

WD401 L4

Submitted by - Swanand Bhuskute

Problem statement:

1. To configure the testing framework for my WD301 project for unit testing
 2. To write unit tests and understand how testing works.
 3. Understand code coverage
 4. Implementation of Integration tests configured using Cypress
 5. To configure GitHub actions to run automatically when a new code is pushed.
-

→ I chose Jest as my unit testing framework as it was mentioned in the level and I also had experience of it integrating with WD201 projects (Todo app and Capstone LMS). In WD201, we integrated it with NodeJs.

This time I saw an opportunity to integrate jest with a frontend library like React, specifically Vite. It took some time initially to make it work as expected, as it was not taught in previous levels and courses, so it was a very good learning opportunity. So, finally after searching a few resources over youtube, google and chatgpt, I managed to integrate Jest with my WD301 SportsLive Vite project.

For now, I have only implemented 3 unit tests, that only checks rendering of the components. I will integrate more later as I research more. (I may choose Vitest over Jest as it more easily integrates with Vite projects. And then write more robust tests.

Advantages of Jest

- Jest is designed to be easy to set up and use. It has minimal configuration and provides rich set of features.
- Jest works seamlessly with React, and its ecosystem includes tools like `react-testing-library` which make it easier to test React components.
- Jest's snapshot testing feature allows you to take a snapshot of your UI component and compare it to the previous snapshot. This is particularly useful for detecting unintended changes in your UI.
- Jest has built-in support for mocking functions, modules, and timers.
- Jest runs tests in parallel, speeding up the testing process.

Potential alternatives and future options:

- Mocha and Chai - Karma - Jasmine
- Testing Library (React Testing Library) - Vitest

➡ **Jest** is an excellent choice for unit testing in React applications due to its ease of use, performance, built-in features, and strong community support. However, as the JavaScript ecosystem evolves, other tools like **Vitest**, which are designed to work with modern build tools like **Vite**, might become more appealing for certain projects.

Unit tests for components:

```
Windows PowerShell
PS E:\Coding\ALL Projects\Sports-Live> npm test

> sports-live@0.0.0 test
> jest

PASS src/pages/articles/__tests__/articles.tsx (5.546 s)
  ● Console

    console.error
      Warning: A suspended resource finished loading inside a test, but the event was not wrapped in act(...).

      When testing, code that resolves suspended data should be wrapped into act(...):

      act(() => {
        /* finish loading suspended data */
      });
      /* assert on the output */

      This ensures that you're testing the behavior the user would see in the browser. Learn more at https://reactjs.org/link/wrap-tests-with-act

      at printWarning (node_modules/react-dom/cjs/react-dom.development.js:86:30)
      at error (node_modules/react-dom/cjs/react-dom.development.js:60:7)
      at warnIfSuspenseResolutionNotWrappedWithActDEV (node_modules/react-dom/cjs/react-dom.development.js:27604:7)
      at pingSuspendedRoot (node_modules/react-dom/cjs/react-dom.development.js:27189:3)

PASS src/pages/matches/__tests__/matches.tsx (5.54 s)
  ● Console

    console.error
      Warning: A suspended resource finished loading inside a test, but the event was not wrapped in act(...).

      When testing, code that resolves suspended data should be wrapped into act(...):

      act(() => {
        /* finish loading suspended data */
      });
      /* assert on the output */

      This ensures that you're testing the behavior the user would see in the browser. Learn more at https://reactjs.org/link/wrap-tests-with-act

      at printWarning (node_modules/react-dom/cjs/react-dom.development.js:86:30)
      at error (node_modules/react-dom/cjs/react-dom.development.js:60:7)
      at warnIfSuspenseResolutionNotWrappedWithActDEV (node_modules/react-dom/cjs/react-dom.development.js:27604:7)
      at pingSuspendedRoot (node_modules/react-dom/cjs/react-dom.development.js:27189:3)

PASS src/pages/teamAndSports/__tests__/teamandsport.tsx (5.775 s)
  ● Console

    console.error
      Warning: A suspended resource finished loading inside a test, but the event was not wrapped in act(...).

      When testing, code that resolves suspended data should be wrapped into act(...):

      act(() => {
        /* finish loading suspended data */
      });
      /* assert on the output */

      This ensures that you're testing the behavior the user would see in the browser. Learn more at https://reactjs.org/link/wrap-tests-with-act

      at printWarning (node_modules/react-dom/cjs/react-dom.development.js:86:30)
      at error (node_modules/react-dom/cjs/react-dom.development.js:60:7)
      at warnIfSuspenseResolutionNotWrappedWithActDEV (node_modules/react-dom/cjs/react-dom.development.js:27604:7)
      at pingSuspendedRoot (node_modules/react-dom/cjs/react-dom.development.js:27189:3)

Test Suites: 3 passed, 3 total
Tests:       3 passed, 3 total
Snapshots:  0 total
Time:        7.277 s
Ran all test suites.
PS E:\Coding\ALL Projects\Sports-Live> |
```

→ Yes it ran with few warnings, I worked on it and it says when we use APIs and promises and Suspense in our projects, we must wrap our tests in `act()` function which uses `async await` for promises and then run tests just fine. But I didn't get the proper configuration. I tried multiple changes in my `tsconfig.json` files, but it did not work as expected.

