Mobile Automation
Understanding Document

Version 1.0
Last Updated: 29/05/2020

Control History

Reference Documents:

Document Name and Link
WebDriverIO Official Link For Setup: https://webdriver.io/docs/gettingstarted.html
Webdriver.io api: https://webdriver.io/docs/api.html
For element Locators: https://webdriver.io/docs/api.html
Appium Service: https://webdriver.io/docs/appium-service.html
Appium Setup: http://appium.io/docs/en/about-appium/getting-started/?lang=en
Node JS Setup: https://nodejs.org/en/
Android Studio: https://developer.android.com/studio

Table of Contents

Definition 4
Introduction: 4
Main Features of WebdriverIO: 4
Objective: 4
Software Tools Required: 4
Issues: 5
Installation Procedure: 5
Creating Webdriver IO Appium project: 6
Configuring wdio.config.js file: 11
Specifying Capabilities: 11
Configuring Reports: 12
Screenshots Generation: 12
Configure Appium Server: 12
Babel Configuration 13
Executing WDIO file: 13
Code: 14
Execution: 14
To Generate Allure Reports: 15
Allure Reports 15
Cucumber Reports 16
Project Folder View 17
Testing Specifications: 17
Framework Structure: 17
Appium Mocha Framework: 17
Appium Cucumber Framework: 19
Pros and Cons: 20
Pros: 20
Cons: 20
Sample Test Application 21
Test Scenarios Covered: 21
Comparison with other tools: 22
Advantages over Other tools: 22

Definition:

Mobile application **testing** is a process by which application software developed for handheld **mobile** devices is tested for its functionality. Automation is the process whereby one automates testing of various applications, either App or browser version - in this case a mobile application – which can be app or mobile browsers. Testing of these applications is achieved by using automation tools which in turn reduces testing time cycle.

Introduction:

WebdriverIO is a custom implementation for selenium's W3C webdriver API. It is coded in JavaScript and packaged into 'npm' and runs on Node. js.

Main Features of WebdriverIO:

- WebdriverIO is a good automation tool, which can automate both web applications and native mobile Apps.
- The integrated test runner let you write asynchronous commands in a synchronous way so that you do not need to care about how to handle a Promise to avoid racing conditions.
- WebdriverIO integrates easily with the CrossBrowser-Testing platform, so that we can perform tests on a wide variety of OS/Device/Browser combinations, all from one test.
- WebdriverIO currently supports mocha, Jasmine, cucumber frameworks.

Objective:

To perform evaluation of Mobile Automation Testing using the WebdriverIO written in Java Script and Packaged into npm and runs on Node.js.

Software Tools Required:

- Appium
- Android Studio (requires JDK 8)
- NodeJS -12v
- WebdriverIO – 6.1.12
    - Chromedriver Service
    - Appium Server Service
    - Reporters
        - Allure Reports
        - HTML Reporter
        - Spec Reporter
        - Dot Reporter

Issues:

In-case if you get any error w.r.t installing a package open cmd and type : ping registry.npmjs.org

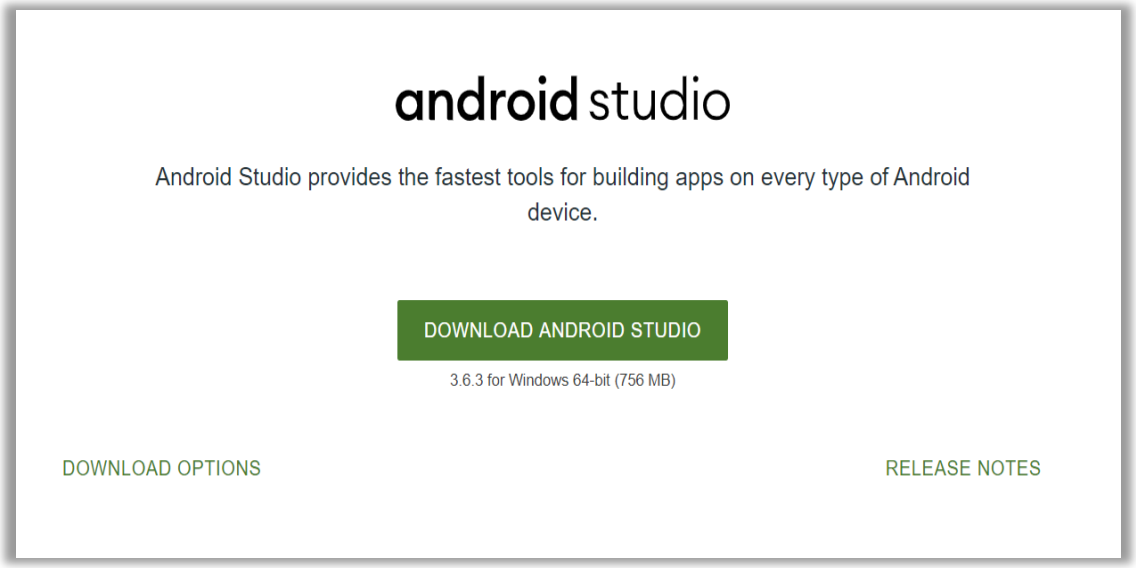
Installation Procedure:

