*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](#_Toc42096015)

[Introduction**:** 4](#_Toc42096016)

[Main Features of WebdriverIO: 4](#_Toc42096017)

[Objective: 4](#_Toc42096018)

[Software Tools Required: 4](#_Toc42096019)

[Issues: 5](#_Toc42096020)

[Installation Procedure: 5](#_Toc42096021)

[Creating Webdriver IO Appium project: 6](#_Toc42096022)

[Configuring wdio.config.js file: 11](#_Toc42096023)

[Specifying Capabilities: 11](#_Toc42096024)

[Configuring Reports: 12](#_Toc42096025)

[Screenshots Generation: 12](#_Toc42096026)

[Configure Appium Server: 12](#_Toc42096027)

[Babel Configuration 13](#_Toc42096028)

[Executing WDIO file: 13](#_Toc42096029)

[Code: 14](#_Toc42096030)

[Execution: 14](#_Toc42096031)

[To Generate Allure Reports: 15](#_Toc42096032)

[Allure Reports 15](#_Toc42096033)

[Cucumber Reports 16](#_Toc42096034)

[Project Folder View 17](#_Toc42096035)

[Testing Specifications: 17](#_Toc42096036)

[Framework Structure: 17](#_Toc42096037)

[Appium Mocha Framework: 17](#_Toc42096038)

[Appium Cucumber Framework: 19](#_Toc42096039)

[Pros and Cons: 20](#_Toc42096040)

[Pros: 20](#_Toc42096041)

[Cons: 20](#_Toc42096042)

[Sample Test Application 21](#_Toc42096043)

[Test Scenarios Covered: 21](#_Toc42096044)

[Comparison with other tools: 22](#_Toc42096045)

[Advantages over Other tools: 22](#_Toc42096046)

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

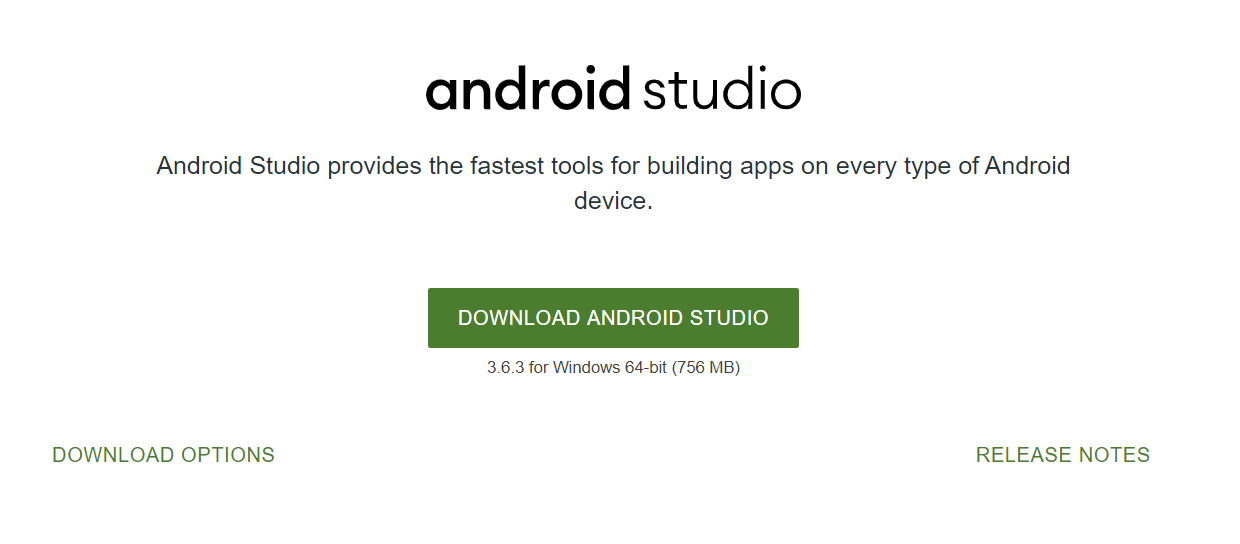
# Installation Procedure:

* **Appium**:

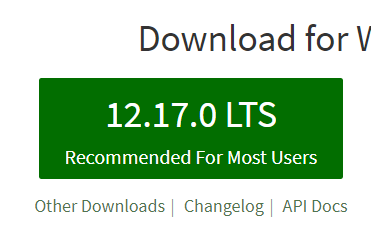
Install Appium Globally:



* **Android Studio with SDK tools:**



* **Node JS: require version 12**



**WebdriverIO : Version 6**

**For reference:** <https://webdriver.io/docs/gettingstarted.html>

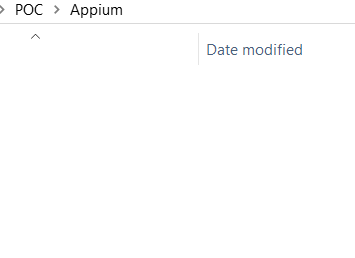
To install: npm i --save-dev @wdio/cli

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:

