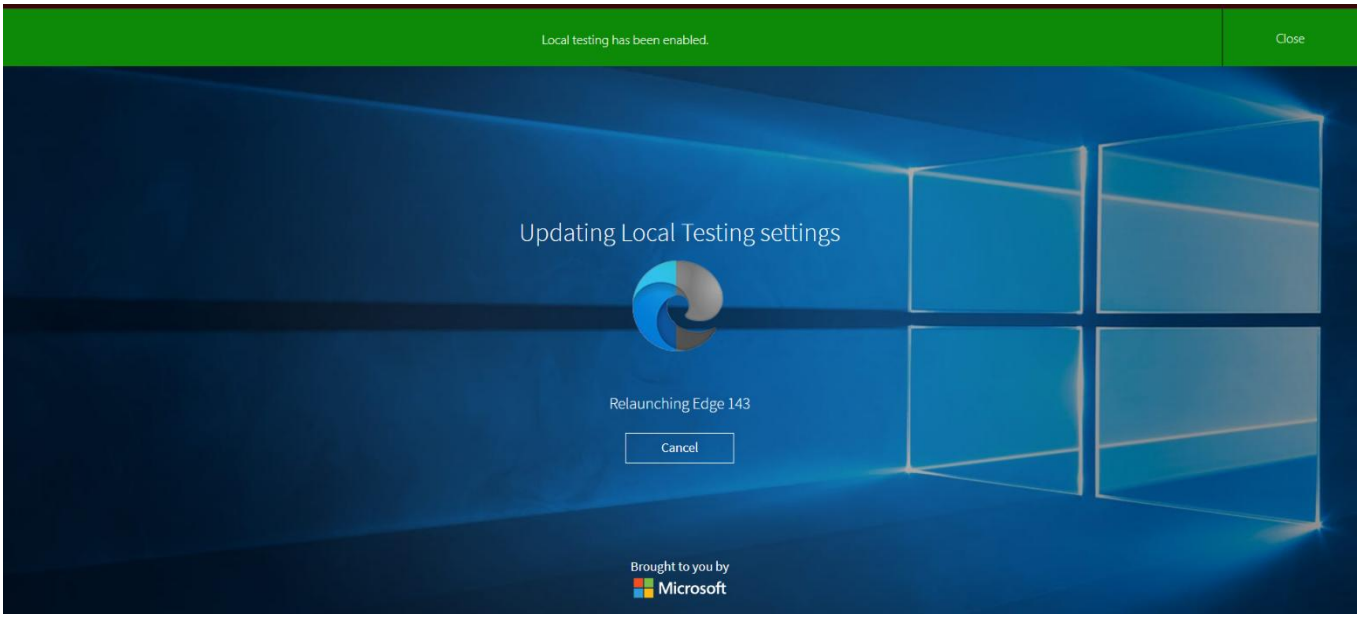
Test Case 1: Google Chrome on Windows 11

I selected **Windows 11** as the operating system and **Google Chrome** (latest version) as the browser. Once the session started, I navigated to the demo website bstackdemo.com. As shown in the screenshot below, the website loaded perfectly, and I was able to interact with the "Add to Cart" buttons without any lag.



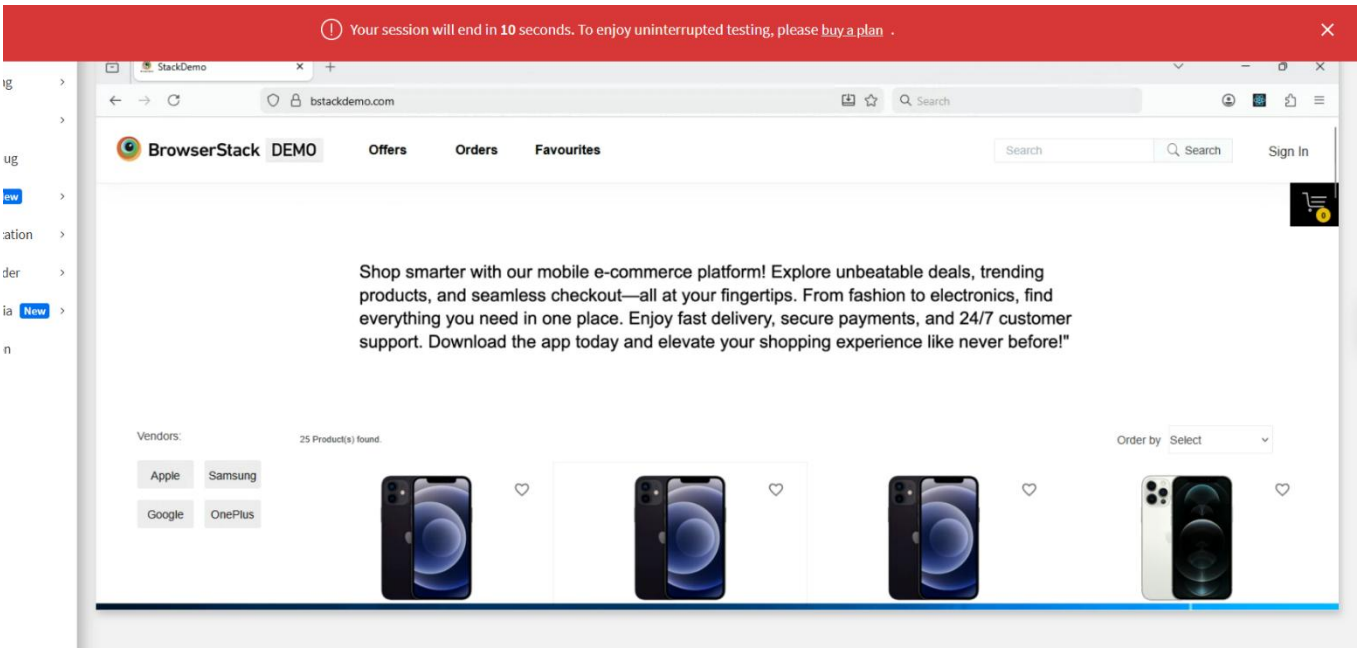
Test Case 2: Microsoft Edge on Windows 11

Next, I wanted to verify if the website performs well on Microsoft's default browser. I switched the configuration to **Microsoft Edge** on Windows 11. The site rendered correctly, and the layout (images and text) looked identical to the Chrome version.



Test Case 3: Mozilla Firefox on Windows 11

Finally, I tested the website on **Mozilla Firefox**. This is important because Firefox uses a different engine than Chrome and Edge. I launched a Firefox session on a Windows 11 machine in the cloud. The shopping cart functionality worked as expected, confirming that the website is compatible with this browser as well.



Mobile Web Testing

After verifying the website on desktop computers, I moved to the next important phase: **Mobile Web Testing**. Since most people today browse the internet using their phones, it is crucial to make sure the website looks good and works correctly on small touchscreens.