- **Appium**:

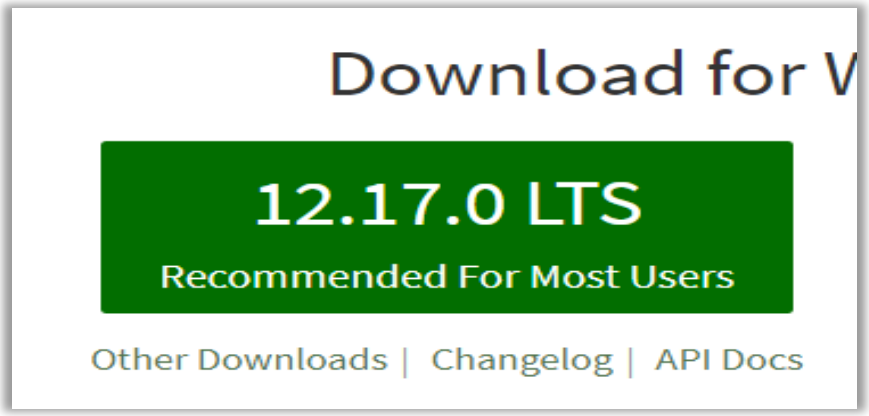
Install Appium Globally:

```
npm install -g appium
```

- **Android Studio with SDK tools:**



- **Node JS: require version 12**



WebdriverIO : Version 6

For reference: https://webdriver.io/docs/gettingstarted.html

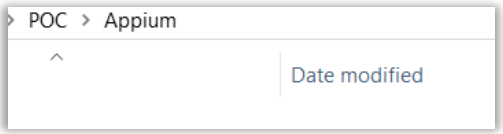
To install: npm i --save-dev @wdio/cli

To crate auto config file: npx wdio config –y

Note: the above code creates **wdio.conf.js** file in the directory, which run on desktop applications

Creating Webdriver IO Appium project:

Create a new Folder:



- Open cmd here:

```
C:\POC\Appium>
```

- Now run command: npm init –y

It should generate below output

```
C:\POC\Appium>npm init -y
Wrote to C:\POC\Appium\package.json:

{
  "name": "Appium",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

- Once the package.json file is created, install wdio:
  - npm i --save-dev @wdio/cli

```
C:\POC\Appium>npm i --save-dev @wdio/cli
[...................] / rollbackFailedOptional: verb npm-session f2abd7bf4ba2fa08
```

Finally, you should see this:

```
C:\POC\Appium>npm i --save-dev @wdio/cli

> ejs@3.1.3 postinstall C:\POC\Appium\node_modules\ejs
> node --harmony ./postinstall.js

Thank you for installing EJS: built with the Jake JavaScript build tool (https://jakejs.com/)

npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@~2.1.2 (node_modules\chokidar\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.1.3: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
npm WARN Appium@1.0.0 No description
npm WARN Appium@1.0.0 No repository field.

+ @wdio/cli@6.1.12
added 266 packages from 252 contributors and audited 267 packages in 18.595s

20 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

- Now we need to create wdio file with appium settings run: npx wdio config
  - Select local.

```
C:\POC\Appium>npx wdio config

==========================
WDIO Configuration Helper
==========================

? Where should your tests be launched? (Use arrow keys)
> local
```

- Select on my local machine

```
C:\POC\Appium>npx wdio config

=========================
WDIO Configuration Helper
=========================

? Where should your tests be launched? local
? Where is your automation backend located? (Use arrow keys)
> On my local machine
  In the cloud using Experitest
  In the cloud using Sauce Labs
  In the cloud using Browserstack or Testingbot or LambdaTest or a different service
  I have my own Selenium cloud
```

- Select type of framework : mocha

```
? Which framework do you want to use? (Use arrow keys)
> mocha
  jasmine
  cucumber
```

- Select execution type: sync

```
? Do you want to run WebdriverIO commands synchronous or asynchronous? (Use arrow keys)
> sync
  async
```

- Specify the location where your test scripts are located, if you want to use default location press enter.

```
? Where are your test specs located? (./test/specs/**/*.js)
```

- Select the type of reports used: spec, dot, allure

```
? Which reporter do you want to use?
 (*) spec
 (*) dot
 ( ) junit
>(*) allure
 ( ) sumologic
 ( ) concise
 ( ) reportportal
(Move up and down to reveal more choices)
```