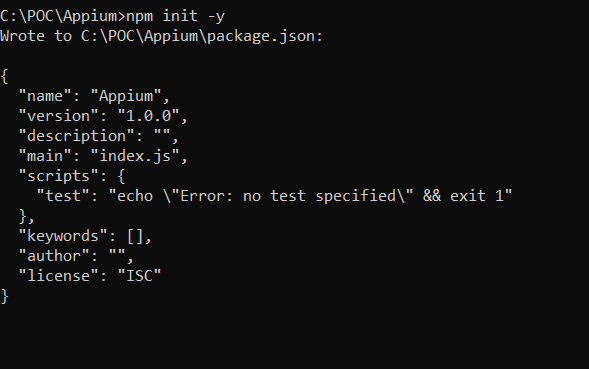


* Open cmd here:

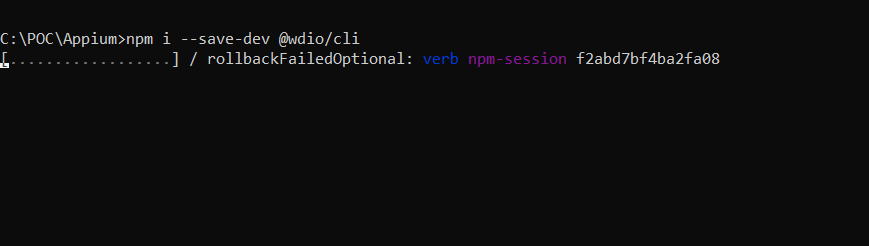


* Now run command: npm init –y

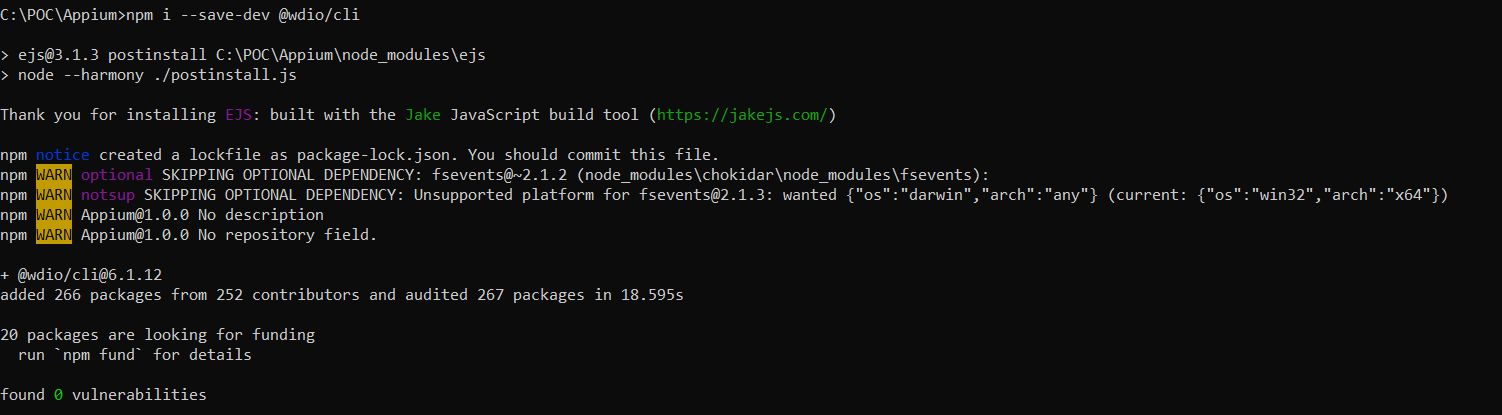
It should generate below output



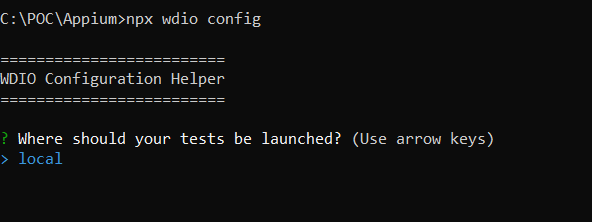
* Once the package.json file is created, install wdio:
  + npm i --save-dev @wdio/cli



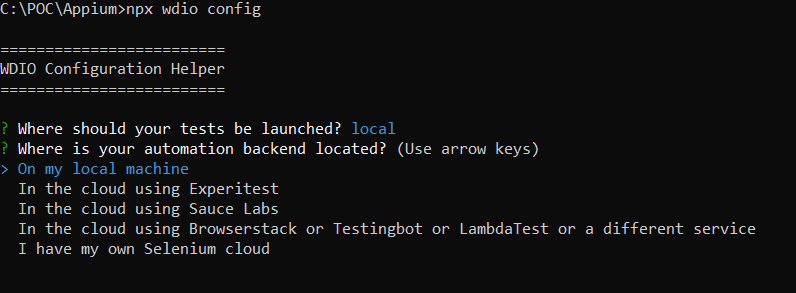
Finally, you should see this:



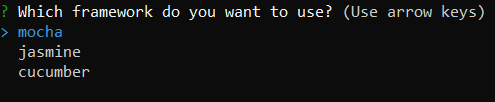
* Now we need to create wdio file with appium settings run: npx wdio config
  + Select local.