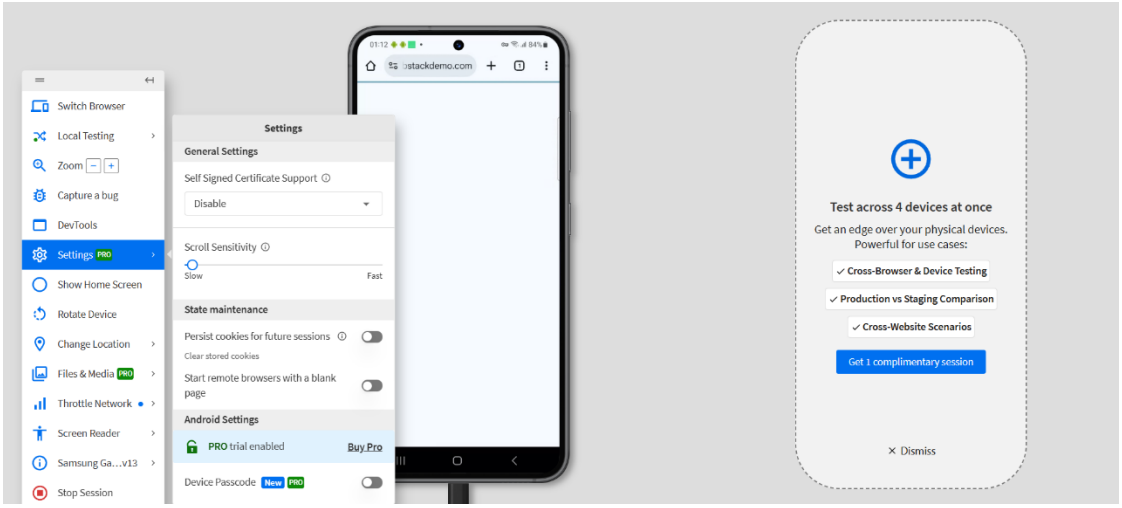
1. What is Mobile Web Testing?

This type of testing involves checking how a website performs on real mobile devices. In the BrowserStack "Live" dashboard, I can select actual physical phones (like iPhones or Samsung Galaxies) hosted in the cloud. This is much better than using a "simulator" on my computer because it shows exactly how the battery, screen size, and mobile browser affect the website in the real world.

Test Case: Samsung Galaxy S23 (Chrome Browser)

For this test, I selected a **Samsung Galaxy S23** from the Android list. I opened the **Google Chrome** browser on this phone and visited bstackdemo.com.

As seen in the screenshot below, the website successfully adapted to the mobile screen. The menu changed to a mobile-friendly version, the images were resized to fit perfectly, and I could scroll and tap on items just like I was holding the real phone in my hand.



Native App Testing (App Live)

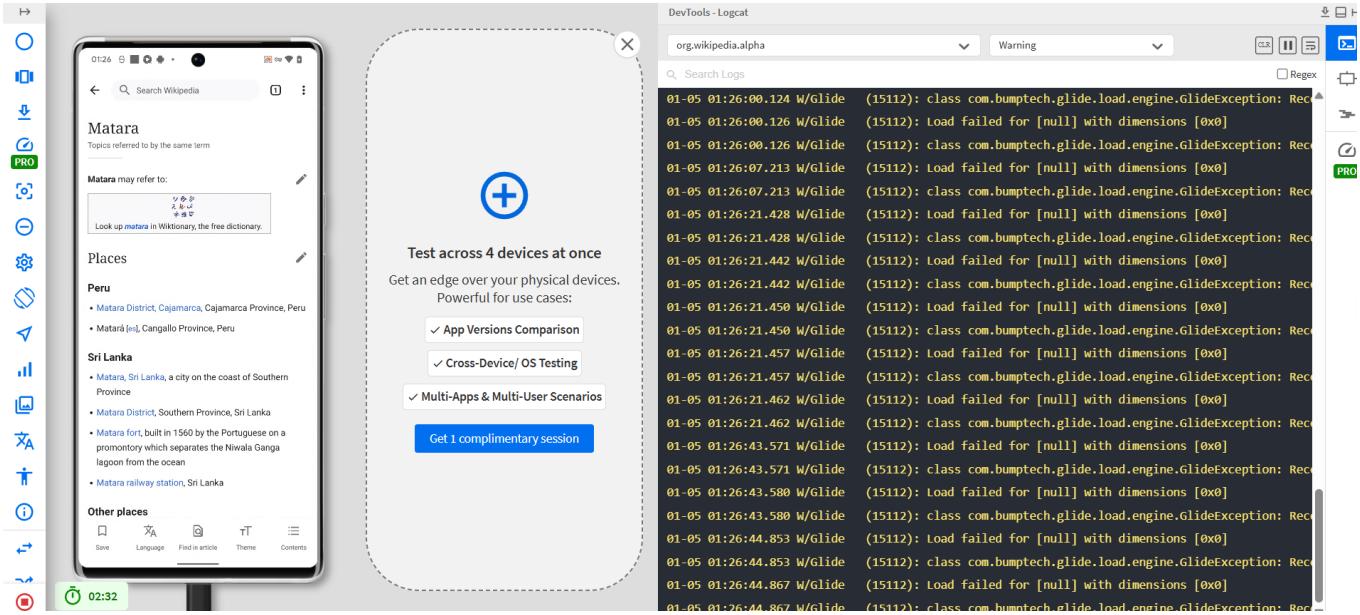
After testing websites, I moved on to **Native App Testing**. This is different from the previous step because instead of using a browser (like Chrome), I am testing a real application installed on the phone (like the .apk files we install on Android).

1. What is "App Live"?

"App Live" allows us to upload our own mobile applications (or use sample apps) and run them on real physical devices in the BrowserStack cloud. This is extremely useful because it helps us verify if an app crashes, if the buttons work, or if it looks good on different screen sizes without needing to buy every phone model.

Test Case 1: Wikipedia App on Samsung Galaxy S23

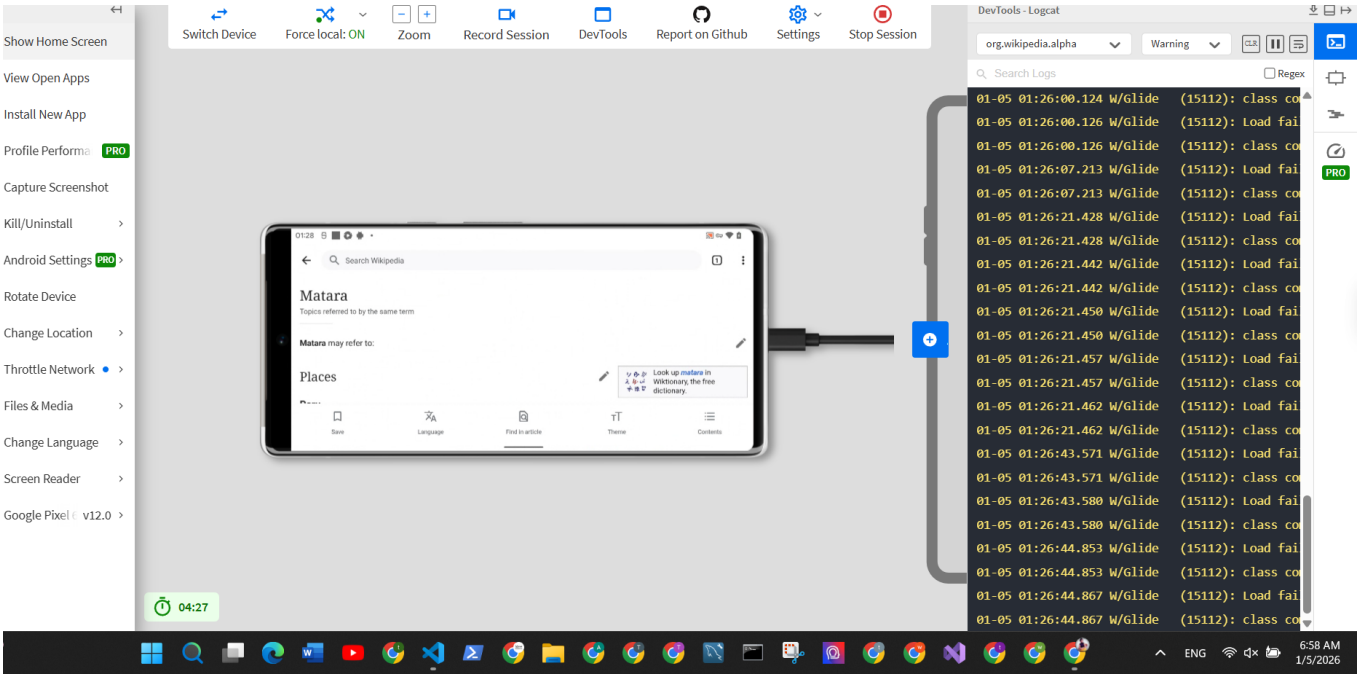
For this , I used the **"Wikipedia"** sample app provided by BrowserStack. I selected a **Samsung Galaxy S23** device. As shown in the screenshot below, the app installed instantly on the cloud device. I opened the app, clicked the search bar, and successfully searched for a topic. The app responded smoothly, just like it would on a personal phone.