My project is set to target ES2020, which was one way of configuration, but still didn't work. So I left it for now.

Code Coverage

→ Code coverage is a metric that can help you understand how much of your source is tested. It's a very useful metric that can help you assess the quality of your test suite. It is a software testing strategy that's a part of continuous delivery (CD).

There are four major metrics used for performing code coverage:

- Statement coverage: The number of statements in a program that get executed
- Condition coverage: The number of Boolean expressions that get tested for a true or false value
- Function coverage: The number of defined functions that get called during the run of a program
- Line coverage: The number of lines of source code that get tested during the run of a program

Cypress - E2E testing tool

→ **Cypress** is a modern front-end testing tool built for the web. It is designed to be fast, easy to use, and reliable, making it a popular choice for developers who need to test their web applications.

Benefits of cypress/E2E testing tool

1. End-to-End Testing : Cypress is primarily used for end-to-end (E2E) testing. It allows you to simulate real user interactions with your application, ensuring that everything works as expected from the user's perspective.\
2. Fast and Reliable : Cypress is built on a new architecture that runs in the same run-loop as the application. This allows us for faster test execution and more reliable tests, as Cypress has direct access to the DOM.

3. Automatic Waiting : Cypress automatically waits for elements to appear and for commands and assertions to complete before moving on to the next step. This eliminates the need for explicit waits and reduces flaky tests.
4. Real-Time Reloads : When we save your test file, Cypress automatically reloads and runs the tests again.

Steps to integrate

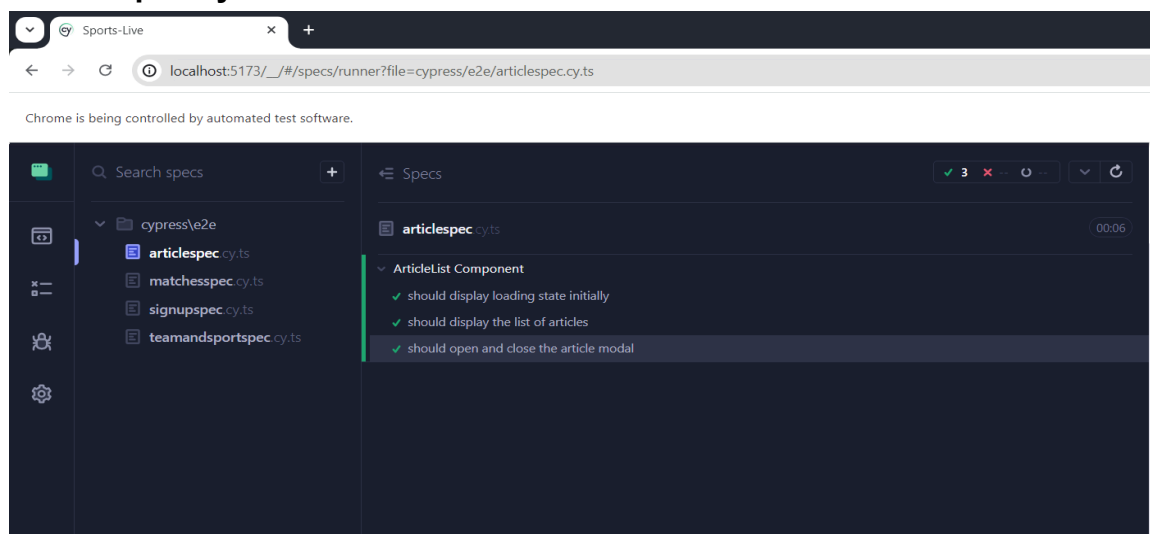
1. Install Cypress in the project
 - `npm install cypress --save-dev`
2. Open Cypress
 - `npx cypress open`
3. Write Tests
 - `describe('My First Test', () => {`
 - `it('Visits the app', () => {`
 - `cy.visit('http://localhost:5173/');`
 - `cy.contains('Sports Live');`
 - `});`
 - `});`

Or whatever tests we have to write, for multiple tests, create multiple files.

