77. how to fix flaky tests with test retry option in playwright config file

how to retry the tests which are failed ?

sometimes when you run some hundreds of test cases, you might see 1o tests failed,

But when you retry them, most of them will pass. On first execution, there could be some test of flakiness. It could be network unstable or the way you have written automation script without proper synchronization.

So that rerunning process can we can make it automatic.

So if you tell playwright to rerun the failed test cases, it will simply collect failed tests and it will rerun them. That's automatically done by playwright. If you simply set one property called (retries : ) set this in config file object level not in use.

So once you get execution results of your overall project, then this will retries that values and it will retry. So this should be declared in global config level.

So let's say you retries : 1, So that means your failed test will retry for one more time.

This is in config file.

testDir: './tests',

  retries : 1,

    timeout : 30 \* 1000,

  expect:  {   // assertions

    timeout : 5000

  },

78. Understand how playwright run tests in serial & parallel mode and update settings

So by default tests run in serial mode for the test, presenting the same file.

So if you execute one single spec file and there could be multiple tests in that, so one by one each test will execute.

But if you run this whole test folder anytime, so that means it looks like there are around six test here.All these tests will run in parallel.

So that means test files will trigger parallel, but individual tests which are present in that file

will run in sequence.

npx playwright test ( it will default Run all the test cases present in this test folder )

So each test file it will assign to one worker. So here, worker means test execution process.

One environment where you are tests will get executed.

you did not add any code to run tests in parallel. So by default, the playwright supports running tests in parallel mode.

Let's say if you put workers equal to one, ( workers : 1, )then that means you are saying that at a time only one worker should run. That means only one test will run.

So that means you are disabling parallel mechanism. If you said to one worker.

So that's how you can control how many tests you want to run in parallel.

79. How to run tests parallely from the same file by extending test option

if you want to run the tests in the same file in parallel mode.

there is another mode called serial mode.

that is used when you are tests are inter-dependent on each other.

So let's say you are working on orders confirmation page. This test is working.

This test is making you to bring order confirmation page level.

So if it don't go to that page, what is the point validating in the confirmation --?so this will obviously will fail.So instead of making false fails, you can skip the remaining executions.

so now when you put serial mode, now lets the second test fail, then it won't execute to test

it will skip the third one.

test.describe.configure({mode:'prallel'});

npx playwright test tests/MoreValidations.spec.js --config playwright.config1.js --project=chrome

Running 3 tests using 1 worker

test.describe.configure({mode:'serial'});

80. Reason for test failures - Race condition - Fix them

So let's say one test placed an order at the same time, other tests reaches this page and here it will be blank because we are using same login and we are acting on the same product and multiple times when we are doing parallel, that could be race condition.

So what happens is when two tests exactly behaving the same way prallely.

Then you know, there could be race condition here.

So that's how in real time we generally don't repeat same test again and again.

So either it will be different credentials or different product name or a different logic at least.

So we are exactly doing same thing that we are ending up in the race condition

test.skip('@Web Client App Login', async ({page})=>    {

81. How to tag tests and control the execution from the command line parameters

@Web

test('@Web Client App Login', async ({page})=>    {

only triiger the tests with are tagged that with @Web ( npx playwright test –grep @web ) where ever this @web tag name is present in the tests name one those tests should be run.

hyphen hyphen, grep . grep stands for regular expression that it

will smartly check for that keyword(@web)

So it will scan all the test titles present in this entire test folder and it will see wherever you have a tag name as web, then those tests only will be picked and it will run.