Testing using Cypress and Cucumber in JavaScript

- Overview
- Requirements
- Description
 - Using Jira and Xray as master
 - Using Git or other VCS as master
- References

Overview

In this tutorial, we will create UI tests as Cucumber Scenario(s)/Scenario Outline(s) and use Cypress to implement the tests in JavaScript.

Source-code for this tutorial

Code is available in GiHub; the repo contains some auxiliary scripts.

Requirements

- nodejs
- npm packages
 - cypress
 - cypress-cucumber-preprocessor
 - cucumber-json-merge

Description

For the purpose of this tutorial, we will use a dummy website (source-code here) containing just a few pages to support login/logout kind of features; we aim to test precisely those features.

	D
Login	Page
Please input y button.	our user name and password and click the login
User Name: [
Password:	
[LOGIN

We need to configure Cypress to use the cypress-cucumber-preprocessor, which provides the ability to understand .feature files and also to produce Cucumber JSON reports.

cypress/plugins/index.js

/cypress.json

```
const cucumber = require('cypress-cucumber-preprocessor').default
/**
 * @type {Cypress.PluginConfig}
 */
module.exports = (on, config) => {
    // `on` is used to hook into various events Cypress emits
    // `config` is the resolved Cypress config
    on('file:preprocessor', cucumber())
}
```

In Cypress' main configuration file, define the base URL of the website under test, the regex of the files that contain the test scenarios (i.e. <...>.feature files). Other options may be defined e.g for bypassing chromeWebSecurity, additional reporters, the ability to upload results to Cypress infrastructure in the cloud, etc).

{ "baseUrl": "https://robotwebdemo.herokuapp.com/", "testFiles": "**/*.feature", "ignoreTestFiles": ["*.js", "*.md"], "reporter": "junit", "reporterOptions": { "mochaFile": "test-results/test-output-[hash].xml" }, "chromeWebSecurity": false, "projectId": "bfi83g" }

Next, here is an example of the contents of package.json.

package.json

```
{
  "name": "cypress-cucumber-robotdemo",
  "version": "1.0.0",
  "description": "An example for Cypress and Cucumber usage using Robot login demo website",
 "main": "index.js",
  "scripts": {
   "cypress:open:local": "CYPRESS_ENV=localhost npm run cypress:open",
    "cypress:open:prod": "CYPRESS_ENV=production npm run cypress:open",
    "cypress:open": "cypress open",
    "test:local": "CYPRESS_ENV=localhost npm run test --spec 'cypress/integration/**/*.feature",
   "test:prod": "CYPRESS_ENV=production npm run test",
   "test": "cypress run --spec 'features/**/*.feature' --config integrationFolder=.",
   "test:debug:local": "CYPRESS_ENV=localhost npm run test:debug",
    "test:debug:prod": "CYPRESS_ENV=production npm run test:debug",
    "test:debug": "cypress run --headed --browser chrome --env TAGS='@e2e-test' --spec 'cypress/integration/**
/*.feature'",
   "test:pull-features": "git submodule update --remote gherkin-features && cp -rf gherkin-features/* cypress
/integration && node ./scripts/remove-old-features.js",
   "attach_screenshots": "node attach_screenshots.js"
 },
  "author": "",
  "license": "Private",
  "dependencies": {
   "axios": "^0.18.0",
   "cucumber-json-merge": "0.0.4",
   "fs-extra": "^7.0.1",
   "qlob": "^7.1.3"
 },
  "devDependencies": {
   "cypress": "^5.5.0",
   "cypress-cucumber-preprocessor": "^4.0.0",
   "eslint": "^5.13.0",
   "eslint-config-airbnb-base": "^12.1.0",
   "eslint-config-prettier": "^2.9.0",
    "eslint-plugin-import": "^2.11.0",
    "eslint-plugin-prettier": "^2.6.0",
   "husky": "^1.3.1",
   "lint-staged": "^8.1.3"
 },
  "cypress-cucumber-preprocessor": {
   "nonGlobalStepDefinitions": true,
    "cucumberJson": {
     "generate": true,
     "outputFolder": "cypress/cucumber-json",
     "filePrefix": "",
     "fileSuffix": ".cucumber"
   }
 },
  "husky": {
   "hooks": {
     "pre-commit": "lint-staged"
   }
 },
  "lint-staged": {
   "*.js": [
     "eslint".
     "git add"
   ]
 }
}
```

Before moving into the actual implementation, we need to decide which workflow we'll use: do we want to use Xray/Jira as the master for writing the declarative specification (i.e. the Gherkin based Scenarios), or do we want to manage those outside using some editor and store them in Git, for example?

Learn more

Please see Testing in BDD with Gherkin based frameworks (e.g. Cucumber) for an overview of the possible workflows.

The place that you'll use to edit the Cucumber Scenarios will affect your workflow. There are teams that prefer to edit Cucumber Scenarios in Jira using Xray, while others prefers to edit them by writing the .feature files by hand using some IDE.

Using Jira and Xray as master

This section assumes you will use Xray as master, i.e. the place that you'll be using to edit the specifications (e.g. the scenarios that are part of .feature files).

The overall flow would be something like this:

- 1. create Scenario/Scenario Outline as a Test in Jira; usually, it would be linked to an existing "requirement"/Story (i.e. created from the respective issue screen)
- 2. implement the code related to Gherkin statements/steps and store it in Git, for example
- 3. generate .feature files based on the specification made in Jira
- 4. checkout the code from Git
- 5. run the tests in the CI
- 6. import the results back to Jira

Usually, you would start by having a Story, or similar (e.g. "requirement"), to describe the behavior of a certain feature and use that to drive your testing.

If you have it, then you can just use the "Create Test" on that issue to create the Scenario/Scenario Outline and have it automatically linked back to the Story/"requirement."

Otherwise, you can create the Test using the standard (issue) Create action from Jira's top menu.

Calculator / CAl	LC-7905 I can login the a	application						
Edit Q Commen	t Assign More 🗸	Start Progress	Close Issue A	dmin 🖌				
✓ Details								
Type:	 Story 			Status:	OPE	N (View Workflo	(wc	
Priority:	🔶 Major			Resolution:	Unre	solved		
Affects Version/s:	None			Fix Version/s	S: Non	e		
Component/s:	None							
Labels:	None							
Requirement Status:	UNCOVERED							
 Description 								
As a user, I can login t	he application							
✓ Test Coverage								
						Create Test	Create Sub-Test Execution	+ Link ~
No Tests were found to	esting the requirement.							
Calculator / CA	LC-7906 I can logout th	e application						
🖋 Edit 🛛 Q Commer	nt Assign More	Start Progress	Close Issue	Admin 🖌				
✓ Details								
Type:	 Story 		Statu	s:	OPEN (View	Workflow)		
Priority:	🔶 Major		Resol	ution:	Unresolved			
Affects Version/s:	None		Fix Ve	ersion/s:	None			
Component/s:	None							
Labels:	None							
Requirement Status:	UNCOVERED							
 Description 								
As a user, I can logout	t the application							
 Test Coverage 								
				Creat	e Test Cre	ate Sub-Te <u>st E</u>	ixecution + Link -	
No Tests were found t	esting the requirement							

In this case, we'll create a Cucumber Test, of Cucumber Type "Scenario."