- Select type of services required for the project: Chromedriver, Appium

```
? Do you want to add a service to your test setup?
 ( ) intercept
 ( ) docker
 ( ) visual-regression-testing
>(*) chromedriver
 ( ) sauce
 ( ) testingbot
 ( ) selenium-standalone
(Move up and down to reveal more choices)
```

```
? Do you want to add a service to your test setup?
 ( ) devtools
 ( ) applitools
 ( ) browserstack
>(*) appium
 ( ) firefox-profile
 ( ) crossbrowsertesting
 ( ) lambdatest
(Move up and down to reveal more choices)
```

- Specify the URL of the website or press enter:

```
? What is the base url? (http://localhost)
```

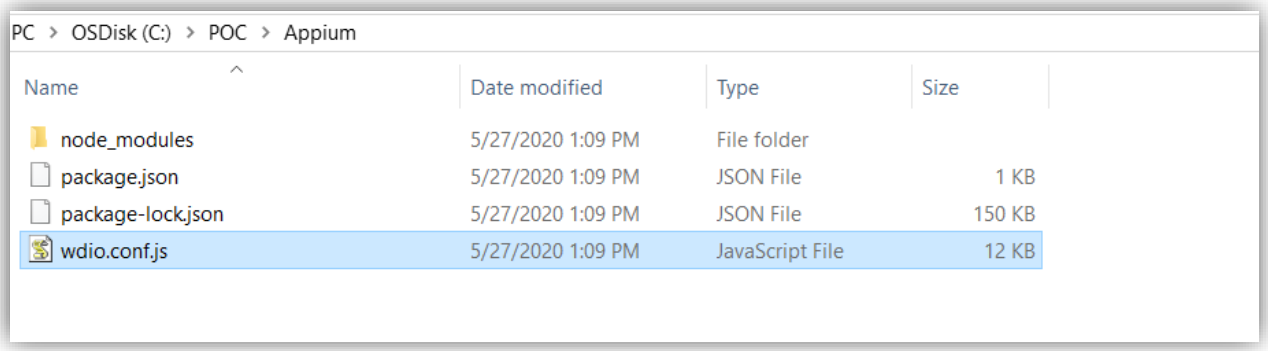
- Then the installation process will start which will download all the specified modules for the project.

```
? What is the base url? http://localhost

Installing wdio packages:
- @wdio/local-runner
- @wdio/mocha-framework
- @wdio/spec-reporter
- @wdio/dot-reporter
- @wdio/allure-reporter
- wdio-chromedriver-service
- @wdio/appium-service
- @wdio/sync
- chromedriver
[...................] / rollbackFailedOptional: verb npm-session db66a438676e3dd9
```

```
+ chromedriver@83.0.0
+ wdio-chromedriver-service@6.0.3
+ @wdio/mocha-framework@6.1.8
+ @wdio/spec-reporter@6.1.12
+ @wdio/sync@6.1.8
+ @wdio/dot-reporter@6.1.9
+ @wdio/local-runner@6.1.12
+ @wdio/appium-service@6.1.0
+ @wdio/allure-reporter@6.1.12
added 189 packages from 133 contributors and audited 456 packages in 25.796s

35 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities


Packages installed successfully, creating configuration file...

Configuration file was created successfully!
To run your tests, execute:
$ wdio run wdio.conf.js
```

- Installation is completed.

- wdio.conf.js should be created.



Open the config file.

Configuring wdio.config.js file:

- Here you need to specify capabilities (App or Browser
- Need to configure reports (additional options for report generation).
- Screenshots generation (Allure reports and HTML reporter)
- Adding Babel for ES6 conversion.
    - To enable import feature in scripts.

Specifying Capabilities:

- App:

```
{      platformName: "Android",
       platformVersion: "10",
       deviceName: "emulator-5554",
       maxInstances: 1,
       app: "C:/POC/Sample/app/Android-NativeDemoApp-0.2.1.apk",
       appPackage: "com.wdiodemoapp",
       appActivity: "com.wdiodemoapp.MainActivity",
       automationName: "UiAutomator2",
  }
```

- Chrome Browser in App:

```
{
    platformName: "Android",
    platformVersion: "10",
    deviceName: "emulator-5554",
    maxInstances: 1,
    browserName: 'chrome',
    automationName: "UiAutomator2",
    chromedriverExecutable: "./drivers/chromedriver.exe"
  }
```

```
capabilities: [{

    platformName: "Android",
    platformVersion: "10",
    deviceName: "emulator-5554",
    maxInstances: 1,
    //app: "C:/POC/Sample/app/Android-NativeDemoApp-0.2.1.apk",
    appPackage: "com.wdiodemoapp",
    appActivity: "com.wdiodemoapp.MainActivity",
    //browserName: 'chrome',
    automationName: "UiAutomator2",
    //chromedriverExecutable: "./drivers/chromedriver.exe"
}],
//
```