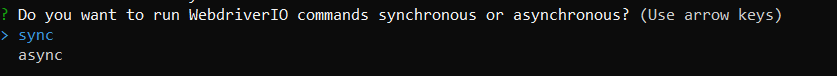
* + Select on my local machine



* + Select type of framework : mocha



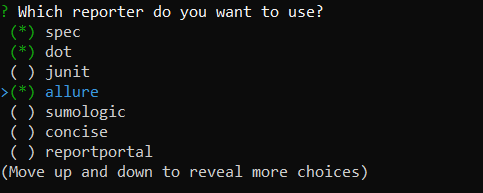
* + Select execution type: sync



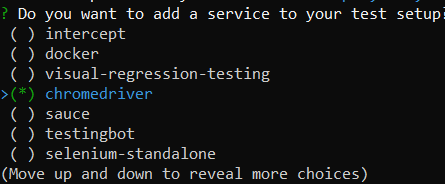
* + Specify the location where your test scripts are located, if you want to use default location press enter.

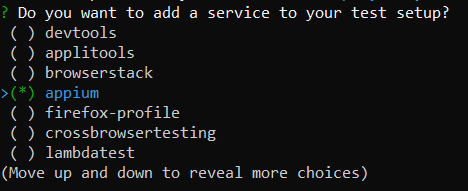


* + Select the type of reports used: spec, dot, allure



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

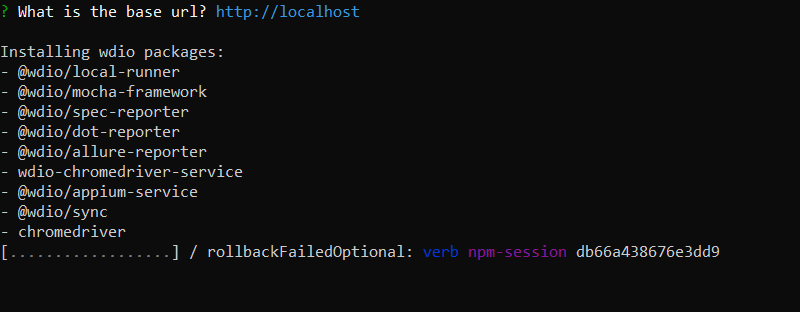


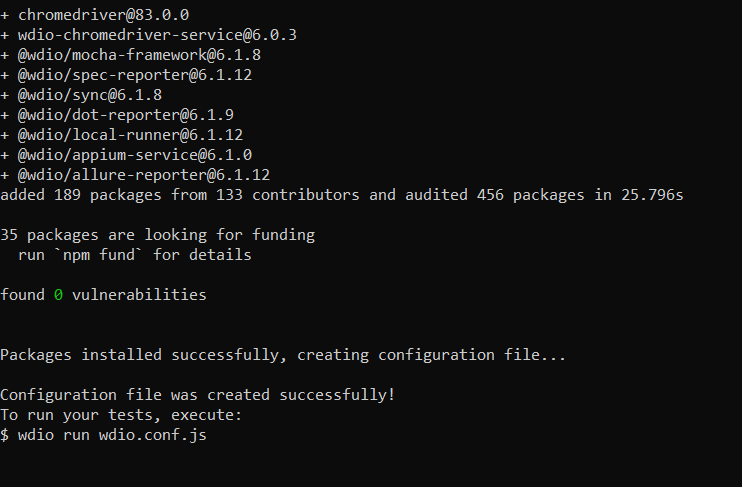


* + Specify the URL of the website or press enter:

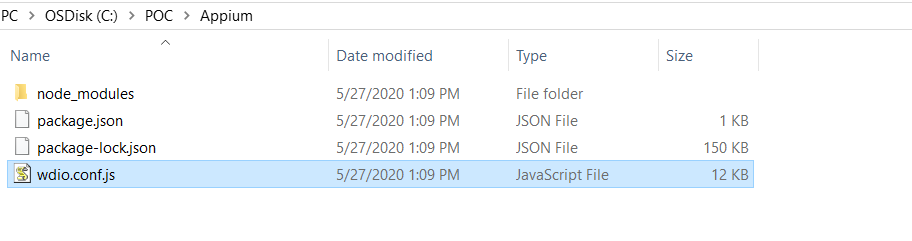


* + Then the installation process will start which will download all the specified modules for the project.





* 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"

}

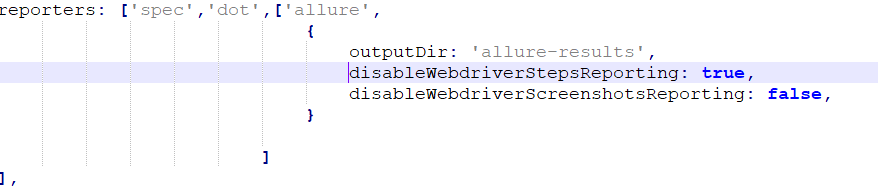


## Configuring Reports:

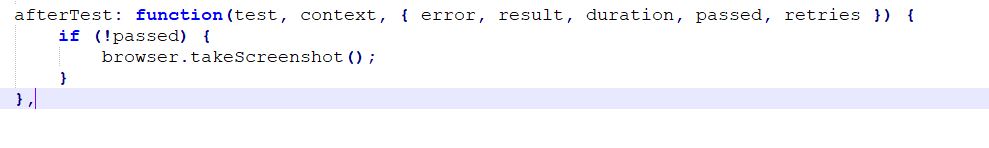
Extra config for Allure reports:

disableWebdriverStepsReporting: true,

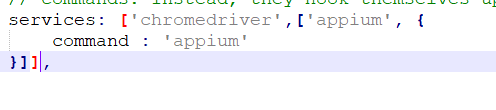
disableWebdriverScreenshotsReporting: false,