We can fill out the Gherkin statements immediately on the Jira issue "create dialog" or we can create the Test issue first and fill out the details on the next screen, from within the Test issue. In the latter case, we can take advantage of the built-in Gherkin editor which provides auto-complete for Gherkin steps.



After the Test is created it will impact the coverage of related "requirement," if any.

The coverage and the test results can be tracked in the "requirement" side (e.g. user story). In this case, you may see that coverage changed from being UNCOVERED to NOTRUN (i.e. covered and with at least one test not run).

Ca	alculator / CA	LC-7905 I can log	in the applic	ation					
🖋 Edit	Q Commer	t Assign	More 🖌 Star	t Progress	Close Issue	Admin 🗸			
✓ Details									
Type:		Story			State	ıs:	OPEN (View)	Workflow)	
Priority:		斧 Major			Resc	lution:	Unresolved		
Affects	Version/s:	None			Fix V	ersion/s:	None		
Compor	nent/s:	None							
Labels:		None							
Require	ment Status:	NC	DTRUN		-				
As a use	er, I can login t verage	he application	1			Crea	ite Test Crea	ate Sub-Test Execut	tion + Link ×
TEST C	OVERAGE FOR TH	HE FOLLOWING A	NALYSIS SCOPE						
Sco	pe: Version; V	ersion: None	- latest execution;	Environme	nt: All Environn	nents 👻			
Ę	F Filter(s)								
·								Show 10 🗸 entries	s Columns -
4	P 🔶 Sta	tus 🗍	Resolution	🔺 Key		Summary	Test Run	s 🔶 Test St	tatus
	↑ OPEI	N (Unresolved	CALC-7	901	Valid Login	≣0		ODO
Showing	1 to 1 of 1 entri	ies						First Previo	us 1 Next Last

Additional tests could be created, eventually linked to the same Story or linked to another one (e.g. logout).

The related statement's code is managed outside of Jira and stored in Git, for example.

In Cypress, the test code is stored under cypress/integration directory, which itself contains several other directories. In this case, we've organized them as follows:

• cypress/integration/common: step implementation files, in JavaScript.

cypress/integration/common/login.js

```
import { Given, When } from 'cypress-cucumber-preprocessor/steps';
import LoginPage from '../../pages/login-page';
import LoginResultsPage from '../../pages/login-results-page';
Given(/^browser is opened to login page$/, () => {
 LoginPage.visit();
});
When('user {string} logs in with password {string}', (username, password) => {
 LoginPage.enter_username(username);
 LoginPage.enter_password(password);
 LoginPage.pressLogin();
});
Then(/^welcome page should be open$/, () => {
    LoginResultsPage.expect().toBeSuccessful();
  });
Then(/^error page should be open$/, () => {
   LoginResultsPage.expect().toBeUnsuccessful();
  });
```

cypress/integration/common/logout.js

```
import { Given, When } from 'cypress-cucumber-preprocessor/steps';
import LoginPage from '../../pages/login-page';
import LoginResultsPage from '../../pages/login-results-page';
Given(/^browser is opened to login page$/, () => {
 LoginPage.visit();
});
When('user {string} logs in with password {string}', (username, password) => {
 LoginPage.enter_username(username);
 LoginPage.enter_password(password);
 LoginPage.pressLogin();
});
Then(/^welcome page should be open$/, () => {
   LoginResultsPage.expect().toBeSuccessful();
  });
Then(/^error page should be open$/, () => {
   LoginResultsPage.expect().toBeUnsuccessful();
  });
```

cypress/integration/pages: abstraction of different pages, somehow based on the page-objects model

```
    cypress/integration/pages/login.js
```

```
import LoginResultsPage from './login-results-page';
const USERNAME_FIELD = 'input[id=username_field]';
const PASSWORD_FIELD = 'input[id=password_field]';
const LOGIN_BUTTON = 'input[type=submit]';
const LOGIN_TEXT = 'LOGIN';
class LoginPage {
 static visit() {
   cy.visit('/');
  }
 static enter_username(username) {
   cy.get(USERNAME_FIELD)
     .type(username);
  }
  static enter_password(password) {
   cy.get(PASSWORD_FIELD)
     .type(password);
  }
 static pressLogin() {
   cy.get(LOGIN_BUTTON).contains(LOGIN_TEXT)
      .click();
   return new LoginResultsPage();
  }
}
export default LoginPage;
```

```
    cypress/integration/pages/logout-results-page.js
```

```
const RESULT_HEADER = 'h1';
class LogoutResultsPage {
    static expect() {
       return {
            toBeSuccessful: () => {
               cy.get(RESULT_HEADER).should('have.text', 'Login Page')
            },
        };
    }
}
export default LogoutResultsPage;
```

cypress/integration/pages/welcome-page.js

```
import LoginPage from './login-page';
const LOGOUT_LINK = 'a';
const LOGOUT_TEXT = 'logout';
class WelcomePage {
  static visit() {
    cy.visit('/welcome.html');
  }
  static pressLogout() {
    cy.get(LOGOUT_LINK).contains(LOGOUT_TEXT)
    .click();
    return new LoginPage();
  }
}
export default WelcomePage;
```

You can then export the specification of the test to a Cucumber .feature file via the REST API, or the **Export to Cucumber** UI action from within the Test /Test Execution issue or even based on an existing saved filter. A plugin for your CI tool of choice can be used to ease this task.

