

# Framework Structure

**Framework:** WDIO with Cucumber

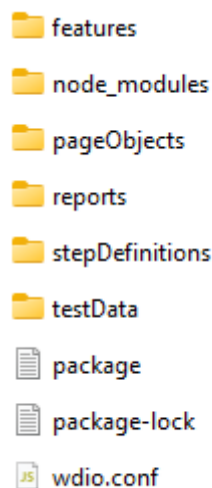
**Programming Language:** JavaScript

**Test Data:**

Examples - Some common test data are passed using 'Examples' by using Cucumber keyword

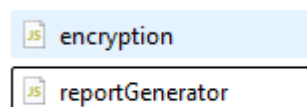
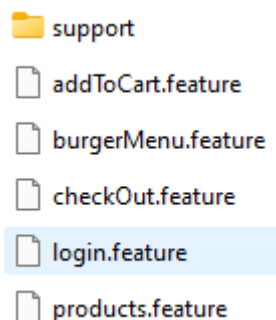
JSON - User credentials and Products details were stored in JSON files.

**Complete Folder Structure:**

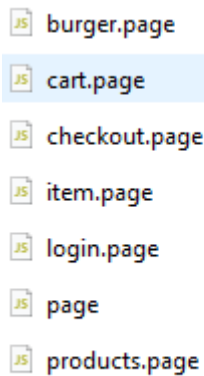


**The root folder contains the below folders and files:**








1. **Features**: Contains all the feature files and a support folder. Each feature file contains the scenarios specific to that feature. Support folder includes, 'encryption.js' and 'reportGenerator.js' files to decrypt the password given in testdata and to generate report.



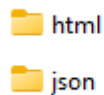
2. **Node Modules**: Contains all the dependencies and the other packages that were installed during setup. Need to add this in .gitignore not to upload in git to avoid space issue
3. **pageObjects**: Contains files specific to each page and the actual code with respect to perform user actions and validations.





A list of JavaScript files representing page objects. Each item consists of a small 'JS' icon followed by the filename. The 'cart.page' item is highlighted with a light blue background.

-  burger.page
-  cart.page
-  checkout.page
-  item.page
-  login.page
-  page
-  products.page

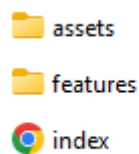
4. **Reports** : Contains html reports and json reports.






Two folders are shown, each with a yellow folder icon. The first folder is named 'html' and the second is named 'json'.

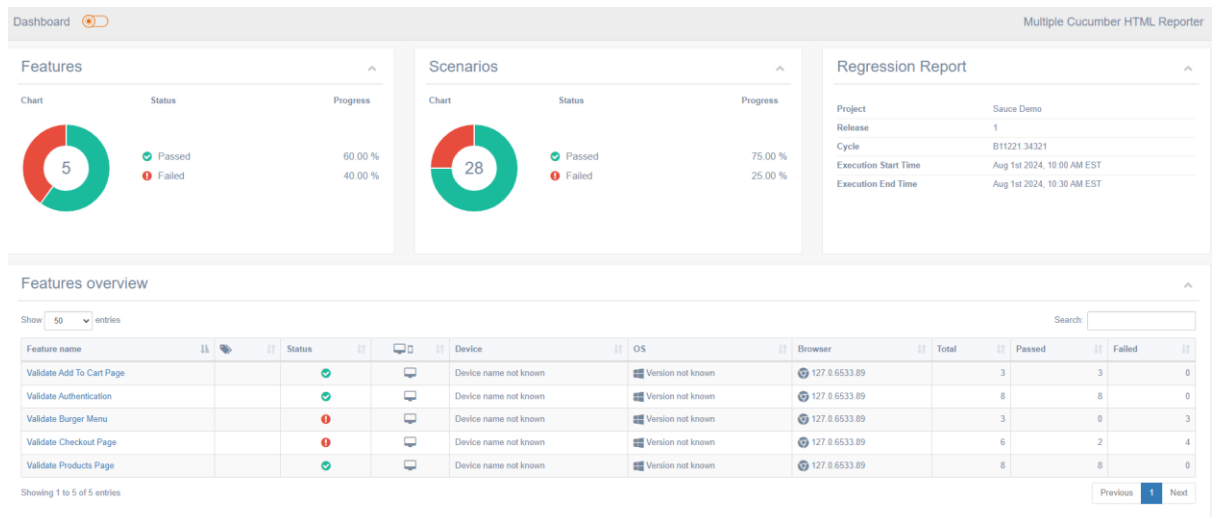
-  html
-  json

Html report folder contains the final report(index) in readable format.




Three items are shown: two folders and one file. The first two are folders with yellow icons, named 'assets' and 'features'. The third is a file with a Chrome icon, named 'index'.