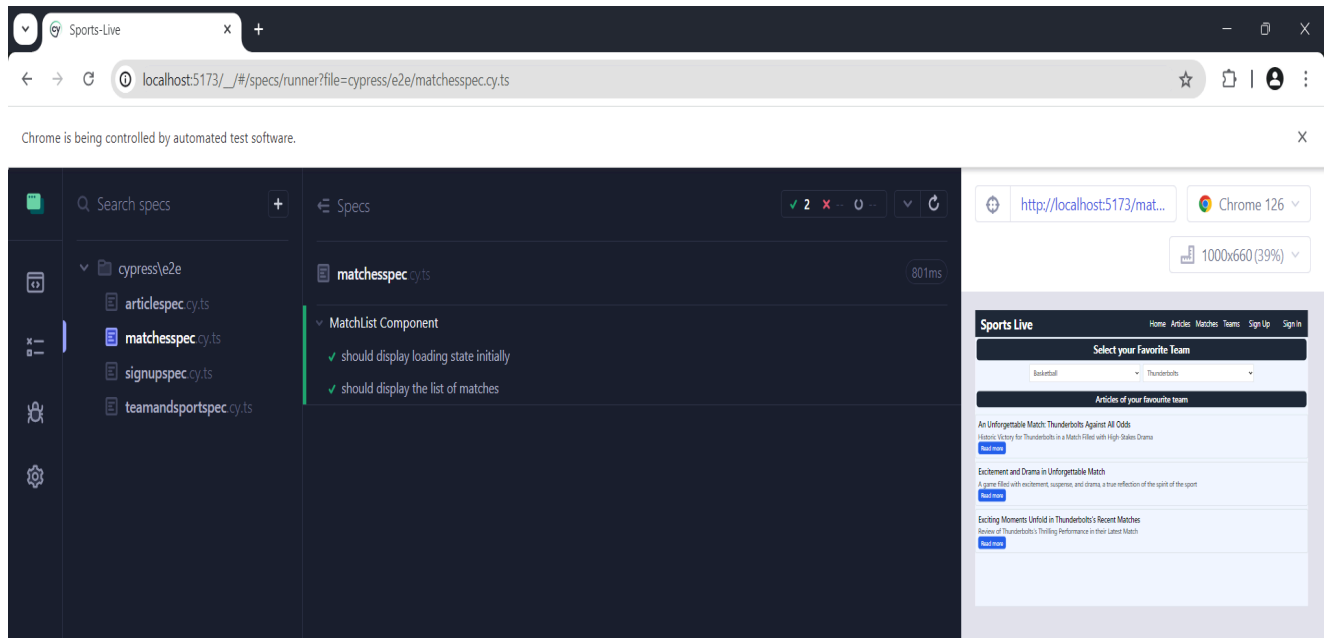
Steps may increase/decrease based on the configuration needed and system used.
(Mine is windows 11 64 bit architecture, node -v20, so it ran smoothly on chrome)