So, you can either:

use the UI

Í	Cal	Iculator / CAI	_C-7901]			
	🖋 Edit	Q Commen	t Assign	More 🗸	Start Progress	Resolve Issue
~	Details			Log work		
	Type: Priority: Affects V Compone	/ersion/s: ent/s:	 Test Medium None None 	Agile Boar Rank to To Rank to Bo	d pp ottom	Status: Resolut Fix Vers
	Labels:		cypress/inte	inte Attach files		
~	Descript Click to a	ion add descriptio	n	Voters Stop watch Watchers	hing	
~	Test Det	ails		Create sub	o-task	
	Type:		Cucumber	Convert to	sub-task	
	Scenario	Туре:	Scenario	Move		
	Scenario	:	Given brows When user " Then welcom	Link Clone	5	page sword "mode"
				Delete		
>	Pre-Con	ditions		Trigger Je Trigger Je	nkins job nkins job an	
*	Test Sets This test	s is not associa	ited with Test	Reset Test	RunStatus	
*	Test Pla r This test	1s is not associa	ited with Test	Export to 0 Export Tes Export Tes	Cucumber st to XML st Runs to CSV	
C						

• use the REST API (more info here)

```
o #!/bin/bash
rm -f features/*.feature
curl -u admin:admin "http://jiraserver.example.com/rest/raven/1.0/export/test?keys=CALC-7905;CALC-
7906&fz=true" -o features.zip
unzip -o features.zip -d features
```

• use one of the available CI/CD plugins (e.g. see an example of Integration with Jenkins)

We will export the features to a new directory named features/ on the root folder of your Cypress project (we'll need to tell Cypress to use this folder).

After being exported, the created .feature(s) will contain references to the Test issue key, eventually prefixed (e.g. "TEST_") depending on an Xray global setting, and the covered "requirement" issue key, if that's the case. The naming of these files is detailed in Export Cucumber Features.

features/1_CALC-7905.feature

```
@REO CALC-7905
Feature: As a user, I can login the application
       #As a user, I can login the application
       @TEST_CALC-7903
       Scenario Outline: Login With Invalid Credentials Should Fail
               Given browser is opened to login page
               When user "<username>" logs in with password "<password>"
               Then error page should be open
                        Examples:
                               | username | password |
                                | invalid | mode
                                | demo | invalid |
| invalid | invalid |
                                demo mode
                                                      @TEST_CALC-7902
       Scenario: Invalid Login
               Given browser is opened to login page
                When user "dummy" logs in with password "password"
               Then error page should be open
       @TEST CALC-7901
       Scenario: Valid Login
               Given browser is opened to login page
               When user "demo" logs in with password "mode"
               Then welcome page should be open
```

features/1_CALC-7906.feature

@REQ_CALC-7906
Feature: As a user, I can logout the application
#As a user, I can logout the application

@TEST_CALC-7904
Scenario: Valid Logout
Given user is on the welcome page
When user chooses to logout
Then login page should be open

To run the tests and produce Cucumber JSON reports(s), we can either use npm or cypress command directly.

npm run test

```
# or instead...
```

node_modules/cypress/bin/cypress run --spec 'features/**/*.feature' --config integrationFolder=.

This will produce one Cucumber JSON report in cypress/cucumber-json directory per each .feature file.

The cypress-cucumber-preprocessor package, as of v4.0.0, does not produce reports containing the screenshots embedded.

However, the following script (credits to the user that provided it on GitHub) can be used to update the previous JSON reports so that they contain the screenshots of the failed tests.

attach_screenshots.js

```
const fs = require('fs-extra')
const path = require('path')
const chalk = require('chalk')
const cucumberJsonDir = './cypress/cucumber-json'
const cucumberReportFileMap = {}
const cucumberReportMap = {}
const jsonIndentLevel = 2
const ReportDir = './cypress/reports/cucumber-report'
const screenshotsDir = './cypress/screenshots'
getCucumberReportMaps()
addScreenshots()
//Mapping cucumber json files from the cucumber-json directory to the features
function getCucumberReportMaps() {
   const files = fs.readdirSync(cucumberJsonDir).filter(file => {
       return file.indexOf('.json') > -1
   })
   files.forEach(file => {
       const json = JSON.parse(
           fs.readFileSync(path.join(cucumberJsonDir, file))
       )
       if (!json[0]) { return }
       const [feature] = json[0].uri.split('/').reverse()
       cucumberReportFileMap[feature] = file
       cucumberReportMap[feature] = json
   })
}
//Adding screenshots to the respective failed test steps in the feature files
function addScreenshots() {
   const prependPathSegment = pathSegment => location => path.join(pathSegment, location)
   const readdirPreserveRelativePath = location => fs.readdirSync(location).map(prependPathSegment(location))
   const readdirRecursive = location => readdirPreserveRelativePath(location)
       .reduce((result, currentValue) => fs.statSync(currentValue).isDirectory()
           ? result.concat(readdirRecursive(currentValue))
           : result.concat(currentValue), [])
   const screenshots = readdirRecursive(path.resolve(screenshotsDir)).filter(file => {
       return file.indexOf('.png') > -1
   })
   const featuresList = Array.from(new Set(screenshots.map(x => x.match(/[\w-_.]+\.feature/g)[0])))
   featuresList.forEach(feature => {
       screenshots.forEach(screenshot => {
           const regex = /(?<=\ --\ ).*?((?=\ \(example\ \#\d+\))|(?=\ \(failed\)))/g</pre>
           const [scenarioName] = screenshot.match(regex)
           console.info(chalk.blue(scenarioName))
           console.log(featuresList)
           console.log(feature)
           console.log(cucumberReportMap)
           const myScenarios = cucumberReportMap[feature][0].elements.filter(
               e => scenarioName.includes(e.name)
           )
           if (!myScenarios) { return }
```

```
let foundFailedStep = false
        myScenarios.forEach(myScenario => {
            if (foundFailedStep) {
                return
            }
            let myStep
            if (screenshot.includes('(failed)')) {
                myStep = myScenario.steps.find(
                    step => step.result.status === 'failed'
                )
            } else {
                myStep = myScenario.steps.find(
                    step => step.name.includes('screenshot')
                )
            }
            if (!myStep) {
                return
            }
            const data = fs.readFileSync(
                path.resolve(screenshot)
            )
            if (data) \{
                const base64Image = Buffer.from(data, 'binary').toString('base64')
                if (!myStep.embeddings) {
                    myStep.embeddings = []
                    myStep.embeddings.push({ data: base64Image, mime_type: 'image/png' })
                    foundFailedStep = true
                }
            }
        })
        //Write JSON with screenshot back to report file.
        fs.writeFileSync(
            path.join(cucumberJsonDir, cucumberReportFileMap[feature]),
            JSON.stringify(cucumberReportMap[feature], null, jsonIndentLevel)
        )
   })
})
```

The cucumber-json-merge utility may be handy to merge the results of each feature, so they can be then submitted to Xray as one single file.