-  assets
-  features
-  index




5. **stepDefinitions**: Contains 'steps.js' used for mapping the steps in feature file with the actual code in pages.

 **steps**

6. **testData** : Contains the test data for login(encrypted) and products details in JSON format.

 **products**

 **testData**

```

config > testData > {} testData.json > [] empty > {} 0
1  {
2    "valid":[
3      {
4        "username":"standard_user",
5        "password":"c4ea80ad40022d790acb21ec6761c9b9"
6      }
7    ],
8    "invalid":[
9      {
10       "username":"invalid_user",
11       "password":"wrongpassword"
12     }
13   ],
14   "empty":[
15     {
16       "username":"","
17       "password":""
18     }
19   ],
20   "required":[
21     {
22       "username":"Username is required",
23       "password":"Password is required"
24     }
25   ],
26   "error":[
27     {
28       "errText":"Username and password do not match any user in this service"
29     }
30   ]
31 }

```

7. **package.json** – This is a json file which contains the devDependencies needed for project setup and the test details.

```

{
  "devDependencies": {
    "@wdio/cli": "^8.39.1",
    "@wdio/cucumber-framework": "^8.39.0",
    "@wdio/local-runner": "^8.39.1",
    "@wdio/spec-reporter": "^8.39.0",
    "multiple-cucumber-html-reporter": "^3.7.0",
  },
  ▶ Debug
  "scripts": {
    "regression": "wdio run wdio.conf.js"
  },
  "dependencies": {
    "wdio-cucumberjs-json-reporter": "^5.2.1"
  }
}

```

8. **package-lock** – This is an autogenerated file when doing the initial setup and used to lock the version in order to avoid version discrepancies of the packages during installation/the next setup

```
{
  "name": "config",
  "lockfileVersion": 2,
  "requires": true,
  "packages": {
    "": {
      "devDependencies": {
        "@wdio/allure-reporter": "^8.39.0",
        "@wdio/cli": "^8.39.1",
        "@wdio/cucumber-framework": "^8.39.0",
        "@wdio/local-runner": "^8.39.1",
        "@wdio/spec-reporter": "^8.39.0",
        "allure-commandline": "^2.29.0",
        "wdio-html-nice-reporter": "^8.1.6"
      }
    },
    "node_modules/@babel/code-frame": {
      "version": "7.24.7",
      "resolved": "https://registry.npmjs.org/@babel/code-frame/-/code-frame-7.24.7.tgz",
      "integrity": "sha512-BcVY01W3Yqj84GzdPjuzKjIC6hsVpR/z2JZpcYg12hv4rltrWIA3xHpcNdI4oFKTLfuiEb0EtdvdvKPjvh3Njg==",
      "dev": true,
      "dependencies": {
        "@babel/highlight": "^7.24.7",
        "picocolors": "^1.0.0"
      },
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/helper-validator-identifier": {
      "version": "7.24.7",
      "resolved": "https://registry.npmjs.org/@babel/helper-validator-identifier/-/helper-validator-identifier-7.24.7.tgz",
      "integrity": "sha512-xR67Na/ZqXg80F5uV6+EN/677FVC05KBhYnV2iKMznE2hF9XWNRwG5KAXKtS0b8abQv8zLNaPGf35FajmFA=="
    }
  }
}
```

9. **wdio.conf** – Skeleton/Core of this framework where the entire configuration would be mentioned. Below are some of the main configurations that are updated in the config file.
- Runner** – whether execution is local or any cloud
  - Framework** – cucumber
  - Reporter** – uses 'spec' as well as 'cucumberjs-json'

```
reporters: ['spec',
  ['cucumberjs-json', {
    jsonFolder: './reports/json/',
    language: 'en',
  }],
],
```

- CucumberOpts** – Includes files that are required for execution like stepDefinition and hooks

```
cucumberOpts: {
  // <string[]> (file/dir) require files before executing features
  require: ['./stepDefinitions/*.js'],
}
```

- e. **Hooks** – To include code which needs to be performed at specific stage of testing cycle. Here is an example snapshot of 'onComplete' hook to rename the generated report to its feature

```
onComplete: function (exitCode, config, capabilities, results) {  
  // Run your JavaScript file after the tests are complete  
  const { exec } = require('child_process');  
  exec('node ../config/features/support/reportGenerator.js', (error, stdout, stderr) => {  
    if (error) {  
      console.error(`Error executing script: ${error}`);  
      return;  
    }  
    if (stderr) {  
      console.error(`stderr: ${stderr}`);  
      return;  
    }  
    console.log(`stdout: ${stdout}`);  
  });  
},
```