Images of my cypress tests

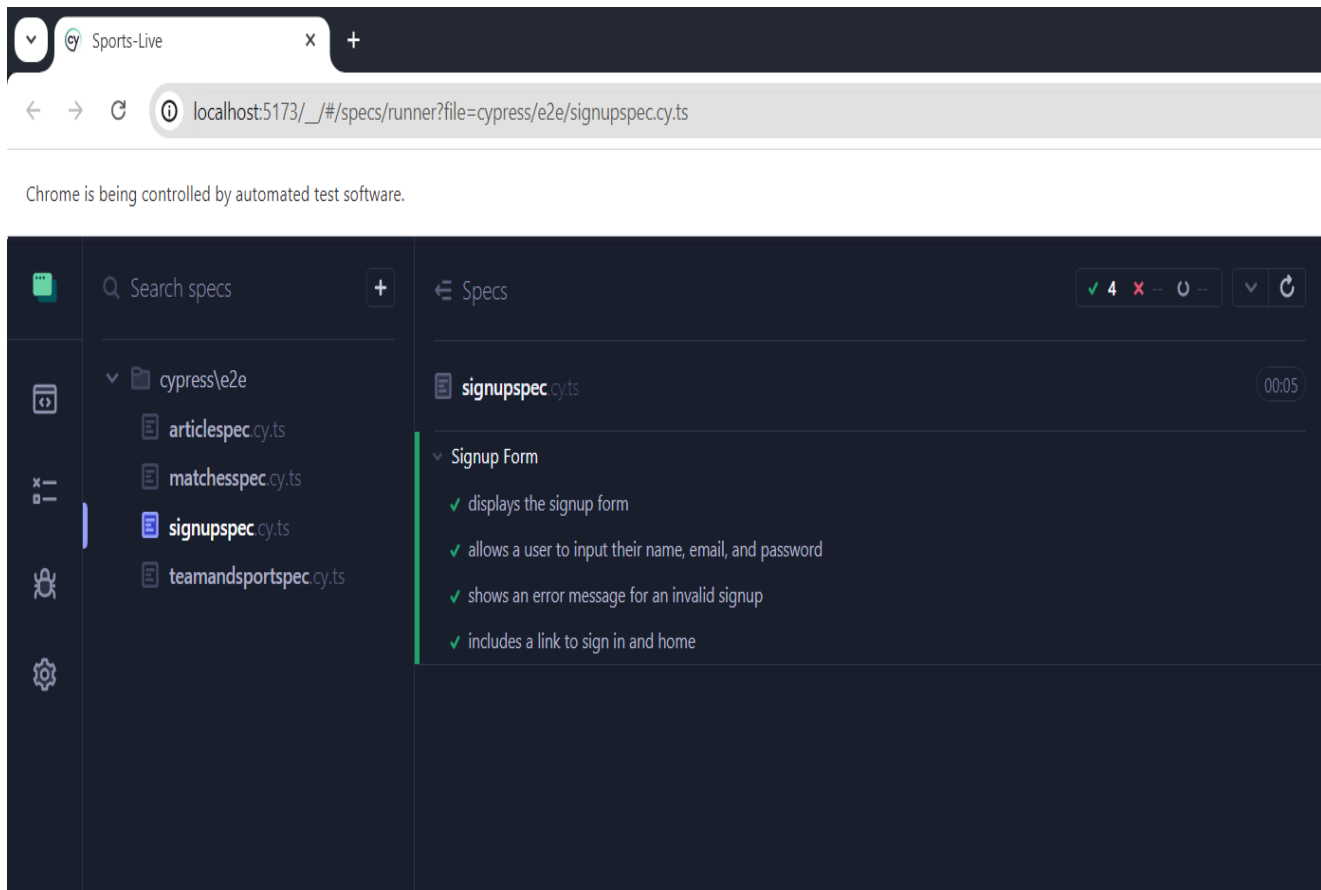
1. articlespec.cy.ts



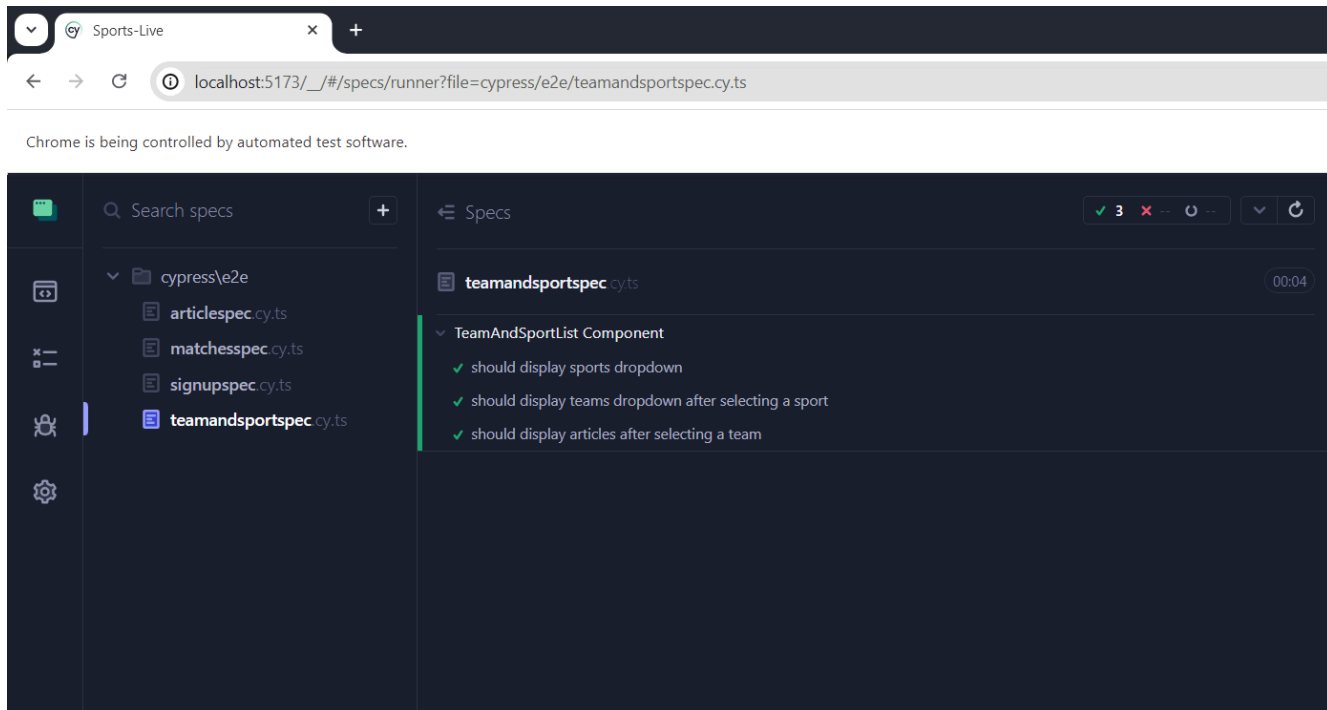
2. matchesspec.cy.ts