Next, is an example of a shell script with all these steps.

example of a Bash script to run the tests and produce a unified Cucumber JSON report

#!/bin/bash
rm -f cypress/cucumber-json/*
npm run test
npm run attach_screenshots
cucumber-json-merge -d cypress/cucumber-json/

After running the tests, results can be imported to Xray via the REST API, or the **Import Execution Results** action within the Test Execution, or by using one of the available CI/CD plugins (e.g. see an example of Integration with Jenkins).

curl -H "Content-Type: application/json" -X POST -u admin:admin --data @"report.json" http://jiraserver.example. com/rest/raven/1.0/import/execution/cucumber

}



A new Test Execution will be created (unless you originally exported the Scenarios/Scenario Outlines from a Test Execution).



One of the tests fails (on purpose).

The execution screen details of the Test Run will provide overall status information and Gherkin statement-level results, therefore we can use it to analyze the failing test.

Tests

								+	Add ¥
Overall Ex	ecution Statu	ıs						_	
3 PASS	s 1 fail								_
Total Tests	s: 4								
Ŧ	Filter(s)								
·								Show 100 V entries Col	umns 🗸
	🔺 Rank	Key	🜲 Summary	🝦 Test Type	#Req	#Def	Assignee	🔷 Status	
	1	CALC- 7903	Login With Invalid Credentials Should Fail	Cucumber	1	0	Administrator	FAIL	
	2	CALC- 7902	Invalid Login	Cucumber	1	0	Administrator	PASS	Execution Details
	3	CALC- 7901	Valid Login	Cucumber	1	0	Administrator	PASS	EXECUTE INLINE PASS
	4	CALC- 7904	Valid Logout	Cucumber	1	0	Administrator	PASS	TODO EXECUTING
Showing 1	to 4 of 4 en	ntries						First Previous 1 Ne	xt ABORTED

A given example can be expanded to see all Gherkin statements and, if available, it is possible to see also the attached screenshot(s).

Calculator / Test E .ogin With Inv	ecution: CALC-7916 / Test: CALC-7903 alid Credentials Should Fail	ú	Import Execution Results	Export to Cucumber	Return to Test Execution	Next
None						
Test Issue Links	(1)					^
tests	• CALC-7905 As a user, I can login the application				۵	OPEN
Custom Fields						~
There are no Test	Run Custom Fields defined.					
Test Details						^
Test Type:	Cucumber					
Scenario Type:	Scenario Outline					
Scenario.	2 Wihen user "cusername" logs in with password " <password "cpassword"<="" td=""> 3 Then error page should be open 4 5 Examples: 6 username password 7 invalid mode 8 demo invalid 9 invalid invalid 10 demo imode </password>	words "				
Examples						^
<userna< td=""><td>ne> <passw< td=""><td>vord></td><td></td><td>Duratio</td><td>n Status</td><td></td></passw<></td></userna<>	ne> <passw< td=""><td>vord></td><td></td><td>Duratio</td><td>n Status</td><td></td></passw<>	vord>		Duratio	n Status	
invalid	mode			1513.000 m	PASS	
demo	invalid			779.000 m	PASS	
invalid	invalid			858.000 m	PASS	
🔶 🕨 demo	mode			4783.000 m	FAIL	

Exam	les			
	<username></username>	<pre>cpassword></pre>	Duration	Status
	invalid	mode	1513.000 ms	PASS
	demo	invalid	779.000 ms	PASS
	invalid	invalid	858.000 ms	PASS
	demo	mode	4783.000 ms	FAIL
	Steps			
	Given browser is opened to login page		176.000 ms	PASS
	When user "demo" logs in with password "mode"		612.000 ms	PASS
	Then error page should be open		(1) 3995.000	FAIL
	<pre>MesertileAfron: Timed out retrying respected ' ' to have text 'Error Page', but the text was 'Welcome Page' + expected - actual - 'Welcome Page' + 'Error Page' at Object.toBeUnsuccessful (https://robotwebdemo.herokuapp.com/_cypres at Context.eval (https://robotwebdemo.herokuapp.com/_cypress/tests?p=: at Context.eval (https://robotwebde</pre>	ss/tests?p=features/1_CALC-7905.feature:134:33) Ceatures/1_CALC-7905.feature:26:41) comcypress/test2p=features/1_CALC-7905.feature:10674:9) Features/1_CALC-7905.feature:10015:35)	Same and Same	

Note: in this case, the bug was on the Scenario Outline example which was using a valid username/password combination.

Results are reflected on the covered item (e.g. Story). On the issue screen, coverage now shows that the item is OK based on the latest testing results which can also be tracked within the Test Coverage panel bellow.

Ē		Calcul As a	lator / CALC a user, l	c-7905 can logi	n the ap	oplication					
	• Edit	C	Comment	Assign	More 🗸	Start Progress	Close Issue	Admin 🗸			
~	Details	S									
-	Type:			Story			Status:		OPEN (View Wor	cflow)	
I	Priority	y:		🔶 Major			Resolution	:	Unresolved		
	Affects	s Vers	sion/s:	None			Fix Version	ı/s:	None		
(Compo	onent	/s:	None							
1	Labels	:		None							
I	Requir	emen	t Status:		NOK						
× 1	Descri As a u: Test C	iption ser, I o overa	can login the	e application							
								Create Test	Create Sub-T	est Execution	+ Link ~
	TEST	COVE	RAGE FOR THE	FOLLOWING A	NALYSIS SCOI	ЪЕ					
	Sc	ope: '	Version; Ve r	rsion: None -	latest exec	ution; Environme	nt: All Environme	ents 🔻			NOK
		Ţ Filt	ter(s)								
	·								Show 1	0 ✔ entries	Columns -
		∳ P	Status	Resolution	🔺 Key	Summar	y		Test Runs	Test Status	
		٠	OPEN	Unresolved	CALC-7	7901 Valid Logi	n		≣0	PASS	
		٠	OPEN	Unresolved	CALC-7	902 Invalid Log	jin		≣0	PASS	
		٠	OPEN	Unresolved	CALC-7	903 Login With	n Invalid Credentia	als Should Fai	ii ≣0	FAIL	

Using Git or other VCS as master

You can edit your .feature files using your IDE outside of Jira (eventually storing them in your VCS using Git, for example) alongside the remaining test code.

In any case, you'll need to synchronize your .feature files to Jira so that you can have visibility of them and report results against them.