Test Case 2: Testing Device Features (Screen Rotation)

One of the best features of BrowserStack is the **Toolbar**, which lets us simulate real-world actions. I wanted to test if the app works correctly when the user turns their phone sideways.

I used the **"Rotate Device"** button from the side toolbar. The phone screen immediately switched from Portrait (vertical) to Landscape (horizontal) mode. The Wikipedia app adjusted perfectly to the wider screen, proving that the app's layout is responsive.

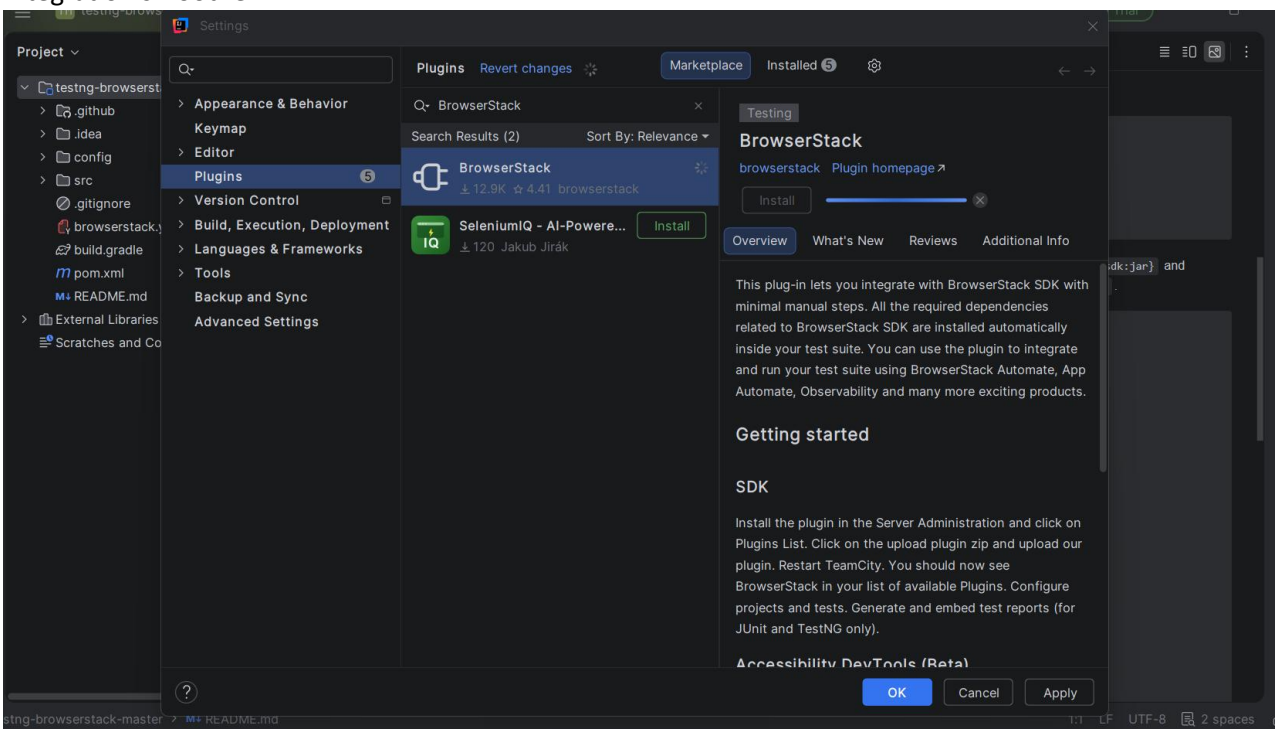


Automation Setup (Java)

After successfully completing the manual and mobile testing, the final phase of this was setting up **Automation Testing**. This involves writing code to run tests automatically on the BrowserStack cloud, rather than clicking manually.

1. **Prerequisites & Project Setup**

To begin, I ensured my development environment was ready. I used **IntelliJ IDEA** as my editor and **Maven** to manage the project dependencies.I verified that the project structure was correct, containing the necessary folders for the source code and test files. I also installed the **BrowserStack Plugin** for IntelliJ to make the integration smoother.



2. **Configuration (browserstack.yml)**

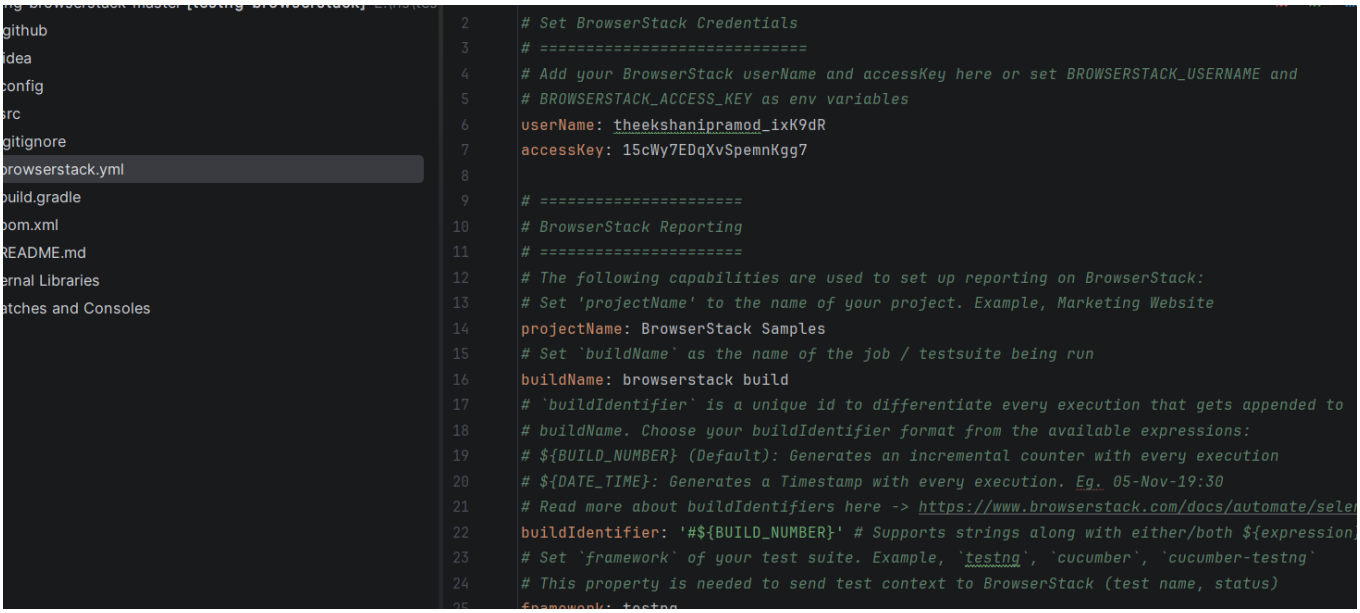
The most important part of the setup was connecting the code to my personal BrowserStack account. This is done using a configuration file named browserstack.yml.