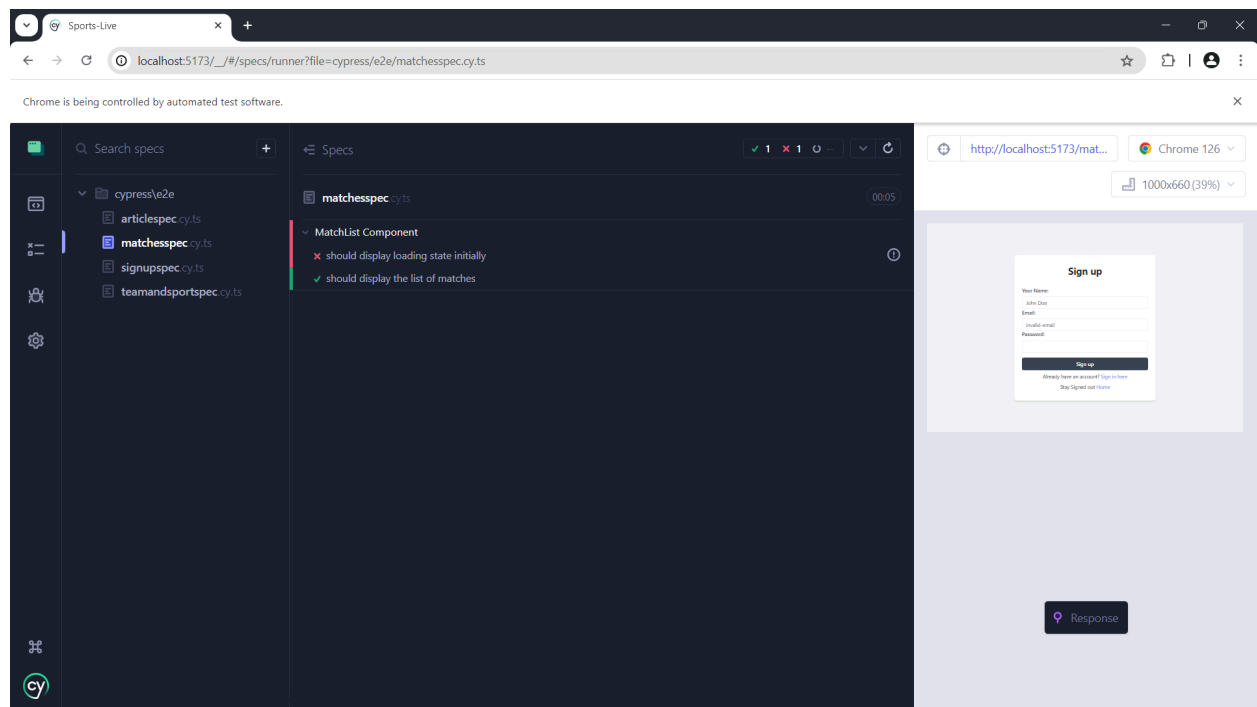
3. signupspec.cy.ts

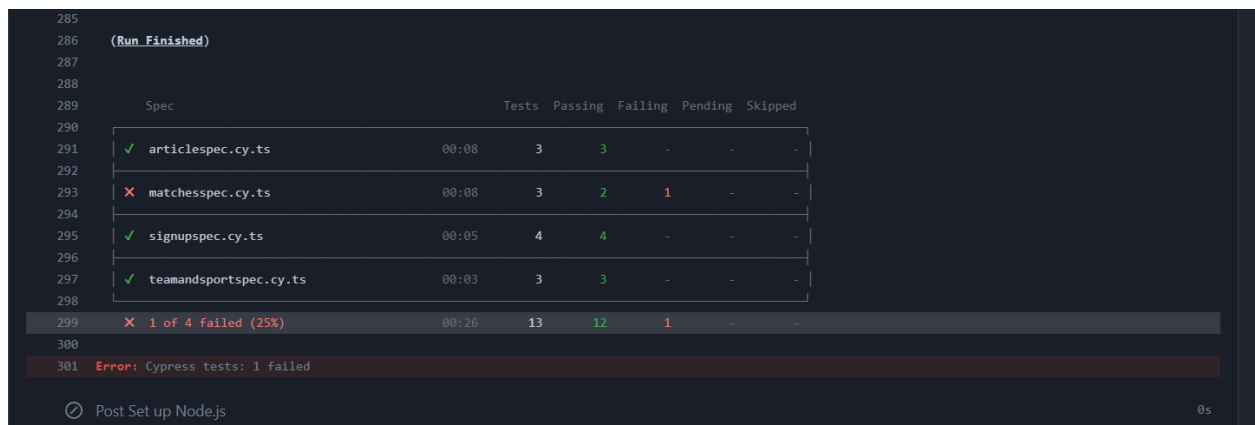
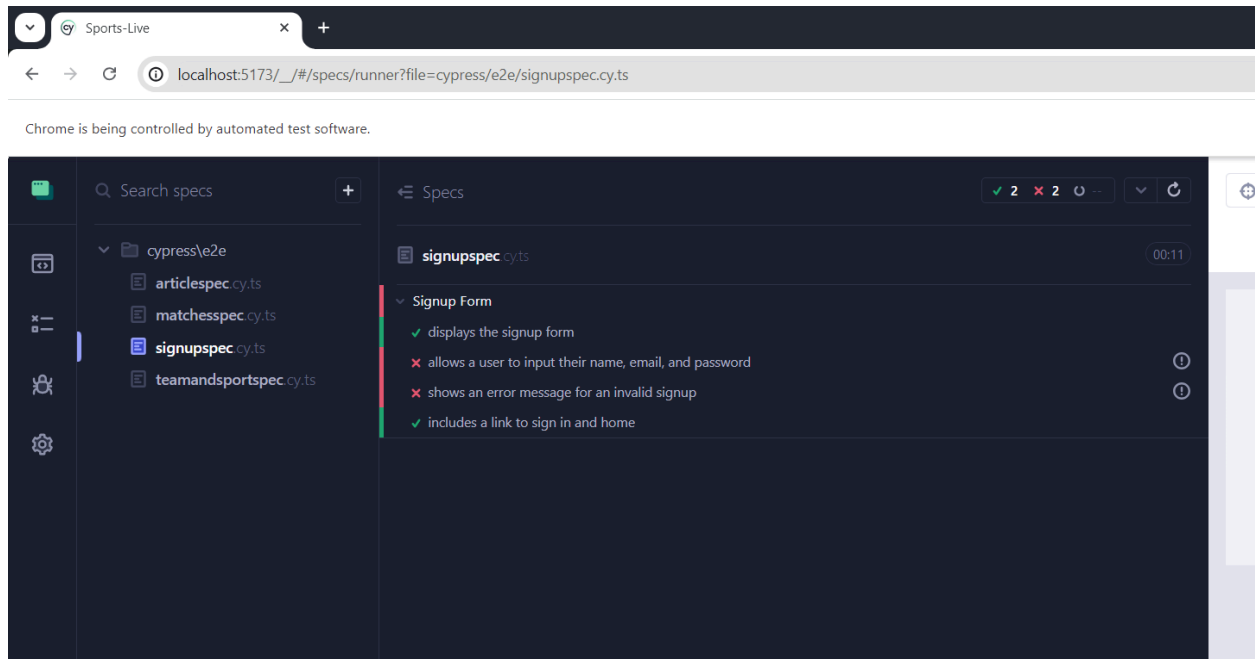