The overall flow would be something like this:

- 1. look at the existing "requirement"/Story issue keys to guide your testing; keep their issue keys
- 2. specify Cucumber/Gherkin .feature files in your IDE supporting Cypress and store it in Git, for example
- 3. implement the code related to Gherkin statements/steps and store it in Git, for example

- 4. import/synchronize the .feature files to Xray to provision or update corresponding Test entities
- export/generate feature files from Jira, so that they contain references to Tests and requirements in Jira
 checkout the Cypress related code from Git
 run the tests in the CI

- 8. import the results back to Jira

Usually, you would start by having a Story, or similar (e.g. "requirement"), to describe the behavior of a certain feature and use that to drive your testing.

Edit Comment Asign More Start Progress Close issue Admin * Protry: Story Starts: Concording: None Requirement Status: UNCOVERED Create Sub-Test Execution * Uncoverage Create Sub-Test Execution * Uncoverage Create Sub-Test Execution * Uncoverage * Create Sub-Test Execution * Create Sub	Calculator / CAl	LC-7905 I can login the a	application							
Portails Type: Story Statu: CERENT Conserved Advects Version/s: None Labela: None Labela: None V Description As a user, I can login the application * Test Coverage Create Sub-Test Execution * C	Edit Q Commen	nt Assign More 🗸	Start Progress	Close Issue	Admin 🖌					
Type: Story Status: GEEI (View Workflow) Priority: Major Rasolution: Unresolved Affects Version/s: None Fix Version/s: None Labels: None UNCOVERED Version/s: None Priority: None INCOVERED Version/s: None * Description As a user, I can login the application * * * Test Coverage	 Details 									
Picitiy: Major Resolution: Unresolved Affects Version/s: None Fix Version/s: None Component/s: None None Pecifipion As a user, I can logint the application * Test Coverage Create Test I Create Sub:-Test Execution * Line Provide Test I Create Sub:-Test Execution * Line Provide Test I Create Sub:-Test Execution * Line Provide Test I Create Sub:-Test Execution * Line Provide Test I Create Sub:-Test I Create Sub:-Test Execution * Line Provide Test I Create Sub:-Test I Cre	Type:	 Story 			Status:		OPEN (View Workflo	w)		
Affects Version/s: None Component/s: None Requirement Status: UNCOVERED Pescription As a user, I can login the application Create Test Create Sub-Test Execution Create Sub-Test Execution Create Test Create Sub-Test Execution Create Test Create Sub-Test Execution Create Sub-Test Execution Create Sub-Test Execution Create Sub-Test Execution Create Sub-Test Execution Create Sub-Test Execution Create Sub-Test Execution Create Sub-Test Execution Create Sub-Test Execution Create Sub-Test Execution Create Sub-Test Execution Create Sub-Test Execution Create Sub-Test Execution Create Sub-Test Execution Create Sub-Test Execution Create Sub-Test Execution Create Sub-Test Execution Create Sub-Test Execution Crea	Priority:	🔶 Major			Resolution:	ι	Inresolved			
Component/s: None Labels: None Requirement Status: UNCOVERED Pescription As a user, I can login the application * Test Coverage Create Status: Create Status: Create Status: Create Status: Calculator / CALC-7906 As a user, I can logout the application Pioring: Asign More × Start Progress Close Issue Admin × Pioring: None Component/s: None Labels: None Tree Coverage	Affects Version/s:	None			Fix Version,	/s: N	None			
Labels: None Requirement Status: UNCOVERED Description As a user, I can login the application Test Coverage Create Test Coverage	Component/s:	None								
Requirement Statu: UNCOVERED Description As a user, I can logout the application Calculator / CALC-7906 As a user, I can logout the application Calculator / CALC-7906 As a user, I can logout the application Etili Ocomment Assign More V Start Progress Close Issue Admin V Petails Type: Story Status: Outfit: Major Resolution: Unresolved Affects Version/s: None Labels: None Requirement Statu: UNCOVERED	Labels:	None	_							
Description As a user, I can login the application Test Coverage Create Test Create Sub-Test Execution (* Line) No Tests were found testing the requirement. Calculator / CALC-7906 As a user, I can logout the application Edit Ocomment Assign More Start Progress Close Issue Admin * Priority: Major Affects Version/s: None Component/s: None Label: None Label: None Eaquirement Status: UNCOVERED Description As a user, I can logout the application	Requirement Status:	UNCOVERED								
As a user, I can login the application Test Coverage	Description									
Test Coverage Create Sub-Test Execution Calculator / CALC-7906 Calculator / CALC-7906 As a user, I can logout the application Edit Comment Asign Major Start Progress Close Issue Admin ~ Priority: Major Resolution: Unresolved Affects Version/s: None Component/S: None Perciption As a user, I can logout the application Description As a user, I can logout the application	As a user, I can login t	he application								
Create Test Create Stub-Test Execution 1 create Stub-Test Execution	 Test Coverage 									
No tests were found testing the requirement. Ediculator / CALC-7906 A ca user, I can logout the application Cetile Type: Story Start Progress Close Issue Admin ~ Details Type: Story Start Resolution: Unresolved Affects Version/s: None Fix Version/s: None Component/s: None Labels: None Requirement Status: UNCOVERED Description As a user, I can logout the application							Create Test	Create Su	b-Test Execution	+ Link ~
Calculator / CALC-7906 As a user, I can logout the application	No Tests were found to	esting the requirement.								
As a user, I can logout the application Comment Asign More Start Progress Close Issue Admin Type: Story Starts Starts: COEN (View Workflow) Poscription As a user, I can logout the application <td>Calculator / CA</td> <td>LC-7906</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>	Calculator / CA	LC-7906								
Edit Q Comment Assign More v Start Progress Close Issue Admin v Type: Story Status: OPEN Type: Story Status: OPEN Priority: Major Resolution: Unresolved Affects Version/s: None Fix Version/s: None Component/s: None Fix Version/s: None Labels: None UncoVERED View Vorkflow: Pescription As a user, I can logout the application	📰 As a user,	I can logout th	e application							
Details Type: Story Status: OPER (View Workflow) Priority: Major Resolution: Unresolved Affects Version/s: None Fix Version/s: None Component/s: None Fix Version/s: None Labels: None Version/s: Version/s: None Pescription As a user, I can logout the application Version status: Version status: Version status: Test Coverage Version status Version status Version status Version status	Edit Q Commer	nt Assign More 🗸	Start Progress	Close Issue	Admin 🖌					
Type: Image: Story Status: Image: View Workflow) Priority: Image: Major Resolution: Unresolved Affects Version/s: None Fix Version/s: None Component/s: None None None Labels: None None None Requirement Status: UNCOVERED None	Details									
Priority: Major Resolution: Unresolved Affects Version/s: None Component/s: None Labels: None Requirement Status: UNCOVERED Description As a user, I can logout the application Test Coverage Item Cove	Type:	 Story 		Statu	IS:	OPEN (Vi	ew Workflow)			
Affects Version/s: None Component/s: None Labels: None Requirement Status: UNCOVERED Description As a user, I can logout the application Test Coverage	Priority:	😤 Major		Reso	lution:	Unresolve	ed			
Component/s: None Labels: None Requirement Status: UNCOVERED Description As a user, I can logout the application Test Coverage	Affects Version/s:	None		Fix V	ersion/s:	None				
Labels: None Requirement Status: UNCOVERED Description As a user, I can logout the application Test Coverage	Component/s:	None								
Requirement Status: UNCOVERED Description As a user, I can logout the application Test Coverage Instrument of the state o	Labels:	None								
Description As a user, I can logout the application Test Coverage	Requirement Status:	UNCOVERED								
Test Coverage	Description									
Test Coverage	As a user I can logout	t the application								
Test Coverage										
	Test Coverage									
Create Test Create Sub-Test Execution + Link ~					Crea	ate Test	Create Sub-Test E	xecution	+ Link ~	
No Tests were found testing the requirement.	No Tests were found t	testing the requirement.								