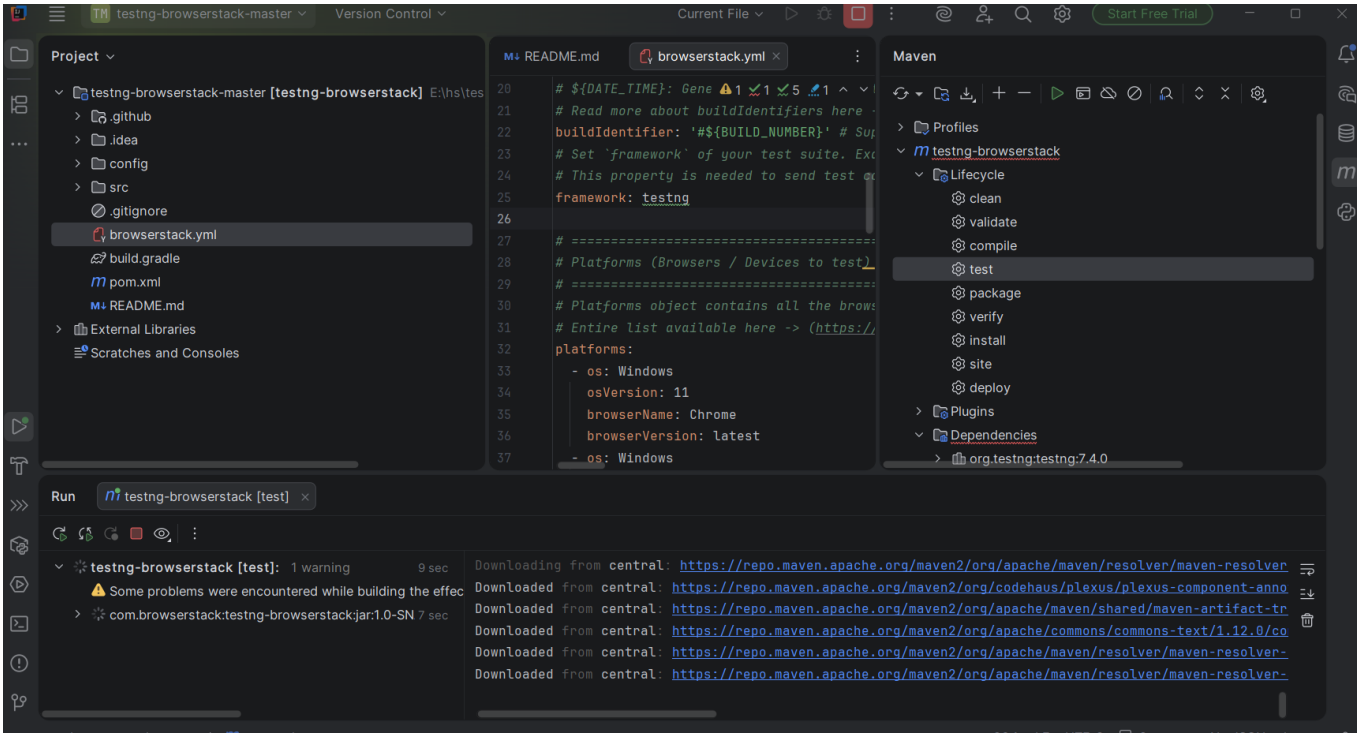
I opened this file and entered the **User Name** and **Access Key** that I generated in Step 1. This file acts as a "passport," allowing my code to log in to the BrowserStack cloud automatically.

In this file, I also defined the specific environments I wanted to test:

- **Operating System:** Windows 11
- **Browsers:** Chrome, Edge, and Firefox

```yaml
# Entire list available here -> (https://www.browserstack.com/list-of-browsers-
platforms:
  - os: Windows
    osVersion: 11
    browserName: Chrome
    browserVersion: latest
  - os: Windows
    osVersion: 11
    browserName: Edge
    browserVersion: latest
  - os: Windows
    osVersion: 11
    browserName: Firefox
    browserVersion: latest
    browserstackLocal: true
  - os: OS X
    osVersion: Big Sur
    browserName: Chrome
    browserVersion: latest
  - os: Windows
    osVersion: 10
    browserName: Edge
    browserVersion: latest
  - deviceName: Samsung Galaxy S22 Ultra
    browserName: chrome # Try 'samsung' for Samsung browser
    osVersion: 12.0
    buildName: browserstack-build-1
```





Execution & Results

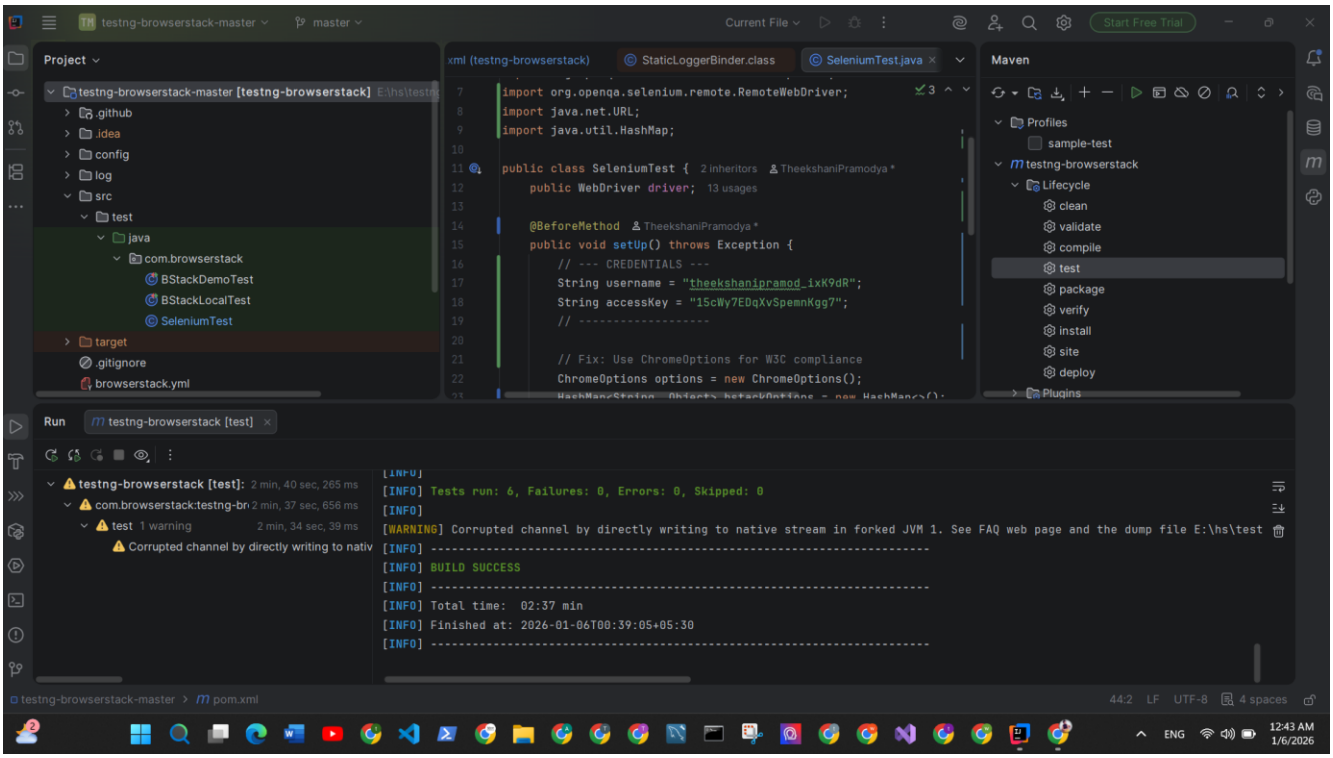
Once the configuration was complete, I proceeded to run the automation script to verify that the code could successfully communicate with the BrowserStack cloud and execute the tests on real devices.

