Playwright + Cucumber Setup: Parallel Execution & Rerun Failed Tests

This guide walks you through how to:

- Run your Cucumber tests in **parallel**
- Automatically **rerun only the failed tests**
- Set up necessary configuration using `cucumber.js`
- Use scripts in `package.json` for easy execution

Step 1: Configure Parallel Execution

To speed up your test runs, you can run multiple browser instances in parallel using the `parallel` option in `cucumber.js`.

`cucumber.js` file configuration

Create or update a file named `cucumber.js` in your project root:

```
module.exports = {
  default: {
    require: [
      'src/test/steps/*.ts',           // Step definitions
      'src/test/hooks/hooks.ts'        // Hooks (e.g. Before, After)
    ],
    requireModule: [
      'ts-node/register'               // To run TypeScript files
    ],
    format: [
      'progress-bar',                                    // Console output
      'html:test-results/cucumber-report.html',          // HTML report
      'json:test-results/cucumber-report.json',          // JSON report
      'junit:test-results/cucumber-report.xml'           // JUnit report
(optional)
    ],
    parallel: 2  // Number of parallel browser instances
  }
};
```

Why Parallel?

Running tests in parallel helps reduce execution time significantly, especially when your test suite grows large.

Step 2: Setup Rerun for Failed Scenarios

After executing your tests, if any scenarios fail, Cucumber will generate a file named `@rerun.txt`. This file lists all failed scenarios with their feature file paths and line numbers.

We can use this file to re-run **only** the failed tests.

Add a `rerun` profile in `cucumber.js`

Extend your `cucumber.js` file with a `rerun` profile:

```
module.exports = {
  default: {
    // default configuration as above
  },
  rerun: {
    require: [
      'src/test/steps/*.ts',
      'src/test/hooks/hooks.ts'
    ],
    requireModule: ['ts-node/register'],
    formatOptions: {
      snippetInterface: 'async-await'
    },
    publishQuiet: true,
    dryRun: false,
    format: [
      'progress-bar',
      'html:test-results/cucumber-report.html',
      'json:test-results/cucumber-report.json',
      'junit:test-results/cucumber-report.xml',
      'rerun:@rerun.txt'   // Reference to the rerun file
    ],
    parallel: 2  // Run rerun tests in parallel as well
  }
};
```

You don't need to specify feature file paths here because `@rerun.txt` will handle that automatically.

Step 3: Add Scripts to `package.json`

These scripts help simplify running full tests and rerunning failed ones.

```
"scripts": {
  "test": "cucumber-js test || exit 0",
  "posttest": "npx ts-node src/helper/report.ts",
  "test:failed": "cucumber-js -p rerun @rerun.txt"
}
```

- `"test"`: Runs all scenarios, even if some fail.
- `"posttest"`: Generates your report after test execution.
- `"test:failed"`: Runs only the scenarios listed in `@rerun.txt`.

Step 4: How to Use

Run all tests

```
npm run test
```

This will:

- Run all scenarios in parallel
- Generate HTML, JSON, and JUnit reports
- Create `@rerun.txt` if any tests fail

Rerun only failed scenarios

```
npm run test:failed
```

This will:

- Read failed scenarios from `@rerun.txt`
- Re-execute only the failed tests
- Clear the `@rerun.txt` file if all rerun tests pass

Step 5: Clean-up Behavior

After running `npm run test:failed`, if all previously failed scenarios now pass, then:

- The `@rerun.txt` file is automatically cleared
- This confirms that all failures are fixed
- If any failures persist, the file will retain those scenarios

---

Summary Table

Feature	How to Configure
Parallel Execution	`"parallel": 2` in `cucumber.js`
Failed Tests Rerun	Add `rerun` profile with `rerun:@rerun.txt` in `cucumber.js`
Test Scripts	Add `test`, `posttest`, and `test:failed` in `package.json`
Auto Clean-up of Failures	`@rerun.txt` is cleared if rerun tests all pass

---