Cypress Integration Steps

Creating Seamless Automated Testing Workflows

Neeharica Madanu

Studio-Ghibli

31 March 2024

# Cypress Frontend Testing

Cypress is a JavaScript-based open-source testing tool that offers a comprehensive framework for end-to-end, integration, and unit testing of modern web applications. Key features of Cypress include real-time reloading, automatic waiting, time travel, and network traffic control. Real-time reloading enables developers to see changes as they write tests, while automatic waiting eliminates the need for manual waits and sleeps. Time travel allows developers to debug their tests by stepping through each command, and network traffic control helps developers test their application's response to various network conditions.

# Cypress Cloud

Cypress Cloud is a secure and scalable platform developed by the creators of Cypress for running tests at scale. This platform supports cross-browser testing, parallel test execution, and video recording, which streamlines the test management and analysis process. Cypress Cloud also integrates seamlessly with popular CI/CD tools, making it a valuable resource for optimising testing efforts.

# Mochawesome Reporter

Mochawesome Reporter is a sophisticated tool that works flawlessly with Mocha, a feature-rich JavaScript testing framework for Node.js and browsers. This tool provides advanced reporting capabilities that enable users to create detailed and visually appealing HTML reports, making it easier to analyze test results. Key features of Mochawesome Reporter include the ability to generate comprehensive HTML reports, customization options, visual appeal, and seamless integration with Mocha. Users can easily install Mochawesome Reporter as a dependency in their Node.js project and configure Mocha to use the reporter during test runs. Mochawesome Reporter is an excellent choice for organizations looking to enhance their test reporting capabilities within the Mocha framework, providing a user-friendly and visually rich experience for analyzing test results.

# Cypress Integration Steps

To incorporate Cypress into the project, follow these steps:

Step 1: Install Cypress by running the command "npm install cypress --save-dev" in the project's root directory.

Step 2: Initialise Cypress by running "npx cypress open". This will establish the necessary folder structure and files for Cypress within the project.

Step 3: Create a new spec file within the "cypress/integration" directory to write the test cases. For instance, we could create a file named "example\\_spec.js" and employ Cypress syntax to write the test cases.

Step 4: Execute the Cypress tests using the command "npx cypress run". This will run all the test cases defined in the spec files.

Step 5: Customise Cypress configurations by adding a "cypress.json" file in the project's root directory. This file enables us to set diverse options for our tests.

Step 6: Integrate Cypress with our CI/CD pipeline by adding the Cypress run command to our pipeline script. This will enable the tests to be executed automatically as part of our pipeline.

# Mochawesome reporter Integration Steps

To incorporate Cypress test results, follow these steps:

Step 1: Install the Mochawesome reporter and the Mochawesome-merge package to amalgamate multiple test results into a single report.

```

npm install mochawesome mochawesome-merge --save-dev

```

Step 2: In the cypress.json file, include the following configuration to generate Mochawesome reports:

```css

{

"reporter": "mochawesome",

"reporterOptions": {

"reportDir": "cypress/reports",

"overwrite": false,

"html": false,

"json": true

}

}

```

Step 3: Add the following scripts to the package.json to run Cypress tests and merge the reports:

```json

{

"scripts": {

"cy:run": "cypress run",

"cy:merge": "mochawesome-merge cypress/reports/\*.json > cypress/reports/mochareports/report.json"

}

}

```

Step 4: Execute the Cypress tests using the following command:

```css

npm run cy:run

```

Upon completion of the tests, merge the generated reports using:

```css

npm run cy:merge

```

Step 5: View the merged report in the cypress/reports/mochareports/report.json file. Moreover, we can employ a tool like Mochawesome Report Generator to transform the JSON report into HTML for better comprehensibility.

# Cypress Cloud Integration Steps

To integrate Cypress Cloud with the project, please follow these steps:

Step 1: Set up Cypress by installing it by running the command:

```

npm install cypress --save-dev

```

Step 2: Configure Cypress by creating a cypress.json file in the project's root directory and adding the following configuration:

```

{

"projectId": "your\_project\_id"

}

```

Step 3: Install the Cypress Dashboard Service by running:

```

npm install cypress-dash --save-dev

```

Step 4: Add scripts to the package.json file by adding the following:

```

{

"scripts": {

"cypress:open": "cypress open",

"cypress:run": "cypress run --record --key <your\_record\_key>"

}

}

```

Replace <your\\_record\\_key> with the actual record key.

Step 5: Run tests on Cypress Cloud by executing the command:

```

npm run cypress:run

```

Step 6: Access the Cypress Dashboard to view recorded test results and other details after running the tests.