1. Test Execution (Console Output)

I executed the Maven test command from my development environment. The system compiled the code, connected to the BrowserStack servers, and began the test sequence.

After a few minutes of execution, the console confirmed that the build was successful. The "BUILD SUCCESS" message indicated that the script ran without any syntax errors or connection timeouts.

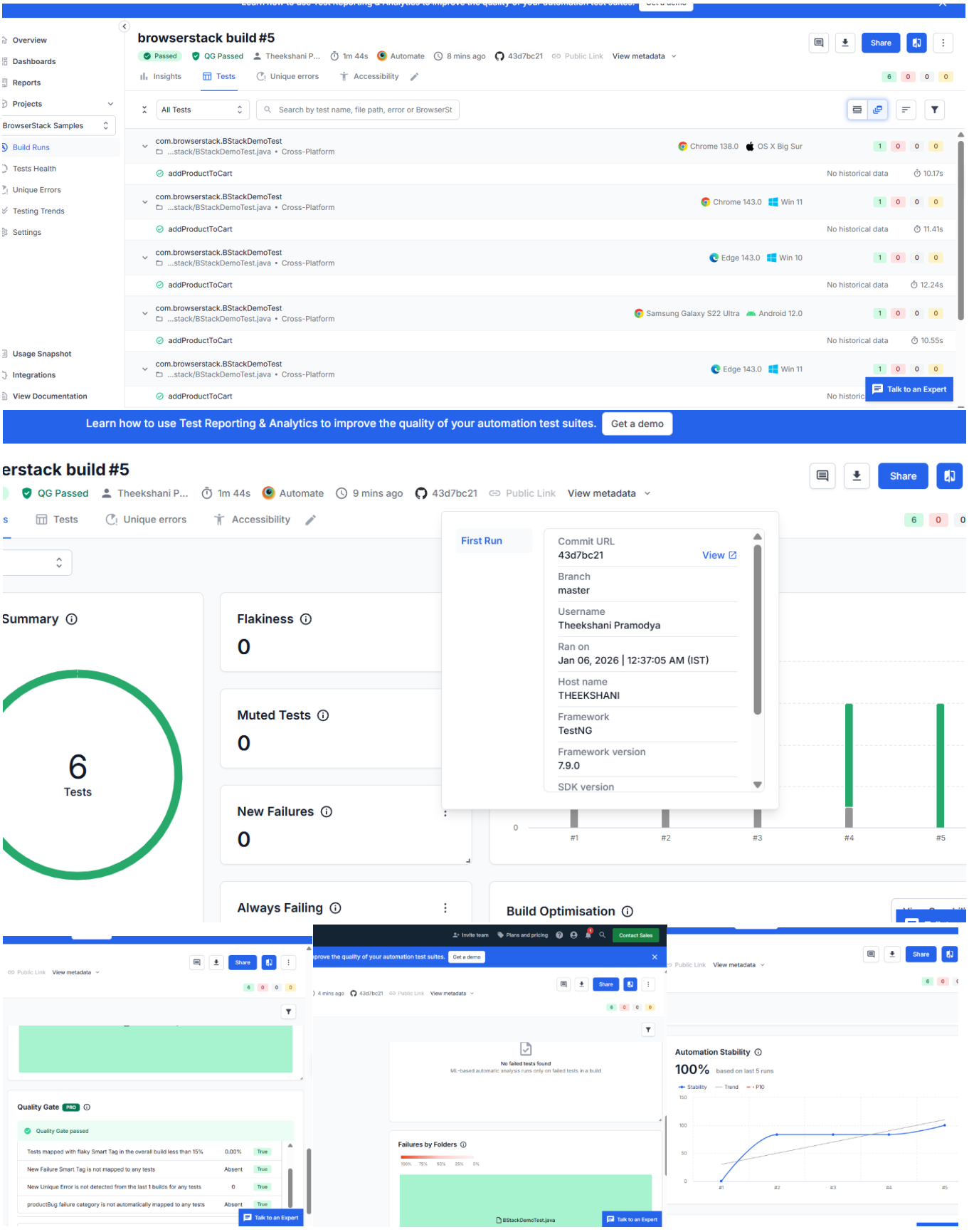
```
[INFO]
[INFO] ----------------< com.browserstack:testng-browserstack >----------------
[INFO] Building testng-browserstack 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- clean:3.2.0:clean (default-clean) @ testng-browserstack ---
[INFO] Deleting E:\hs\testng-browserstack-master\target
[INFO]
[INFO] --- dependency:3.6.0:properties (getClasspathFilenames) @ testng-browserstack ---
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ testng-browserstack ---
[INFO] skip non existing resourceDirectory E:\hs\testng-browserstack-master\src\main\resources
[INFO]
[INFO] --- compiler:3.11.0:compile (default-compile) @ testng-browserstack ---
[INFO] No sources to compile
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ testng-browserstack ---
[INFO] skip non existing resourceDirectory E:\hs\testng-browserstack-master\src\test\resources
[INFO]
[INFO] --- compiler:3.11.0:testCompile (default-testCompile) @ testng-browserstack ---
[INFO] Changes detected - recompiling the module! :source
[INFO] Compiling 3 source files with javac [debug target 1.8] to target\test-classes
[WARNING] bootstrap class path not set in conjunction with -source 8
[INFO]
[INFO] --- surefire:3.1.2:test (default-test) @ testng-browserstack ---
[INFO] Using auto detected provider org.apache.maven.surefire.testng.TestNGProvider
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running TestSuite
```



3. Cloud Dashboard Results

To confirm the actual test results, I logged into the **BrowserStack Automate Dashboard**.

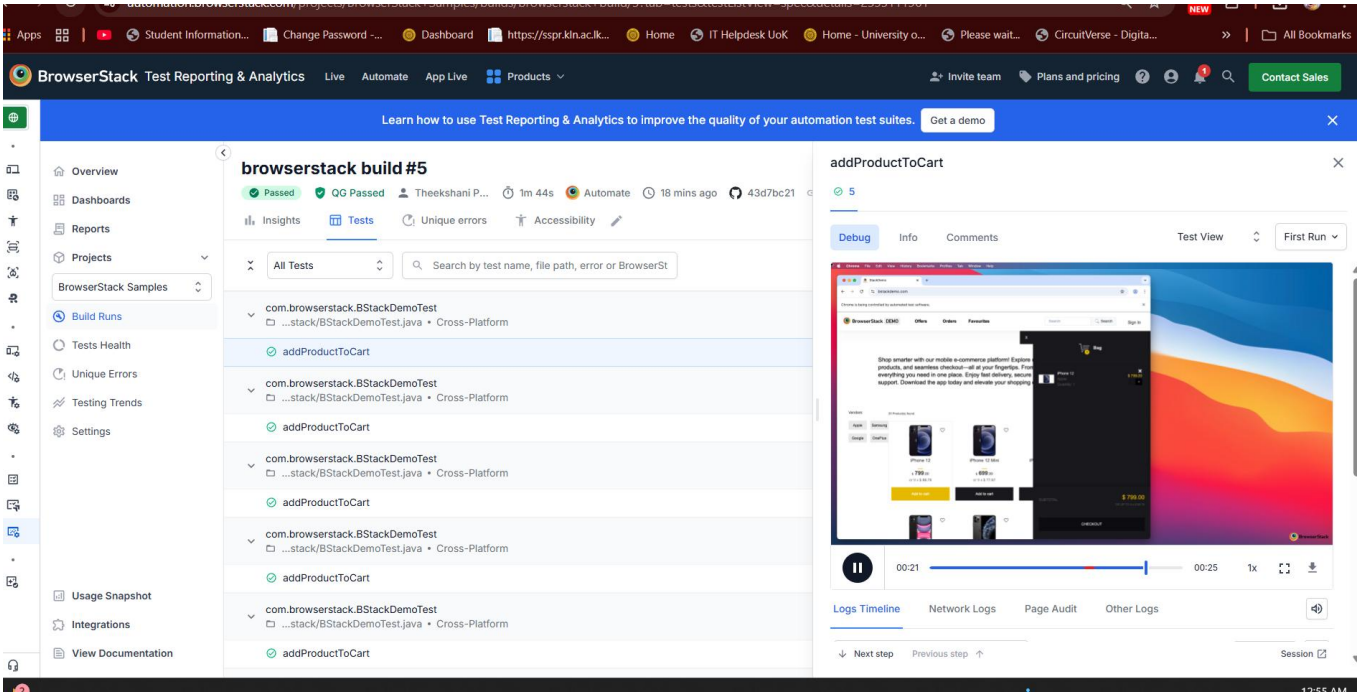
The dashboard showed **Build #5** with a status of **Passed**. A large green circle confirmed that all **6 tests** (covering different browsers like Chrome, Edge, and mobile devices like the Samsung Galaxy S22 Ultra) completed successfully. This proves that the automation script works perfectly across different platforms.