Having those to guide testing, we could then move to Cypress to describe and implement the Cucumber test scenarios.

In Cypress, test related code is stored inside the cypress/integration directory, which itself contains several other directories. In this case, we've organized them as follows:

• cypress/integration/common: step implementation files, in JavaScript.

cypress/integration/common/login.js

```
import { Given, When } from 'cypress-cucumber-preprocessor/steps';
import LoginPage from '../../pages/login-page';
import LoginResultsPage from '../../pages/login-results-page';
Given(/^browser is opened to login page$/, () => {
 LoginPage.visit();
});
When('user {string} logs in with password {string}', (username, password) => {
 LoginPage.enter_username(username);
 LoginPage.enter_password(password);
 LoginPage.pressLogin();
});
Then(/^welcome page should be open$/, () => {
    LoginResultsPage.expect().toBeSuccessful();
  });
Then(/^error page should be open$/, () => {
   LoginResultsPage.expect().toBeUnsuccessful();
  });
```

cypress/integration/common/logout.js

```
import { Given, When } from 'cypress-cucumber-preprocessor/steps';
import LoginPage from '../../pages/login-page';
import LoginResultsPage from '../../pages/login-results-page';
Given(/^browser is opened to login page$/, () => {
 LoginPage.visit();
});
When('user {string} logs in with password {string}', (username, password) => {
 LoginPage.enter_username(username);
 LoginPage.enter_password(password);
 LoginPage.pressLogin();
});
Then(/^welcome page should be open$/, () => {
   LoginResultsPage.expect().toBeSuccessful();
  });
Then(/^error page should be open$/, () => {
   LoginResultsPage.expect().toBeUnsuccessful();
  });
```

cypress/integration/pages: abstraction of different pages, somehow based on the page-objects model

```
    cypress/integration/pages/login.js
```

```
import LoginResultsPage from './login-results-page';
const USERNAME_FIELD = 'input[id=username_field]';
const PASSWORD_FIELD = 'input[id=password_field]';
const LOGIN_BUTTON = 'input[type=submit]';
const LOGIN_TEXT = 'LOGIN';
class LoginPage {
 static visit() {
   cy.visit('/');
  }
 static enter_username(username) {
   cy.get(USERNAME_FIELD)
     .type(username);
  }
  static enter_password(password) {
   cy.get(PASSWORD_FIELD)
     .type(password);
  }
 static pressLogin() {
   cy.get(LOGIN_BUTTON).contains(LOGIN_TEXT)
      .click();
   return new LoginResultsPage();
  }
}
export default LoginPage;
```

```
    cypress/integration/pages/logout-results-page.js
```

```
const RESULT_HEADER = 'h1';
class LogoutResultsPage {
    static expect() {
       return {
            toBeSuccessful: () => {
               cy.get(RESULT_HEADER).should('have.text', 'Login Page')
            },
        };
    }
}
export default LogoutResultsPage;
```

```
    cypress/integration/pages/welcome-page.js
```

```
import LoginPage from './login-page';
const LOGOUT_LINK = 'a';
const LOGOUT_TEXT = 'logout';
class WelcomePage {
  static visit() {
    cy.visit('/welcome.html');
  }
  static pressLogout() {
    cy.get(LOGOUT_LINK).contains(LOGOUT_TEXT)
    .click();
    return new LoginPage();
  }
}
export default WelcomePage;
```

cypress/integration/login: Cucumber .feature files, containing the tests as Gherkin Scenario(s)/Scenario Outline(s). Please note that
each "Feature: <...>" section should be tagged with the issue key of the corresponding "requirement"/story in Jira. You may need to add a prefix (e.
g. "REQ_") before the issue key, depending on a global Xray setting.

cypress/integration/login/login.feature 0

```
@REO_CALC-7905
Feature: As a user, I can login the applicaiton
Scenario: Valid Login
   Given browser is opened to login page
    When user "demo" logs in with password "mode"
   Then welcome page should be open
Scenario: Invalid Login
   Given browser is opened to login page
    When user "dummy" logs in with password "password"
    Then error page should be open
Scenario Outline: Login With Invalid Credentials Should Fail
   Given browser is opened to login page
    When user "<username>" logs in with password "<password>"
   Then error page should be open
    Examples:
        | username | password |
        | invalid | mode
        demo
                   | invalid |
```

0 cypress/integration/login/logout.feature

| invalid | invalid |

```
@REO CALC-7906
Feature: As a user, I can logout the application
Scenario: Valid Logout
    Given user is on the welcome page
    When user chooses to logout
    Then login page should be open
```

Before running the tests in the CI environment, you need to import your .feature files to Xray/Jira; you can invoke the REST API directly or use one of the available plugins/tutorials for CI tools.

```
zip -r features.zip cypress/integration/ -i \*.feature
curl -H "Content-Type: multipart/form-data" -u admin:admin -F "file=@features.zip" "http://jiraserver.example.
com/rest/raven/1.0/import/feature?projectKey=CALC"
```



Each Scenario of each feature will be created as a Test issue that contains unique identifiers, so that if you import once again then Xray can update the existent Test and don't create any duplicated tests.

Afterwards, you can export those features out of Jira, based on some criteria so they are properly tagged with corresponding issue keys; this is important because results need to contain these references.

You can then export the specification of the test to a Cucumber .feature file via the REST API, or the Export to Cucumber UI action from within the Test /Test Execution issue or even based on an existing saved filter. A plugin for your CI tool of choice can be used to ease this task.