## Screenshots Generation:



## Configure Appium Server:



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](https://babeljs.io/docs/en/config-files)

module.exports = {

presets: [

['@babel/preset-env', {

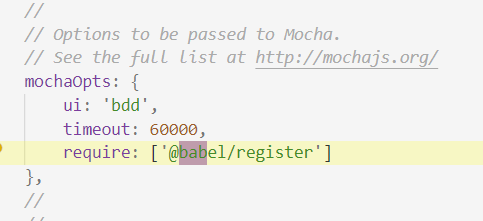
targets: {

node: 12

}

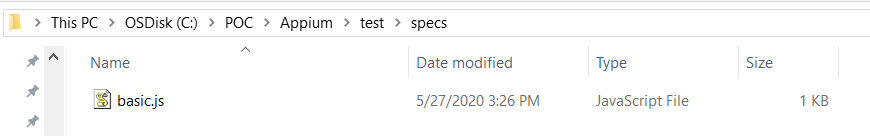
}]

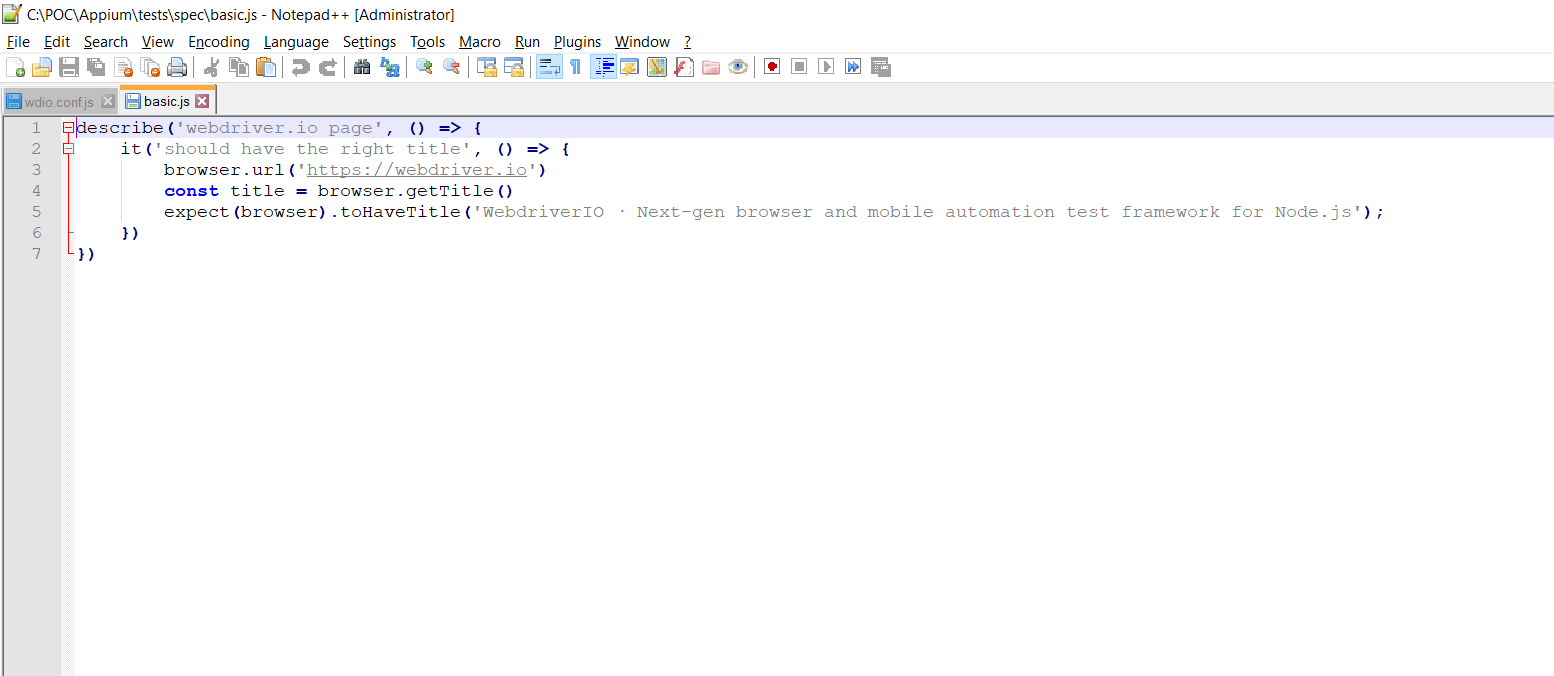
]}





# 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')

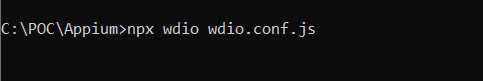
const title = browser.getTitle()

expect(browser).toHaveTitle('WebdriverIO · Next-gen browser and mobile automation test framework for Node.js');