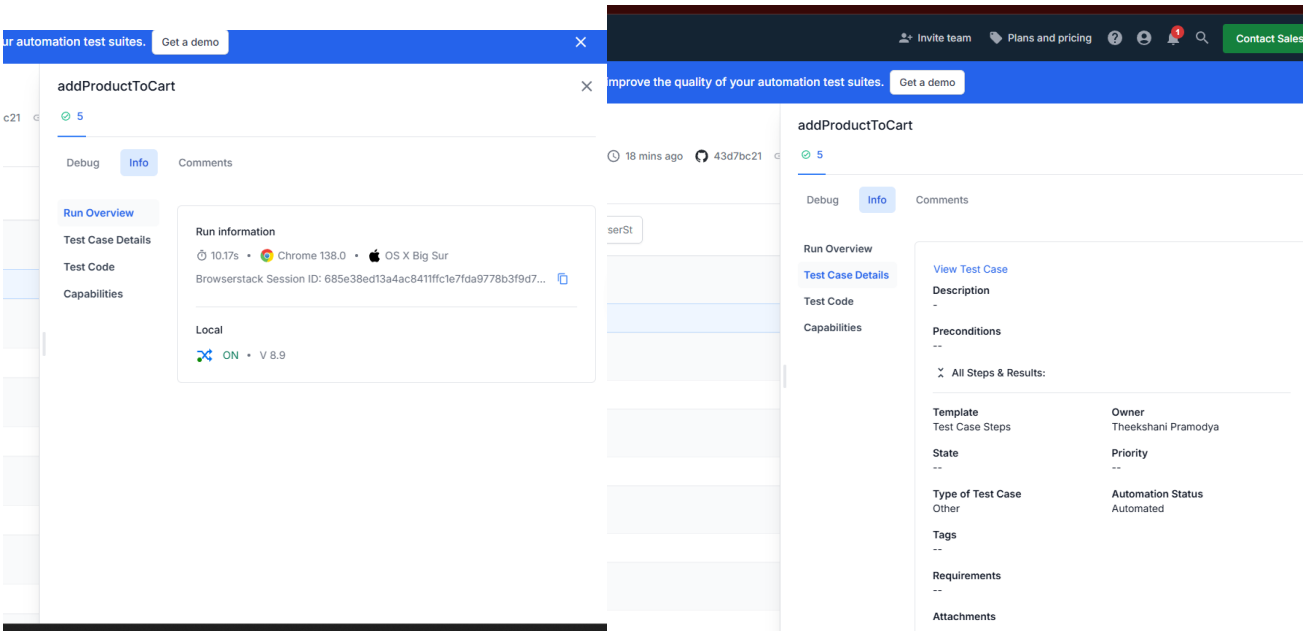

browserstack build #5
Passed
QG Passed
Theekshani P...
1m 44s
Automate
8 mins ago
43d7bc21
Public Link
View metadata
Share
Insights
Tests
Unique errors
Accessibility
Overview
Dashboards
Reports
Projects
BrowserStack Samples
Build Runs
Tests Health
Unique Errors
Testing Trends
Settings
Usage Snapshot
Integrations
View Documentation
All Tests
Search by test name, file path, error or BrowserSt
com.browserstack.BStackDemoTest
...stack/BStackDemoTest.java • Cross-Platform
Chrome 138.0
OS X Big Sur
addProductToCart
No historical data
10.17s
Chrome 143.0
Win 11
11.41s
Edge 143.0
Win 10
12.24s
Samsung Galaxy S22 Ultra
Android 12.0
10.55s
Edge 143.0
Win 11
Talk to an Expert
Learn how to use Test Reporting & Analytics to improve the quality of your automation test suites.
Get a demo
First Run
Commit URL
43d7bc21
View
Branch
master
Username
Theekshani Pramodya
Ran on
Jan 06, 2026 | 12:37:05 AM (IST)
Host name
THEEKSHANI
Framework
TestNG
Framework version
7.9.0
SDK version
Summary
6
Tests
Flakiness
0
Muted Tests
0
New Failures
0
Always Failing
Build Optimisation
Quality Gate
Quality Gate passed
Failures by Folders
Automation Stability
100%
based on last 5 runs