So, you can either:

use the UI

	Calculator / CALC	-7901			
🖋 Edit	Q Comment	Assign	More 🗸	Start Progress	Resolve Issue
 Details Type: 		Test	Log work	d	Status:
Priority Affects Compo	r: 1 s Version/s: N onent/s: N	Medium one one	Rank to To Rank to Bo	op ottom	Resolut Fix Vers
Labels Descri Click to	ption	cypress/inte	Attach file Voters Stop watc	hing	
Test D Type:	etails Cu	ucumber	Create sul Convert to	b-task o sub-task	
Scenar Scenar	io Type: So io: Gi Wi Th	cenario iven brows nen user " nen welcom	Move Link Clone	55	oage sword "mode"
 Pre-Co Test So 	onditions		Delete Trigger Je Trigger Je	nkins job nkins job an	
This te	st is not associate	d with Test	Reset Test Export to Export Tes Export Tes	tRunStatus Cucumber st to XML st Runs to CSV	
niis te	at is not associate	u with rest			

• use the REST API (more info here)

```
o #!/bin/bash
rm -f features/*.feature
curl -u admin:admin "http://jiraserver.example.com/rest/raven/1.0/export/test?keys=CALC-7905;CALC-
7906&fz=true" -o features.zip
unzip -o features.zip -d features
```

• use one of the available CI/CD plugins (e.g. see an example of Integration with Jenkins)

For CI only purpose, we will export the features to a new temporary directory named features/ on the root folder of your Cypress project (we'll need to tell Cypress to use this folder). Please note that while implementing the tests, .feature files should be edited inside the cypress/integration/login folder, in this case;

After being exported, the created .feature(s) will contain references to the Test issue keys, eventually prefixed (e.g. "TEST_") depending on an Xray global setting, and the covered "requirement" issue key, if that's the case. The naming of these files is detailed in Export Cucumber Features.

```
features/1_CALC-7905.feature
@REO CALC-7905
Feature: As a user, I can login the application
       #As a user, I can login the application
       @TEST_CALC-7903 @cypress/integration/login/login.feature
       Scenario Outline: Login With Invalid Credentials Should Fail
               Given browser is opened to login page
               When user "<username>" logs in with password "<password>"
               Then error page should be open
                       Examples:
                               | username | password |
                               | invalid | mode
                               | demo | invalid
                               | invalid | invalid
                                                     demo mode
       @TEST_CALC-7902 @cypress/integration/login/login.feature
       Scenario: Invalid Login
               Given browser is opened to login page
               When user "dummy" logs in with password "password"
               Then error page should be open
       @TEST_CALC-7901 @cypress/integration/login/login.feature
       Scenario: Valid Login
               Given browser is opened to login page
               When user "demo" logs in with password "mode"
               Then welcome page should be open(base)
```

To run the tests and produce Cucumber JSON reports(s), we can either use npm or cypress command directly.



This will produce one Cucumber JSON report in cypress/cucumber-json directory per each .feature file.

The cypress-cucumber-preprocessor package, as of v4.0.0, does not produce reports containing the screenshots embedded.

However, the following script (credits to the user that provided it on GitHub) can be used to update the previous JSON reports so that they contain the screenshots of the failed tests.

attach_screenshots.js

```
const fs = require('fs-extra')
const path = require('path')
const chalk = require('chalk')
const cucumberJsonDir = './cypress/cucumber-json'
const cucumberReportFileMap = {}
const cucumberReportMap = {}
const jsonIndentLevel = 2
```

```
const ReportDir = './cypress/reports/cucumber-report'
const screenshotsDir = './cypress/screenshots'
getCucumberReportMaps()
addScreenshots()
//Mapping cucumber json files from the cucumber-json directory to the features
function getCucumberReportMaps() {
   const files = fs.readdirSync(cucumberJsonDir).filter(file => {
       return file.indexOf('.json') > -1
   })
   files.forEach(file => {
       const json = JSON.parse(
           fs.readFileSync(path.join(cucumberJsonDir, file))
       if (!json[0]) { return }
       const [feature] = json[0].uri.split('/').reverse()
       cucumberReportFileMap[feature] = file
       cucumberReportMap[feature] = json
   })
}
//Adding screenshots to the respective failed test steps in the feature files
function addScreenshots() {
   const prependPathSegment = pathSegment => location => path.join(pathSegment, location)
   const readdirPreserveRelativePath = location => fs.readdirSync(location).map(prependPathSegment(location))
   const readdirRecursive = location => readdirPreserveRelativePath(location)
       .reduce((result, currentValue) => fs.statSync(currentValue).isDirectory()
           ? result.concat(readdirRecursive(currentValue))
            : result.concat(currentValue), [])
   const screenshots = readdirRecursive(path.resolve(screenshotsDir)).filter(file => {
       return file.indexOf('.png') > -1
   })
   const featuresList = Array.from(new Set(screenshots.map(x => x.match(/[\w-_.]+\.feature/g)[0])))
   featuresList.forEach(feature => {
       screenshots.forEach(screenshot => {
           const regex = /(?<=\ --\ ).*?((?=\ \(example\ \#\d+\))|(?=\ \(failed\)))/g</pre>
           const [scenarioName] = screenshot.match(regex)
            console.info(chalk.blue('\n Adding screenshot to cucumber-json report for'))
           console.info(chalk.blue(scenarioName))
           console.log(featuresList)
           console.log(feature)
           console.log(cucumberReportMap)
           const myScenarios = cucumberReportMap[feature][0].elements.filter(
               e => scenarioName.includes(e.name)
           )
           if (!myScenarios) { return }
            let foundFailedStep = false
           myScenarios.forEach(myScenario => {
               if (foundFailedStep) {
                   return
                }
                let myStep
                if (screenshot.includes('(failed)')) {
                    myStep = myScenario.steps.find(
                        step => step.result.status === 'failed'
                    )
                } else {
                   myStep = myScenario.steps.find(
                       step => step.name.includes('screenshot')
                    )
                }
                if (!myStep) {
                   return
                }
```

```
const data = fs.readFileSync(
                    path.resolve(screenshot)
                )
                if (data) {
                    const base64Image = Buffer.from(data, 'binary').toString('base64')
                    if (!myStep.embeddings) {
                        myStep.embeddings = []
                        myStep.embeddings.push({ data: base64Image, mime_type: 'image/png' })
                        foundFailedStep = true
                    }
                }
            })
            //Write JSON with screenshot back to report file.
            fs.writeFileSync(
                path.join(cucumberJsonDir, cucumberReportFileMap[feature]),
                JSON.stringify(cucumberReportMap[feature], null, jsonIndentLevel)
            )
       })
   })
}
```

The cucumber-json-merge utility may be handy to merge the results of each feature, so they can be then submitted to Xray as one single file.