})

})

## Execution:







# 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:



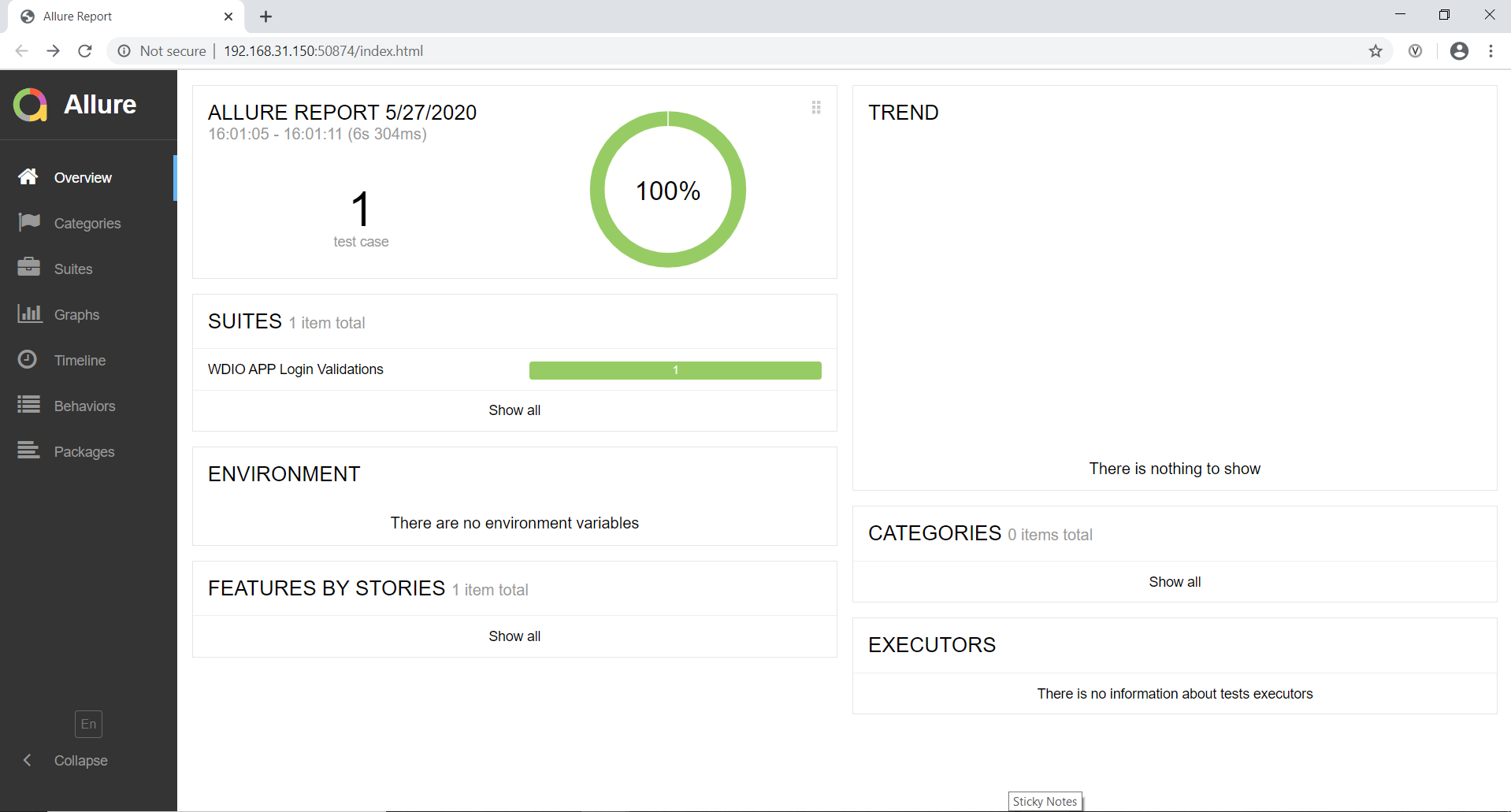
* + To add step:

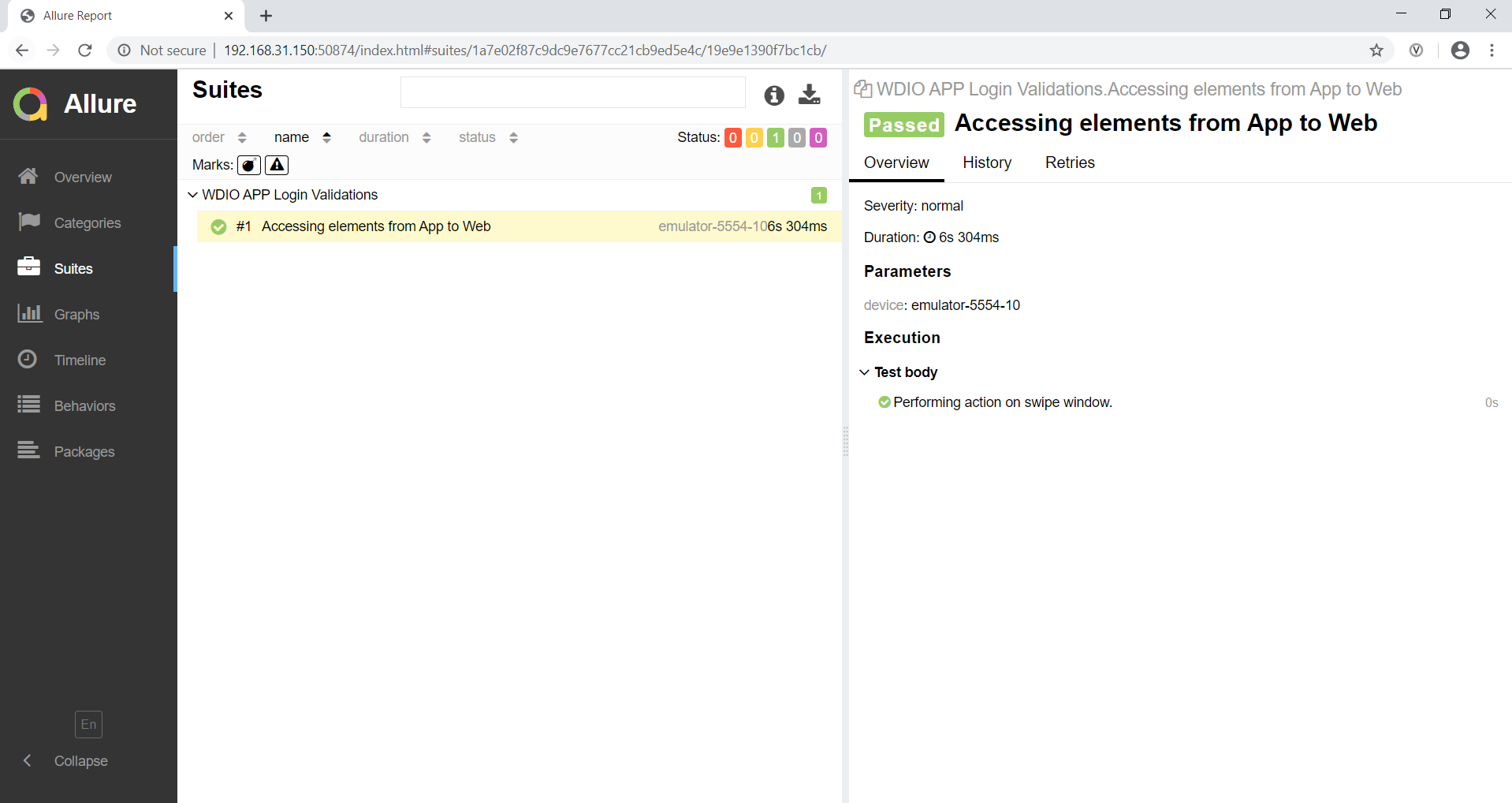


* To generate reports, go to the folder and open cmd and type:
* allure generate [allure\_output\_dir] && allure open