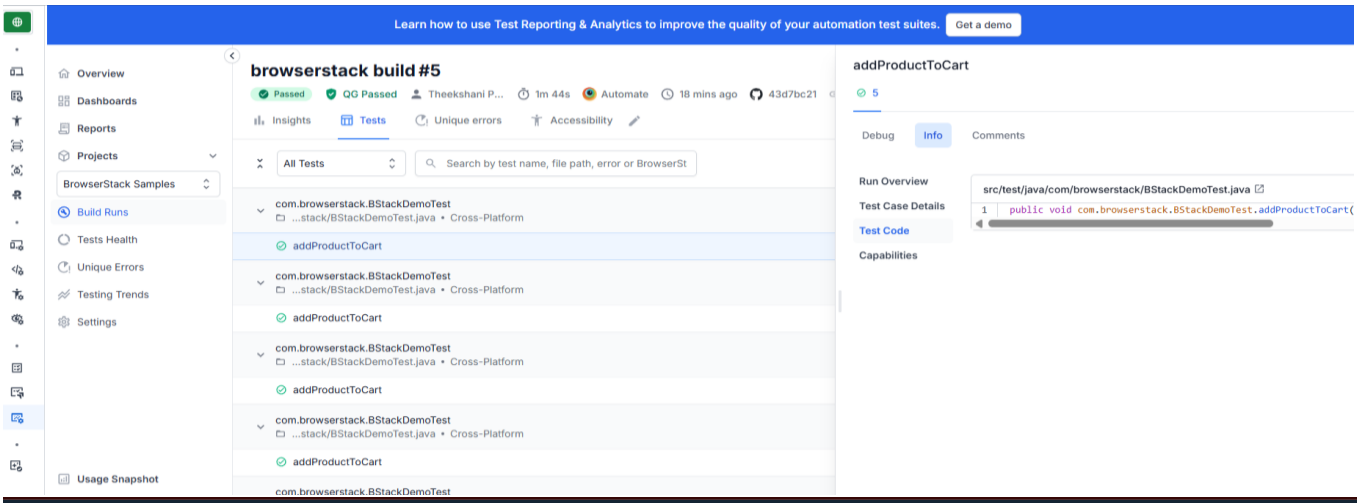
4. Verification via Video Logs

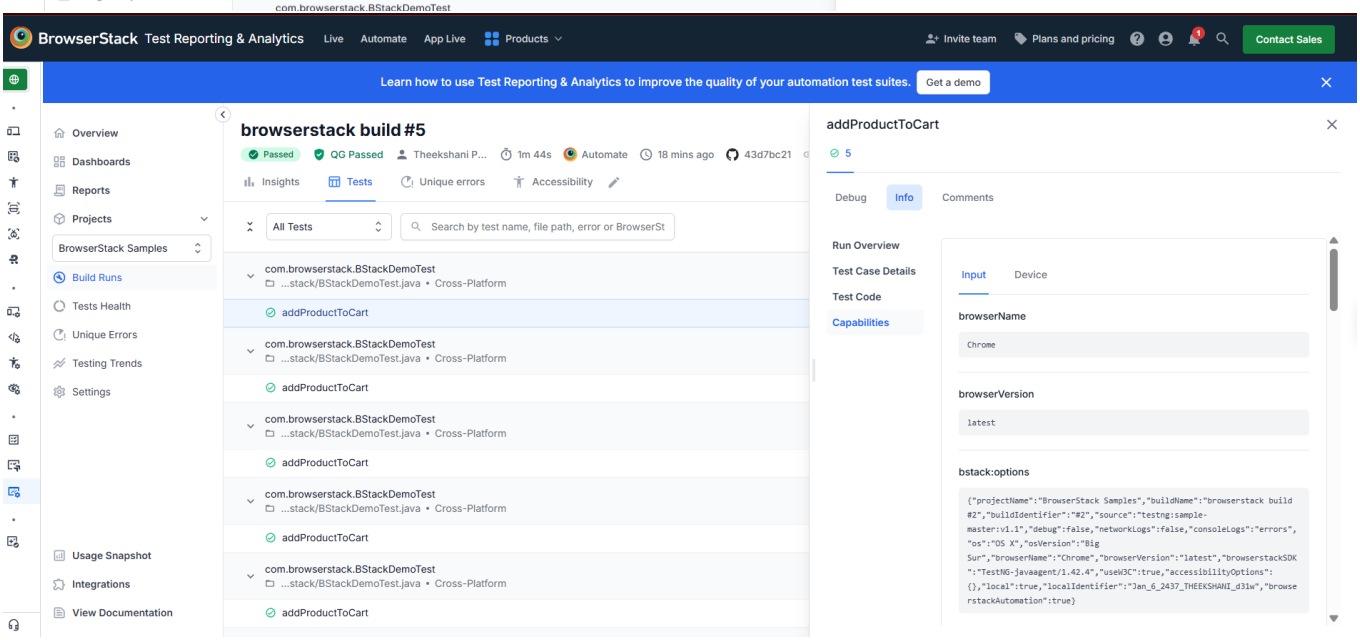
Finally, I clicked on one of the individual test sessions to verify the execution steps. BrowserStack provided a full video recording of the test running on the remote device.

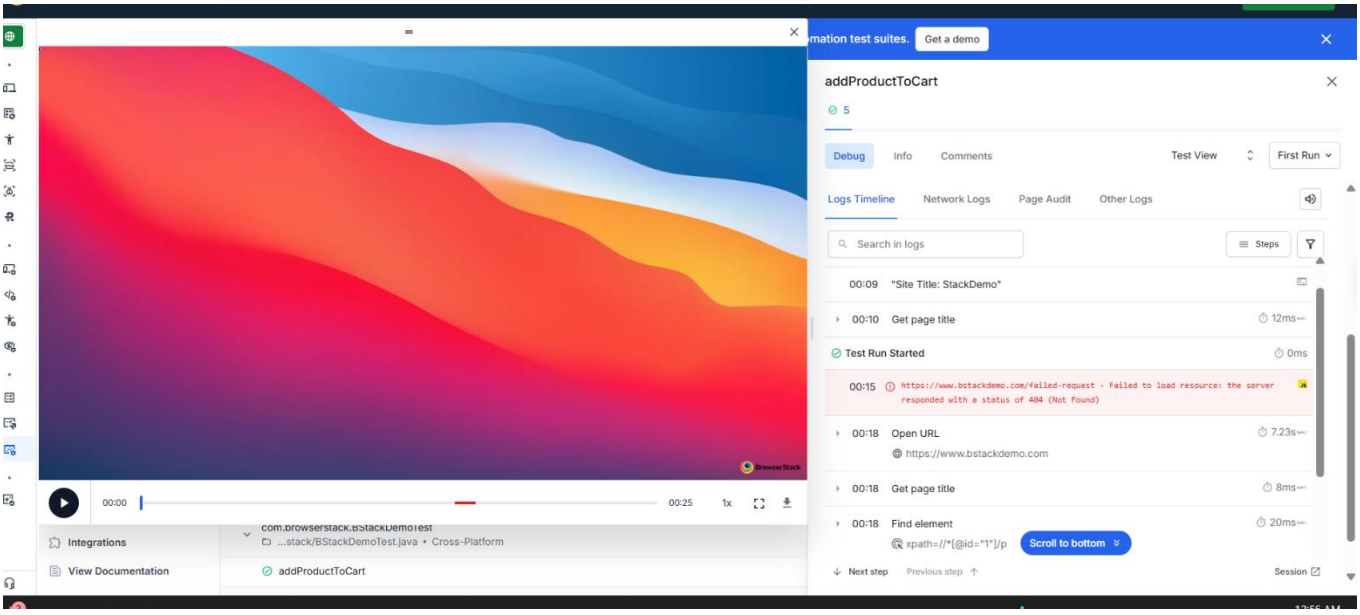
On the screen, I could watch the automated browser open the *StackDemo* website, add a product to the cart, and complete the test steps, exactly as if a human were doing it. The text logs on the right side confirmed every action was successful.