4. teamandsportspec.cy.ts



→ This is how things look when they fail





Github actions

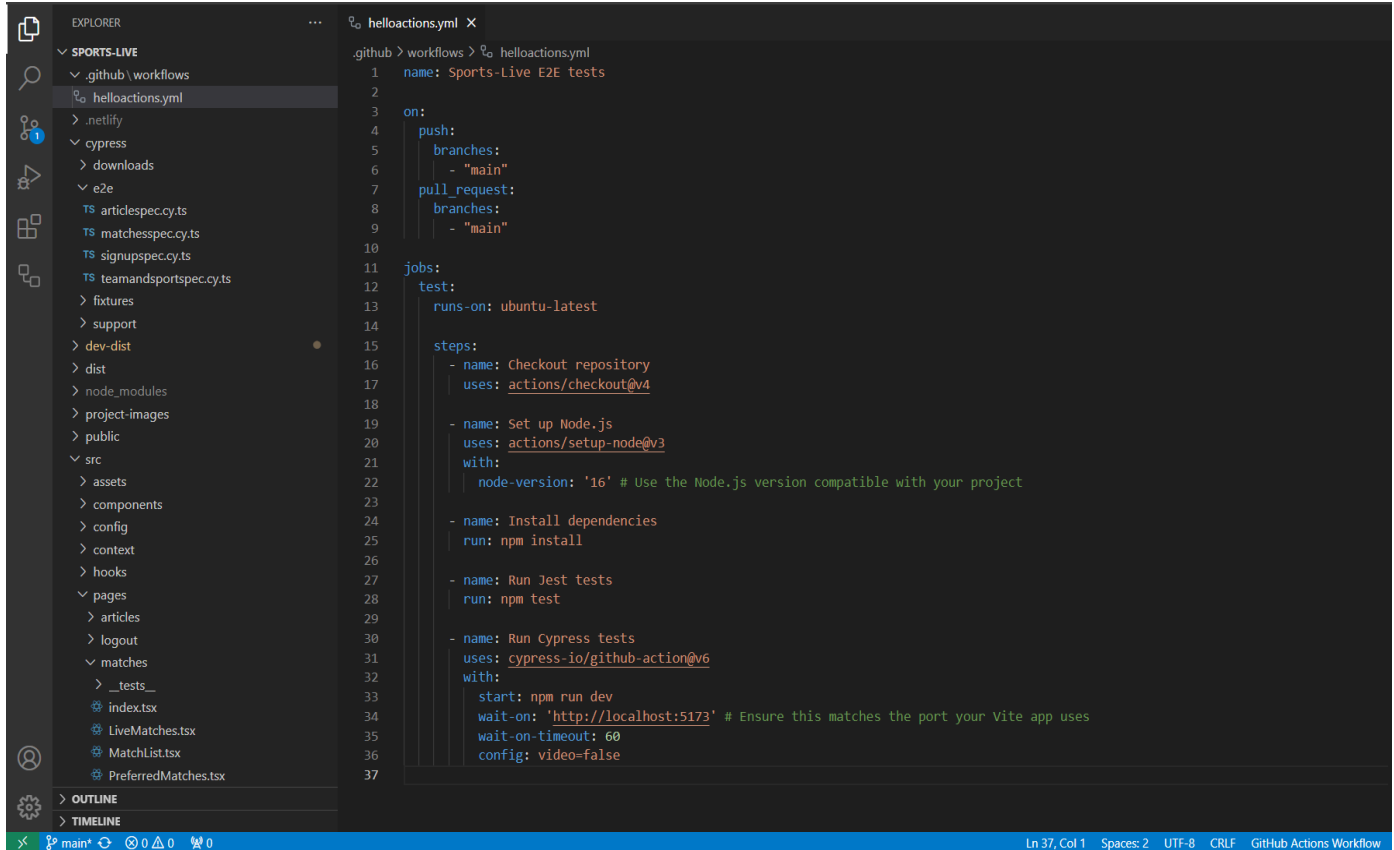
➡ GitHub Actions is a powerful automation platform that integrates with GitHub repositories to automate workflows, including CI/CD (Continuous Integration and Continuous Deployment) processes.

We can automate testing for CI/CD using github actions.