Configuring Reports:

Extra config for Allure reports:

disableWebdriverStepsReporting: true,

disableWebdriverScreenshotsReporting: false,

```
reporters: ['spec','dot',['allure',
                                {
                                    outputDir: 'allure-results',
                                    disableWebdriverStepsReporting: true,
                                    disableWebdriverScreenshotsReporting: false,
                                }

                            ]
],
```

Screenshots Generation:

```
afterTest: function(test, context, { error, result, duration, passed, retries }) {
    if (!passed) {
        browser.takeScreenshot();
    }
},
```

Configure Appium Server:

```
services: ['chromedriver',['appium', {
    command : 'appium'
}]],
```

Babel Configuration: https://webdriver.io/docs/babel.html

- Install babel using:
  npm install --save-dev @babel/core @babel/cli @babel/preset-env @babel/register
- Create a config file `babel.config.js`

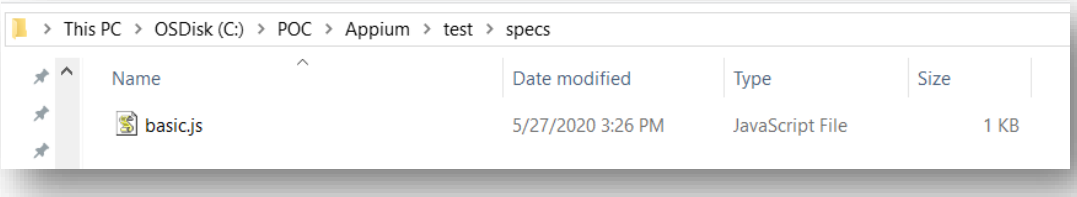
```
module.exports = {
  presets: [
    ['@babel/preset-env', {
      targets: {
        node: 12
      }
    }]
  ]}
```

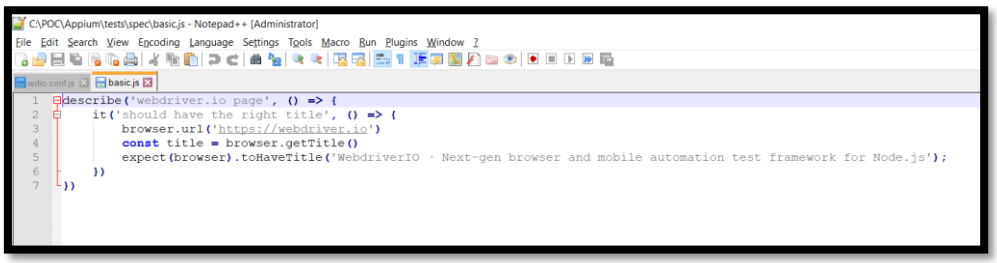
```
//
// Options to be passed to Mocha.
// See the full list at http://mochajs.org/
mochaOpts: {
    ui: 'bdd',
    timeout: 60000,
    require: ['@babel/register']
},
//
```

```
before: function (capabilities, specs) {
    require('@babel/register');
```

Executing WDIO file:

- To run the script use: npx wdio wdio.conf.js
- I have created a basic.js file





Code:

```
describe('webdriver.io page', () => {
  it('should have the right title', () => {
    browser.url('https://webdriver.io')
    const title = browser.getTitle()
    expect(browser).toHaveTitle('WebdriverIO · Next-gen browser and mobile automation test framework for Node.js');
  })
})
```