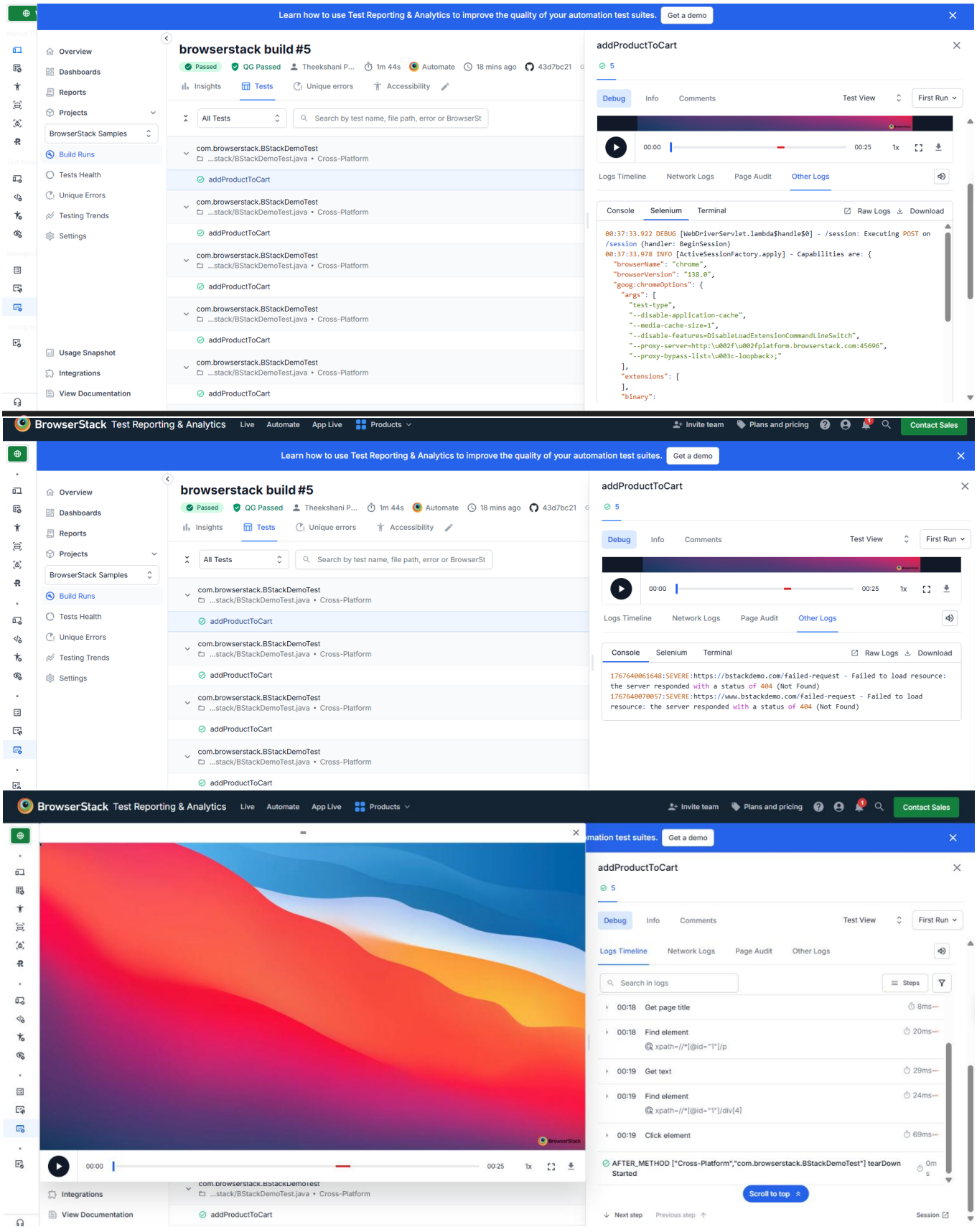





Learn how to use Test Reporting & Analytics to improve the quality of your automation test suites.
Get a demo
browserstack build #5
Passed
QG Passed
Theekshani P...
1m 44s
Automate
18 mins ago
43d7bc21
Insights
Tests
Unique errors
Accessibility
addProductToCart
Debug
Info
Comments
Run Overview
Test Case Details
Test Code
Capabilities
src/test/java/com/browserstack/BStackDemoTest.java
Overview
Dashboards
Reports
Projects
BrowserStack Samples
Build Runs
Tests Health
Unique Errors
Testing Trends
Settings
Usage Snapshot
com.browserstack.BStackDemoTest
addProductToCart


BrowserStack Test Reporting & Analytics
Live
Automate
App Live
Products
Invite team
Plans and pricing
Contact Sales
Learn how to use Test Reporting & Analytics to improve the quality of your automation test suites.
Get a demo
browserstack build #5
addProductToCart
Input
Device
browserName
Chrome
browserVersion
latest
bstack:options
Usage Snapshot
Integrations
View Documentation


Get a demo
addProductToCart
Debug
Info
Comments
Test View
First Run
Logs Timeline
Network Logs
Page Audit
Other Logs
Search in logs
Steps
00:09 "Site Title: StackDemo"
00:10 Get page title
12ms
Test Run Started
0ms
00:15
https://www.bstackdemo.com/failed-request - Failed to load resource: the server responded with a status of 404 (Not Found)
00:18 Open URL
7.23s
https://www.bstackdemo.com
00:18 Get page title
8ms
00:18 Find element
20ms
Scroll to bottom
Next step
Previous step
Session
00:00
00:25
1x
Integrations
View Documentation


Learn how to use Test Reporting & Analytics to improve the quality of your automation test suites.
Get a demo
browserstack build #5
Passed
QG Passed
Theekshani P...
1m 44s
Automate
18 mins ago
43d7bc21
Insights
Tests
Unique errors
Accessibility
Overview
Dashboards
Reports
Projects
BrowserStack Samples
Build Runs
Tests Health
Unique Errors
Testing Trends
Settings
Usage Snapshot
Integrations
View Documentation
All Tests
Search by test name, file path, error or BrowserSt
com.browserstack.BStackDemoTest
...stack/BStackDemoTest.java • Cross-Platform
addProductToCart
addProductToCart
Debug
Info
Comments
Test View
First Run
00:00
00:25
1x
Logs Timeline
Network Logs
Page Audit
Other Logs
Console
Selenium
Terminal
Raw Logs
Download
00:37:33.922 DEBUG [WebDriverServlet.lambda$handle$0] - /session: Executing POST on /session (handler: BeginSession)
00:37:33.978 INFO [ActiveSessionFactory.apply] - Capabilities are: {
"browserName": "chrome",
"browserVersion": "138.0",
"goog:chromeOptions": {
"args": [
"test-type",
"--disable-application-cache",
"--media-cache-size=1",
"--disable-features=DisableLoadExtensionCommandLineSwitch",
"--proxy-server=http:\u002f\u002fplatform.browserstack.com:45696",
"--proxy-bypass-list=\u003c-loopback>;"
],
"extensions": [
],
"binary":
BrowserStack Test Reporting & Analytics
Live
Automate
App Live
Products
Invite team
Plans and pricing
Contact Sales
1767640061648:SEVERE:https://bstackdemo.com/failed-request - Failed to load resource: the server responded with a status of 404 (Not Found)
1767640070057:SEVERE:https://www.bstackdemo.com/failed-request - Failed to load resource: the server responded with a status of 404 (Not Found)
Search in logs
Steps
00:18 Get page title 8ms
00:18 Find element 20ms
xpath=//*[@id="1"]/p
00:19 Get text 29ms
00:19 Find element 24ms
xpath=//*[@id="1"]/div[4]
00:19 Click element 69ms
AFTER_METHOD ["Cross-Platform","com.browserstack.BStackDemoTest"] tearDown Started 0m s
Scroll to top
Next step
Previous step
Session

Conclusion

Completing this has given a clear understanding of how modern software testing works in the real world. By using **BrowserStack**, I was able to move beyond simple manual testing on my own laptop and access a massive library of devices in the cloud.

The Power of Automation

One of the biggest takeaways from this project was seeing the difference between **Manual** and **Automated** testing.

- **Manual Testing:** When I manually tested the *StackDemo* website on Chrome and Firefox, I had to physically click every button and type every input myself. While effective, it was slow and repetitive.

- **Automated Testing:** In contrast, setting up the Java automation script allowed me to write the test once and run it instantly. The "Green Circle" on my dashboard proved that the computer could do the work much faster than I could, without making human errors. This showed me why companies prefer automation for large projects.

Access to Real Devices

A major highlight was the ability to test on real physical devices that I do not own, such as the **Samsung Galaxy S22 Ultra**. Instead of using a "simulated" phone which might hide bugs, BrowserStack connected me to a real device in their data center. This gave me confidence that the application would work correctly for actual users, regardless of what device they are holding.

Overall, this assignment demonstrated that cloud testing platforms are essential for ensuring high-quality software. Mastering tools like IntelliJ, Maven, and BrowserStack has equipped me with valuable skills for the future, enabling me to deliver reliable applications across any browser or operating system.