We have to create a folder named github/workflows and a file *filename.yml*. *.yml* is important as it shows that it is a github workflow file

It is easy to use github actions and it runs parallel when code is pushed and pulled automatically. And it is free and is used by many startups and companies.

.yaml file of my project



```
.github > workflows > helloactions.yaml
1  name: Sports-Live E2E tests
2
3  on:
4    push:
5      branches:
6        - "main"
7    pull_request:
8      branches:
9        - "main"
10
11 jobs:
12   test:
13     runs-on: ubuntu-latest
14
15     steps:
16       - name: Checkout repository
17         uses: actions/checkout@v4
18
19       - name: Set up Node.js
20         uses: actions/setup-node@v3
21         with:
22           node-version: '16' # Use the Node.js version compatible with your project
23
24       - name: Install dependencies
25         run: npm install
26
27       - name: Run Jest tests
28         run: npm test
29
30       - name: Run Cypress tests
31         uses: cypress-io/github-action@v6
32         with:
33           start: npm run dev
34           wait-on: 'http://localhost:5173' # Ensure this matches the port your Vite app uses
35           wait-on-timeout: 60
36           config: video=false
37
```

To make tests run in the cypress and in github actions, we must keep our app running on the localhost with the latest code.

If tests fail, or cypress tests are failing, we simply need to retry by restarting the app, or browser or cypress window, as it may encounter some problems due to half loading of the app.

Web Development 4: x | How to write end-to- x | What is Code Covera x | Code Coverage: Ever x | W6401 L4 - Google x | Cypress Tests For Ap x | SwanandBhuskute/S Sports-Live

github.com/SwanandBhuskute/Sports-Live

SwanandBhuskute / Sports-Live

Code Issues Pull requests Actions Projects Security 16 Insights Settings

Sports-Live Public

github actions try 4 +0 -8 16 hours ago

main 1 Branch SwanandBhuskute committed 16 hours ago

Add file Code

About No description, website, or topics provided.

Readme Activity 0 stars 1 watching 0 forks

Releases No releases published Create a new release

Packages No packages published Publish your first package

Languages JavaScript 64.0% TypeScript 34.9% Other 1.1%

File	Commit	Time
.github/workflows	github actions try 2	17 hours ago
cypress	github actions try 4	16 hours ago
dev-dist	github actions try 3	16 hours ago
dist	github actions try 2	17 hours ago
project-images	Project images added	4 months ago
public	PWA featured	4 months ago
src	github actions	18 hours ago
.env	Signin working, Signup not	4 months ago
.eslintrc.cjs	All done, Parser, ts-Lint added	4 months ago
.gitignore	/ and /signup error try2	4 months ago
README.md	Update README.md	4 months ago

https://github.com/SwanandBhuskute/Sports-Live/commit/b46044ade632195e8a76c7e1902a268fd162e723b3ions

Web Development 4: x | How to write end-to- x | What is Code Covera x | Code Coverage: Ever x | W6401 L4 - Google x | Cypress Tests For Ap x | Workflow runs - Swan x

github.com/SwanandBhuskute/Sports-Live/actions

SwanandBhuskute / Sports-Live

Code Issues Pull requests Actions Projects Security 16 Insights Settings

Actions New workflow

All workflows

Sports-Live E2E tests

Management Caches Attestations Runners

All workflows Showing runs from all workflows

5 workflow runs

Workflow	Status	Branch	Actor	Event	Status	Branch	Actor
github actions try 4	Success	main	SwanandBhuskute	Sports-Live E2E tests #5: Commit b460444 pushed by SwanandBhuskute	16 hours ago	1m 57s	...
github actions try 3	Failure	main	SwanandBhuskute	Sports-Live E2E tests #4: Commit 6b11cd7 pushed by SwanandBhuskute	16 hours ago	1m 43s	...
github actions try 3	Failure	main	SwanandBhuskute	Sports-Live E2E tests #3: Commit cd5973a pushed by SwanandBhuskute	16 hours ago	1m 45s	...
github actions try 2	Failure	main	SwanandBhuskute	Sports-Live E2E tests #2: Commit fdbb928 pushed by SwanandBhuskute	17 hours ago	4m 29s	...
github actions	Failure	main	SwanandBhuskute	Sports-Live E2E tests #1: Commit 6b56354 pushed by SwanandBhuskute	18 hours ago	49s	...

github.com/SwanandBhuskute/Sports-Live/actions/runs/9858812808

< Code Issues Pull requests **Actions** Projects Security 16 Insights Settings

← Sports-Live E2E tests

✓ **github actions try 4 #5** Re-run all jobs ...

Summary

Jobs

✓ test

Run details

Usage

Workflow file

Triggered via push 16 hours ago

SwanandBhuskute pushed → b460444 main

Status **Success** Total duration **1m 57s** Artifacts —

helloactions.yml

on: push

✓ test 1m 47s

test summary

Cypress Results

Result	Passed ✓	Failed ✗	Pending ⚠	Skipped ⏸	Duration ⌚
Passing ✓	12	0	0	0	20.683s

Job summary generated at run-time

github.com/SwanandBhuskute/Sports-Live/actions/runs/9858812808/job/27221051423

SwanandBhuskute / Sports-Live

Type / to search

< Code Issues Pull requests **Actions** Projects Security 16 Insights Settings

← Sports-Live E2E tests

✓ **github actions try 4 #5** Re-run all jobs ...