Execution:

```
C:\POC\Appium>npx wdio wdio.conf.js
```

```
C:\POC\WebDriveIO\webdriverIO_Appium>npm run wdio

> webdriverIO_Appium@1.0.0 wdio C:\POC\WebDriveIO\webdriverIO_Appium
> npx wdio wdio.conf.js


Execution of 1 spec files started at 2020-05-27T10:30:54.811Z

2020-05-27T10:30:54.823Z INFO @wdio/cli:launcher: Run onPrepare hook
2020-05-27T10:30:56.256Z INFO @wdio/cli:launcher: Run onWorkerStart hook
2020-05-27T10:30:56.260Z INFO @wdio/local-runner: Start worker 0-0 with arg: wdio.conf.js
[0-0] 2020-05-27T10:30:56.802Z INFO @wdio/local-runner: Run worker command: run
[0-0] 2020-05-27T10:30:57.069Z INFO webdriverio: Initiate new session using the ./protocol-stub protocol
[0-0] RUNNING in com.wdiodemoapp - C:\POC\WebDriveIO\webdriverIO_Appium\test\specs\WDIO_Login.js
[0-0] 2020-05-27T10:30:58.096Z INFO webdriverio: Initiate new session using the webdriver protocol
[0-0] 2020-05-27T10:30:58.098Z INFO webdriver: [POST] http://localhost:4723/session
[0-0] 2020-05-27T10:30:58.099Z INFO webdriver: DATA {
  capabilities: {
    alwaysMatch: {
      platformName: 'Android',
      platformVersion: '10',
      deviceName: 'emulator-5554',
      appPackage: 'com.wdiodemoapp',
      appActivity: 'com.wdiodemoapp.MainActivity',
      automationName: 'UiAutomator2'
    },
    firstMatch: [ {} ]
  },
  desiredCapabilities: {
    platformName: 'Android',
    platformVersion: '10',
    deviceName: 'emulator-5554',
    appPackage: 'com.wdiodemoapp',
    appActivity: 'com.wdiodemoapp.MainActivity',
    automationName: 'UiAutomator2'
  }
}
```

```
"dot" Reporter:


"spec" Reporter:
------------------------------------------------------------------
emulator-5554 Android 10 #0-0] Spec: C:\POC\WebDriveIO\webdriverIO_Appium\test\specs\WDIO_Login.js
emulator-5554 Android 10 #0-0] Running: emulator-5554 on Android 10
emulator-5554 Android 10 #0-0] Session ID: 572082e4-ddfe-47c8-a9d8-91dd0ddd64a5
emulator-5554 Android 10 #0-0]
emulator-5554 Android 10 #0-0] WDIO APP Login Validations
emulator-5554 Android 10 #0-0]    ✓ Accessing elements from App to Web
emulator-5554 Android 10 #0-0]
emulator-5554 Android 10 #0-0] 1 passing (7.1s)


pec Files:      1 passed, 1 total (100% completed) in 00:00:17
```

To Generate Allure Reports:

- Install allure command line
  - npm install -g allure-commandline --save-dev
- To log steps in report :
  - Import package in script.js file:

```
import allureReporter from '@wdio/allure-reporter';
```

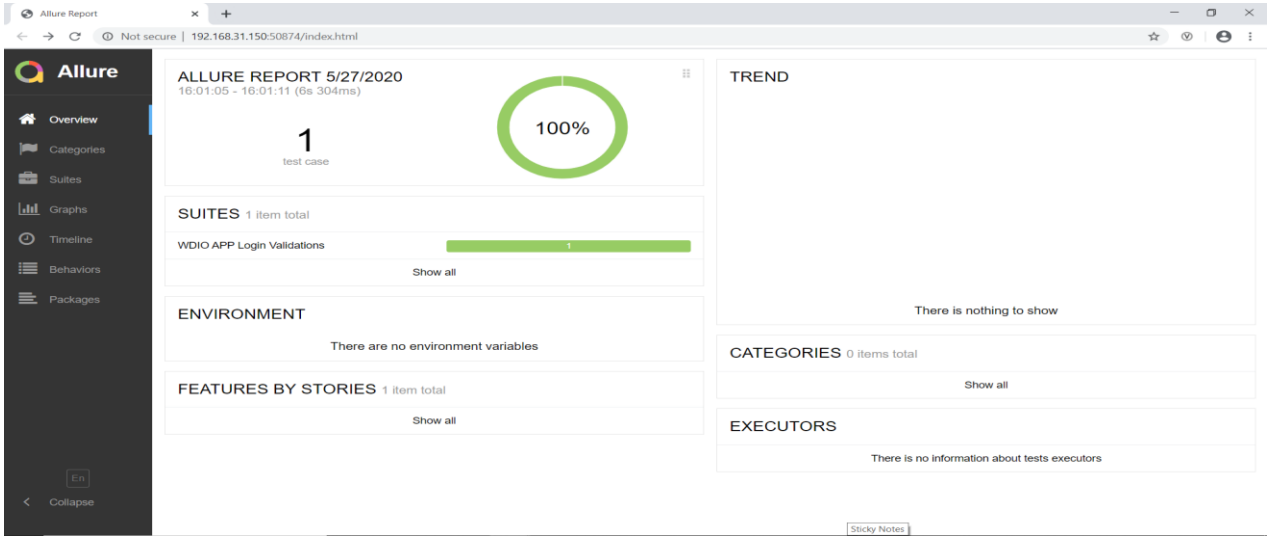
  - To add step:

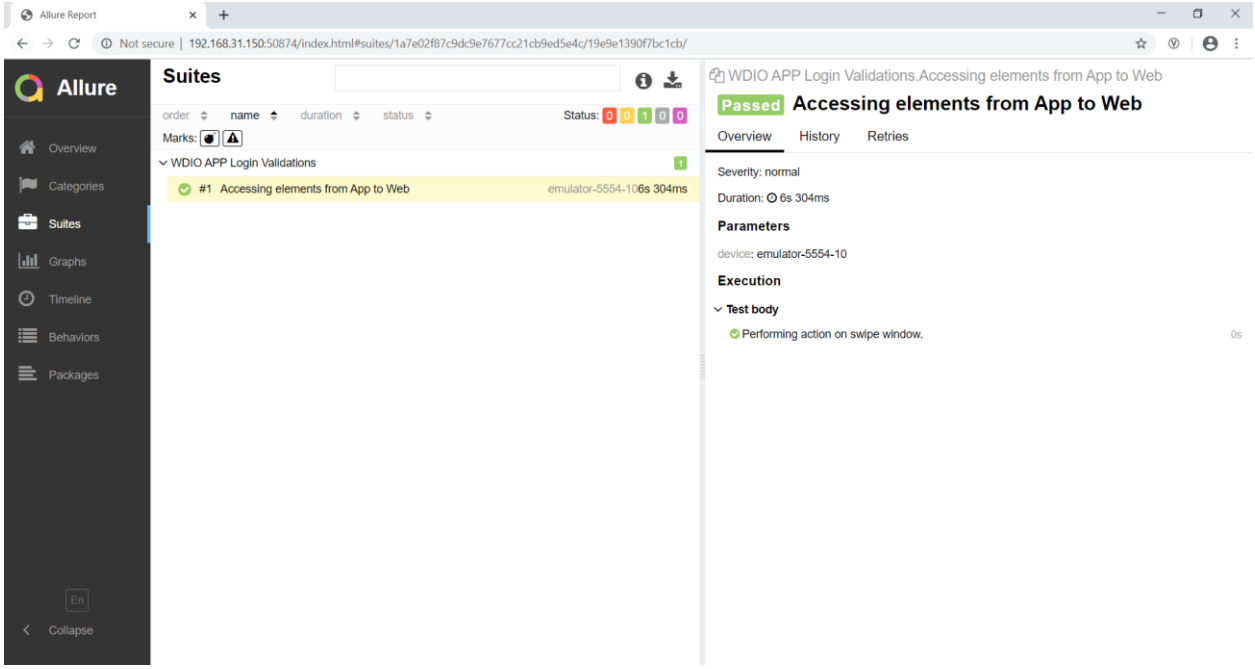
```
allureReporter.addStep("Performing action on swipe window.")
```

- To generate reports, go to the folder and open cmd and type:
- allure generate [allure_output_dir] && allure open

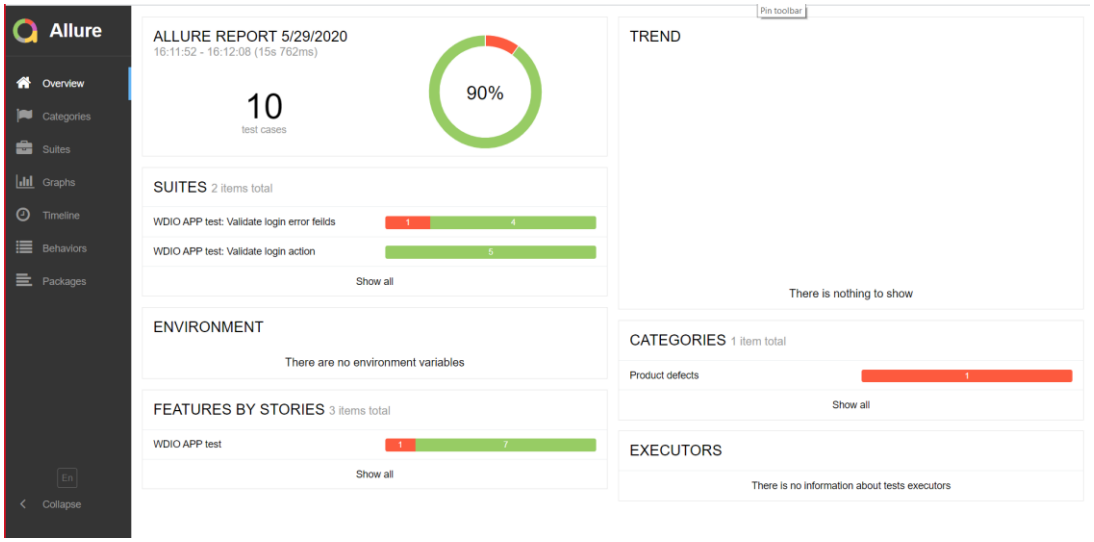
```
C:\POC\WebDriveIO\webdriverIO_Appium>allure generate ./reports/allure-results && allure open
```