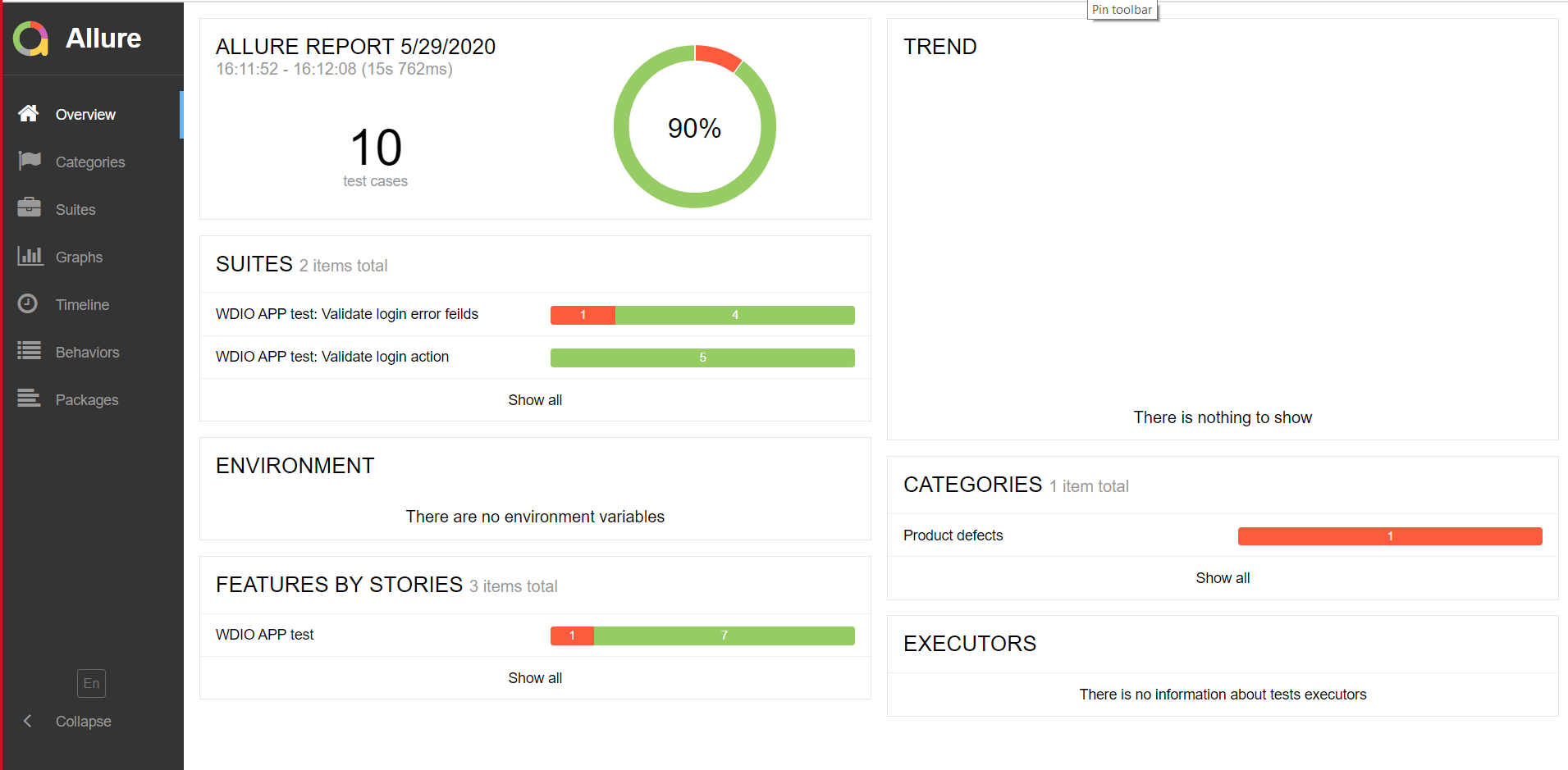


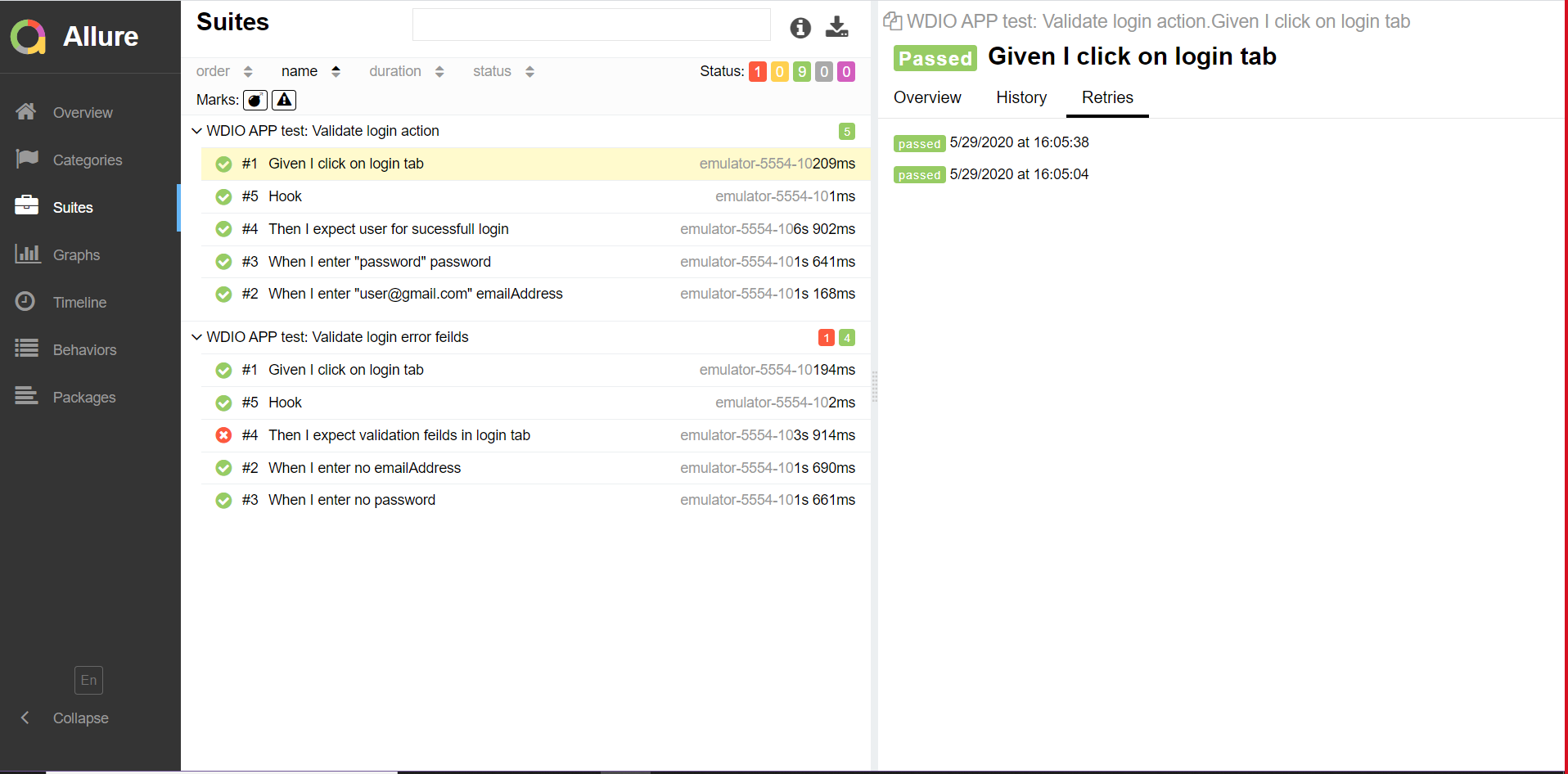
Allure Reports:



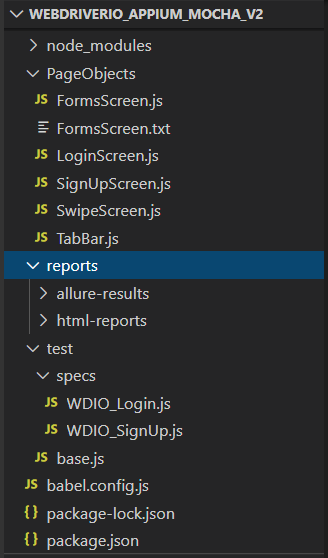


Cucumber Reports:





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

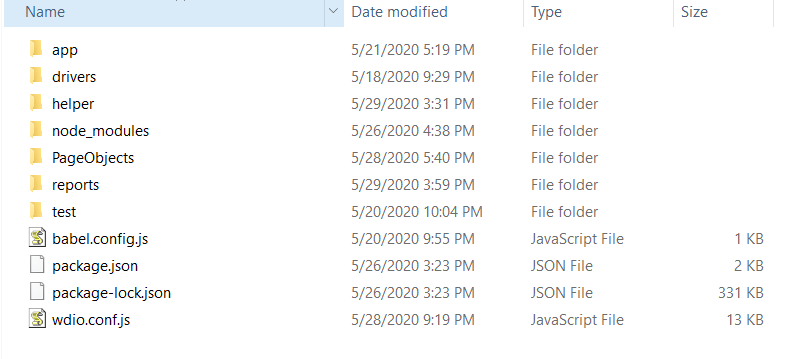
# Framework Structure:

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

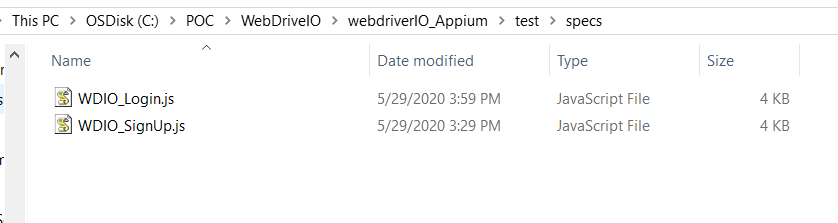
## Appium Mocha Framework:

* 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:**



**Specs Location:**



**Sample Mocha Wido File (save as .json):**

****

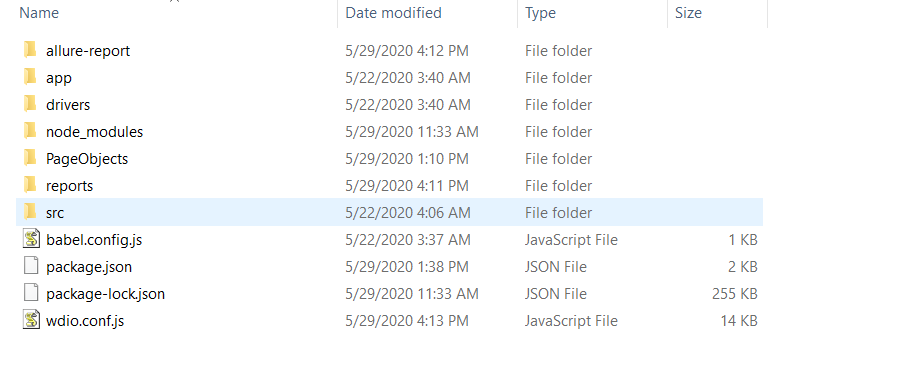
**Sample Appium Package.json file (save as .json):**

****

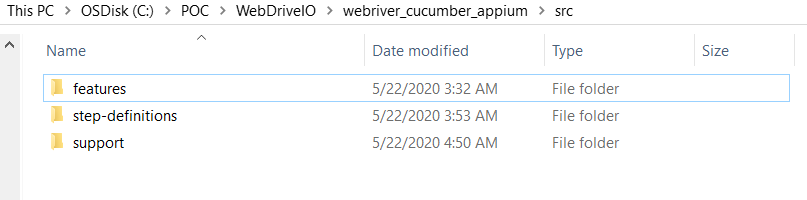
## Appium Cucumber Framework:

* 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:**



**Step-Definitions and Feature File Location:**



**Sample wdio.conf.js file:**

****

**Sample package.json file:**

****

# 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 |