Summary

Jobs

✓ **test**

Run details

Usage

Workflow file

test

succeeded 16 hours ago in 1m 47s

Search logs

- Set up job 2s
- Checkout repository 1s
- Set up Node.js 4s
- Install dependencies 23s
- Run Jest tests 8s
- Run Cypress tests 1m 2s
- Post Set up Node.js 0s
- Post Checkout repository 0s
- Complete job 0s

test

succeeded 16 hours ago in 1m 47s

Search logs



Run Jest tests

8s

```
47
48 PASS src/pages/articles/__tests__/articles.tsx (7.434 s)
49   • Console
50
51   console.error
52     Warning: A suspended resource finished loading inside a test, but the event was not wrapped in act(...).
53
54     When testing, code that resolves suspended data should be wrapped into act(...):
55
56       act(() => {
57         /* finish loading suspended data */
58       });
59       /* assert on the output */
60
61     This ensures that you're testing the behavior the user would see in the browser. Learn more at https://reactjs.org/link/wrap-tests-with-act
62
63     at printWarning (node_modules/react-dom/cjs/react-dom.development.js:86:30)
64     at error (node_modules/react-dom/cjs/react-dom.development.js:60:7)
65     at warnIfSuspenseResolutionNotWrappedWithActDEV (node_modules/react-dom/cjs/react-dom.development.js:27604:7)
66     at pingSuspendedRoot (node_modules/react-dom/cjs/react-dom.development.js:27189:3)
67
68
69 Test Suites: 3 passed, 3 total
70 Tests:       3 passed, 3 total
71 Snapshots:  0 total
72 Time:        8.164 s
73 Ran all test suites.
```

Run Cypress tests

1m 2s

Post Set up Node.js

0s

Post Checkout repository

0s

Run Cypress tests

```
165 ✓ should display the list of articles (5594ms)
166 ✓ should open and close the article modal (2994ms)
167
168
169 3 passing (8s)
170
171
```

(Results)

```
174
175 | Tests:      3
176 | Passing:    3
177 | Failing:    0
178 | Pending:    0
179 | Skipped:    0
180 | Screenshots: 0
181 | Video:      false
182 | Duration:   8 seconds
183 | Spec Ran:   articlespec.cy.ts
184
```

Run Cypress tests1m 2s

185

186

187

188

189Running: matchesspec.cy.ts(2 of 4)

190

191

192MatchList Component

193✓ should display loading state initially (298ms)

194✓ should display the list of matches (1506ms)

195

196

1972 passing (3s)

198

199

200(Results)

201

202

203| Tests: 2

204| Passing: 2

205| Failing: 0

206| Pending: 0

207| Skipped: 0

208| Screenshots: 0

209| Video: false

210| Duration: 3 seconds

211| Spec Ran: matchesspec.cy.ts

212

213

214

Run Cypress tests1m 2s

210| Duration: 3 seconds

211| Spec Ran: matchesspec.cy.ts

212

213

214

215

216

217Running: signupspec.cy.ts(3 of 4)

218

219

220Signup Form

221✓ displays the signup form (334ms)

222✓ allows a user to input their name, email, and password (982ms)

223✓ shows an error message for an invalid signup (2740ms)

224✓ includes a link to sign in and home (138ms)

225

226

2274 passing (6s)

228

229

230(Results)

231

232

233| Tests: 4

234| Passing: 4

235| Failing: 0

236| Pending: 0

237| Skipped: 0

238| Screenshots: 0

239| Video: false

240| Duration: 5 seconds

241| Spec Ran: signupspec.cy.ts

242

```

245
246
247   Running:   teamandsportspec.cy.ts                                (4 of 4)
248
249
250   TeamAndSportList Component
251     ✓ should display sports dropdown (366ms)
252     ✓ should display teams dropdown after selecting a sport (1157ms)
253     ✓ should display articles after selecting a team (737ms)
254
255
256   3 passing (4s)
257
258
259   (Results)
260
261   | Tests:      3
262   | Passing:    3
263   | Failing:    0
264   | Pending:    0
265   | Skipped:    0
266   | Screenshots: 0
267   | Video:      false
268   | Duration:   3 seconds
269   | Spec Ran:   teamandsportspec.cy.ts
270
271
272
273
274   =====
275
276   (Run Finished)
277

```

```

273
274   =====
275
276   (Run Finished)
277
278
279   Spec                                Tests  Passing  Failing  Pending  Skipped
280
281   ✓ articlespec.cy.ts                 00:08    3         3         -         -         -
282
283   ✓ matchesspec.cy.ts                 00:03    2         2         -         -         -
284
285   ✓ signupspec.cy.ts                  00:05    4         4         -         -         -
286
287   ✓ teamandsportspec.cy.ts            00:03    3         3         -         -         -
288
289   ✓ All specs passed!                  00:20   12        12         -         -         -
290

```

> ✓ Post Set up Node.js0s

> ✓ Post Checkout repository0s

> ✓ Complete job0s