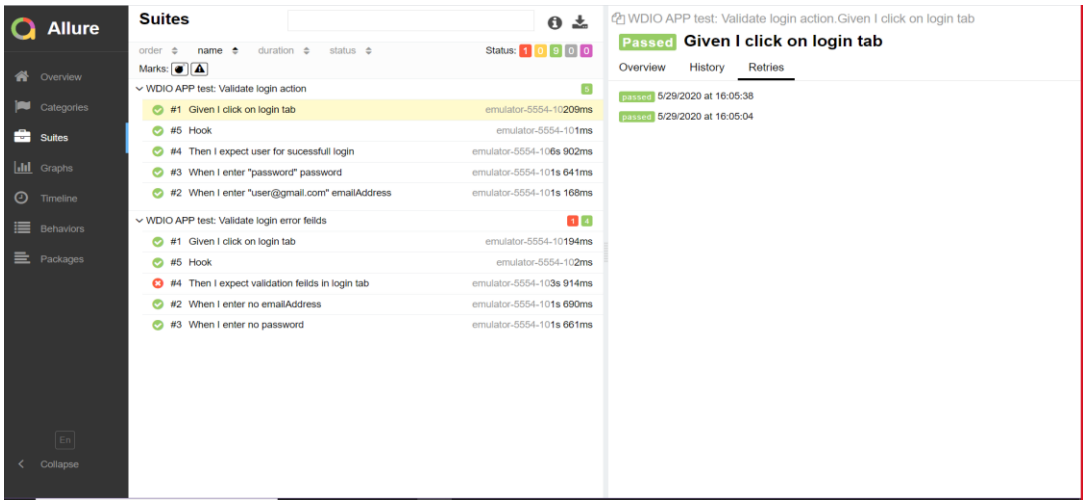
Allure Reports:



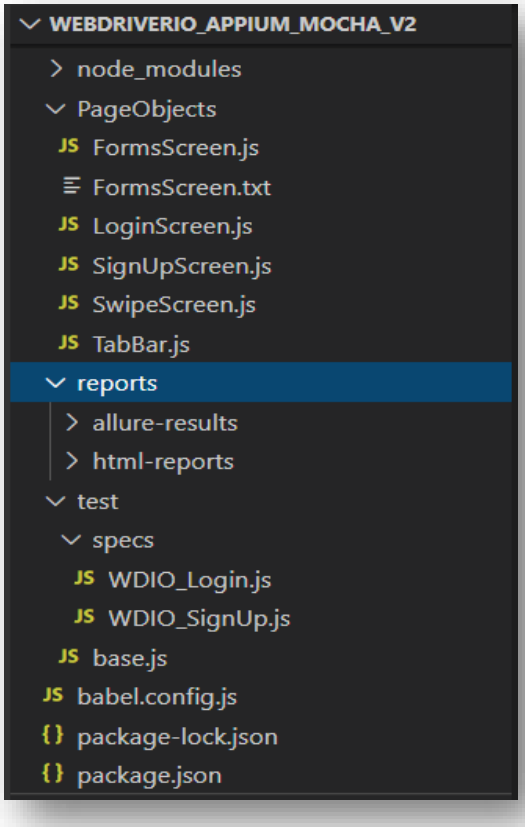


Cucumber Reports:





Project Folder View: VS Code



Testing Specifications:

- **Mobile Device :** Pixel 3, Nexus 5 (Emulators)
- **OS Version:** 9, 10
- **Platform:** Android
- **Browser:** Chrome
- **Demo App:** WebdriverIO Demo App

Framework Structure:

- **Mocha Framework**
- **Chai Assertions**
- **Babel script**

Appium Mocha Framework:

- wdio.conf.js (parameter for framework will be **mocha**)
  - `framework: 'mocha',`
- Retry failed test cases.
- Reporters: spec, dot, allure, html reports
- Assertions: Chai, Mocha
- Automation Framework type: POM Structure.

Folder Structure:

Name	Date modified	Type	Size
app	5/21/2020 5:19 PM	File folder	
drivers	5/18/2020 9:29 PM	File folder	
helper	5/29/2020 3:31 PM	File folder	
node_modules	5/26/2020 4:38 PM	File folder	
PageObjects	5/28/2020 5:40 PM	File folder	
reports	5/29/2020 3:59 PM	File folder	
test	5/20/2020 10:04 PM	File folder	
babel.config.js	5/20/2020 9:55 PM	JavaScript File	1 KB
package.json	5/26/2020 3:23 PM	JSON File	2 KB
package-lock.json	5/26/2020 3:23 PM	JSON File	331 KB
wdio.conf.js	5/28/2020 9:19 PM	JavaScript File	13 KB