Next, is an example of a shell script with all these steps.

example of a Bash script to run the tests and produce a unified Cucumber JSON report

#!/bin/bash

```
rm -f cypress/cucumber-json/*
npm run test
npm run attach_screenshots
cucumber-json-merge -d cypress/cucumber-json/
```

After running the tests, results can be imported to Xray via the REST API, or the **Import Execution Results** action within the Test Execution, or by using one of the available CI/CD plugins (e.g. see an example of Integration with Jenkins).

example of a Bash script to import results using the standard Cucumber endpoint

#!/bin/bash

curl -H "Content-Type: application/json" -X POST -u admin:admin --data @"report.json" http://jiraserver.example. com/rest/raven/1.0/import/execution/cucumber



A new Test Execution will be created (unless you originally exported the Scenarios/Scenario Outlines from a Test Execution).



One of the tests fails (on purpose).

The execution screen details of the Test Run will provide overall status information and Gherkin statement-level results, therefore we can use it to analyze the failing test.

Tests

									+ Add	~
Overall Ex	xecution Stat	us						19191M		_
3 PAS	s 1 fail									
Total Test	ts: 4									
Ŧ	Filter(s)									
, 								Show 100 V entries	Columns	•
	A Rank	Key	Summary	🜲 Test Type	#Req	#Def	Assignee	🔶 Status		
	1	CALC- 7903	Login With Invalid Credentials Should Fail	Cucumber	1	0	Administrator	FAIL		
	2	CALC- 7902	Invalid Login	Cucumber	1	0	Administrator	PASS	Ξ	Execution Details
	3	CALC- 7901	Valid Login	Cucumber	1	0	Administrator	PASS	EX	PASS
	4	CALC- 7904	Valid Logout	Cucumber	1	0	Administrator	PASS		TODO EXECUTING
Showing	1 to 4 of 4 e	ntries						First Previous	1 Next	ABORTED

A given example can be expanded to see all Gherkin statements and, if available, it is possible to see also the attached screenshot(s).

Calculator / Te Login With	st Execution: CALC-7916 / Test: CALC-7903 Invalid Credentials Should Fail	ų	Import Execution Results	Export to Cucumber	Return to Test Execution	Next)
None						
Test Issue L	inks (1)					^
tests	CALC-7905 As a user, I can login the application				۲	OPEN
Custom Fiel	ds					~
There are no 1	Fest Run Custom Fields defined.					
Test Detail	S					^
Test Type:	Cucumber					
Scenario Ty	ype: Scenario Outline					
	1 Given provser is opened to logi n with password " 2 When user "username". Logs in with password " 3 Then error page should be open 4 Examples: 6 username password 7 invalid mode 8 demo invalid 9 invalid invalid 10 demo mode					
Examples						~
<us< td=""><td>ername> <password></password></td><td></td><td></td><td>Durat</td><td>ion Status</td><td></td></us<>	ername> <password></password>			Durat	ion Status	
inva	alid mode			1513.000	ms PASS	
▶ den	no invalid			779.000	ms PASS	
inva	alid invalid			858.000	ms PASS	
🔶 🕨 den	no mode			4783.000	ms FAIL	

Exam	ixamples									
-										
	<username></username>	<pre>cpassword></pre>	Duration	Status						
	invalid	mode	1513.000 ms	PASS						
	demo	invalid	779.000 ms	PASS						
	invalid	invalid	858.000 ms	PASS						
•	demo	made	4783.000 ms	FAIL						
	Steps									
	Given browser is opened to login page		176.000 ms	PASS						
	When user "demo" logs in with password "mode"		612.000 ms	PASS						
	Then error page should be open		(1) 3995.000	FAIL						
	AssertionTrory Timed out retrying: espected ' ' to have text 'Error Page', but the text was 'Welcome Page' + expected - actual -'Welcome Page'		evidence_step_32_0.ang							
	+'Error Page'									
	at Object.toBeUnsuccessful (https://robotwebdemo.herokuapp.com/_cypress/tests?p=features/1_CALC-7905.feature:134:33) at Context.eval (https://robotwebdemo.herokuapp.com/_cypress/tests?p=features/1_CALC-7905.feature:136:41) at Context.resolveAndRunStepDefinition (https://robotwebdemo.herokuapp.com/_cypress/tests?p=features/1_CALC-7905.feature:10674:9) at Context.eval (https://robotwebdemo.herokuapp.com/_cypress/tests?p=features/1_CALC-7905.feature:10674:9)									

Note: in this case, the bug was on the Scenario Outline example which was using a valid username/password combination.

Results are reflected on the covered item (e.g. Story). On its issue screen, coverage now shows that the item is OK based on the latest testing results, that can also be tracked within the Test Coverage panel bellow.

		Calcula	ator / CALC I USER, I	c-7905 can logi	n the ap	plication							
Ø	Edit	Q	Comment	Assign	More 🗸	Start Progress	Close Issue	Admin 🗸					
⊻ De	etails												
Ту	pe:			Story			Status:		OPEN (View Wor	kflow)			
Pr	iority	:		🔶 Major			Resolution	:	Unresolved				
Af	Affects Version/s:		None			Fix Version/s:		None					
Сс	Component/s: N		None										
La	bels			None									
Re	equire	ement	Status:		NOK								
⊻ De	escri	ption											
As	a us	ser, I c	an login the	e application									
✓ Te	✓ Test Coverage												
								Create Test	Create Sub-T	Test Execution	+ Link ×		
	TEST COVERAGE FOR THE FOLLOWING ANALYSIS SCOPE												
Scope: Version; Version: None - latest execution; Environment: All Environments -								NOK					
	₩ Filter(s)												
(-								Show	10 🗸 entries	Columns 🗸		
		₽	Status	Resolution	🔺 Key	🔶 Summary	/		Test Runs	Test Status			
		٠	OPEN	Unresolved	CALC-7	901 Valid Logir	ı		≣0	PASS			
		Ŧ	OPEN	Unresolved	CALC-7	902 Invalid Log	jin		≣0	PASS			
		Ŧ	OPEN	Unresolved	CALC-7	903 Login With	Invalid Credentia	als Should Fai	∎0	FAIL			

References

- Cypress
 Cypress documentation
 cypress-cucumber-example
 issue related to adding screenshots to the cucumber JSON report(s)