Specs Location:

This PC > OSDisk (C:) > POC > WebDriveIO > webdriverIO_Appium > test > specs

Name	Date modified	Type	Size
WDIO_Login.js	5/29/2020 3:59 PM	JavaScript File	4 KB
WDIO_SignUp.js	5/29/2020 3:29 PM	JavaScript File	4 KB

Sample Mocha Wido File (save as .json):

wdio.conf.txt

Sample Appium Package.json file (save as .json):

package.txt

Appium Cucumber Framework:

- wdio.conf.js (parameter for framework will be **cucumber**)
  - `framework: 'cucumber',`
- Retry failed test cases.
- Reporters: spec, allure
- Assertions: Chai, Mocha
- Automation Framework type: POM Structure.

Folder Structure:

Name	Date modified	Type	Size
allure-report	5/29/2020 4:12 PM	File folder	
app	5/22/2020 3:40 AM	File folder	
drivers	5/22/2020 3:40 AM	File folder	
node_modules	5/29/2020 11:33 AM	File folder	
PageObjects	5/29/2020 1:10 PM	File folder	
reports	5/29/2020 4:11 PM	File folder	
src	5/22/2020 4:06 AM	File folder	
babel.config.js	5/22/2020 3:37 AM	JavaScript File	1 KB
package.json	5/29/2020 1:38 PM	JSON File	2 KB
package-lock.json	5/29/2020 11:33 AM	JSON File	255 KB
wdio.conf.js	5/29/2020 4:13 PM	JavaScript File	14 KB

Step-Definitions and Feature File Location:

This PC > OSDisk (C:) > POC > WebDriveIO > webriver_cucumber_appium > src

Name	Date modified	Type	Size
features	5/22/2020 3:32 AM	File folder	
step-definitions	5/22/2020 3:53 AM	File folder	
support	5/22/2020 4:50 AM	File folder	

Sample wdio.conf.js file:

wdio.conf.txt

Sample package.json file:

package.txt

Pros and Cons:

Pros:

- **Synchronous Execution is available as inbuilt feature, which overcomes Asynchronous functionality of Javascript.**
- **Parallel Execution of Scripts on multiple devices and browsers is possible.**
- **It has support for integrating test suites with multiple reporting features.**
- **Framework level configurations for either Mocha, jasmine or Cucumber is made easy.**
- **As it works on NPM, installation of packages required for WDIO is easy to setup using package.json file.**
- **Chromedriver and Geckodriver services are available for auto downloading and handling the selenium server for browser executions without any external creation for selenium server.**
- **User Actions performed in mobile can be easily achieved.**
- **Syntax for locating elements by using xpath, css selector, id etc., has different ways introduced along with existing approach, which made it easy compared to ideal automation approach.**
- **Switching the context between APP to WEBVIEW to Chrome Browser is easily handled.**
- **Can be easily configured with Jenkins for auto triggering the build and can also configure Allure reports in Jenkins Tool for Report generation.**
- **Speed of Execution is good.**

Cons:

- **Has limited source for resolving issues and implementations.**
- **Need to have prior advance knowledge in mobile automation for performing better actions on mobile devices as there is very less information available in docs.**
- **Few actions when upgraded from older version is properly not handled or documented in Latest releases.**

Sample Test Application:

- Demo WebdriverIO mobile APP

Test Scenarios Covered:

S.No	Scenarios(Webdriver.io Demo App)
1	Signup-> Providing username and password Fields. Login Validation for Credentials used for Signup.
2	Login-->WebView --> Validating WebView Contents for WebdriverIO Url.
3	Login--> Forms --> Entering Form details(Handlin Model window, selection from dropdown, Text validation, Button validation, Tick box) and Validating same.
4	Login --> Swipe --> Validating Horizontal Swipe and Each page swipe using dots at bottom of screen.
5	Login --> Swipe --> Clicking one of the Page and Validating link opened in web UI or not.

Comparison with other tools:

Currently Webdriverio stands at 4th position w.r.t other tools namely: Nightwatch.js, Leadfoot ->Intern, Pupeteer. Other JS tools has their own advantages in reporting, scripting, integrations, but few are the comparisons where webdriverio is good.

Advantages over Other tools:

Specs	WebDriver.io	Nightwatch.js	Cypress	Pupeteer
Framework Configurations	Can be configured with Jasmine, Mocha, Cucumber	Has its own framework	Has its own intern framework	Has its own framework
Visual Regression Test	Possible	Not Possible	Possible	Possible
Cucumber BDD Support	Possible	Not Possible	Possible	Not Possible
Size of Complexity of Applications	Low	Low	High	Low
Cross Browser Testing	Possible	Possible	Not Possible only Single Tab	Not Possible