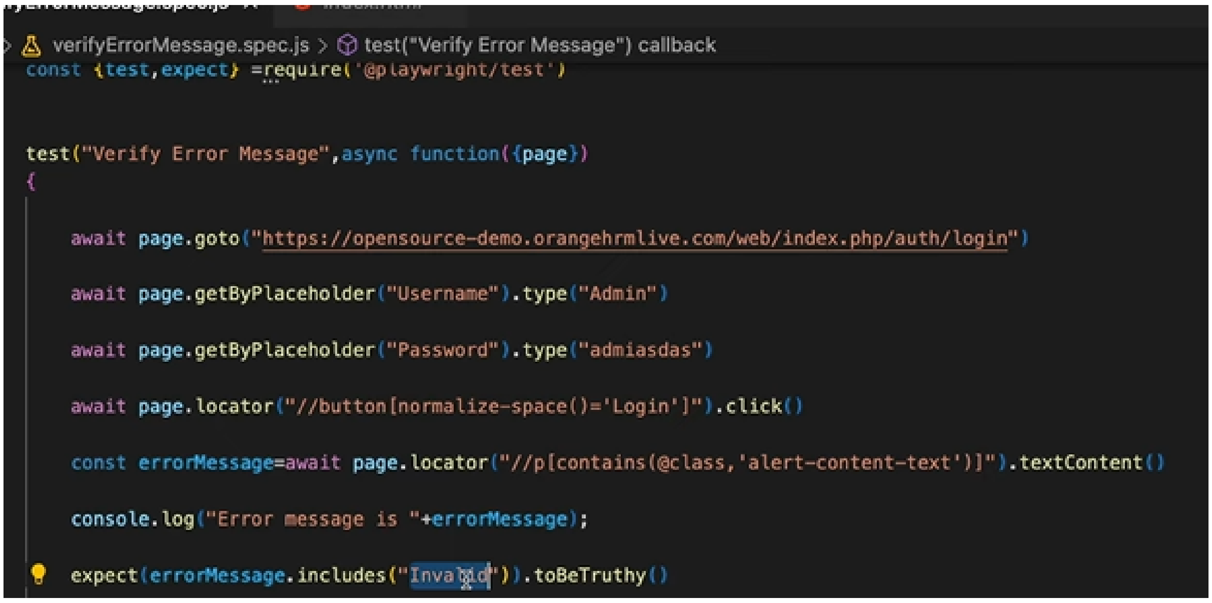
5. How To Get Text In Playwright

If we use xpath use locator

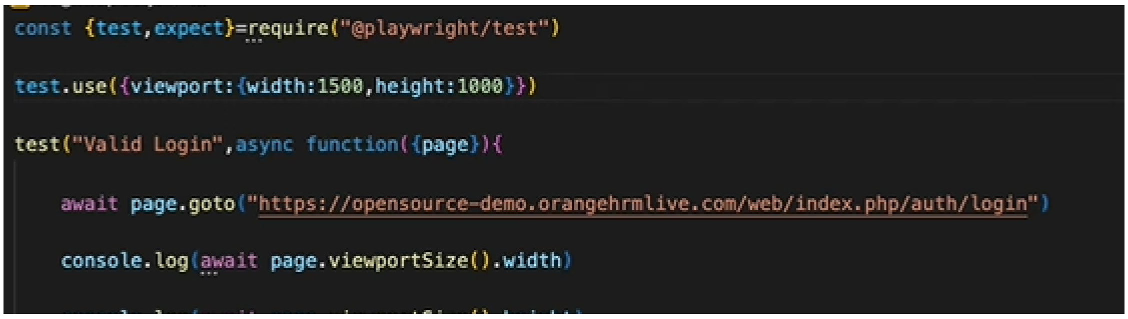


verifyErrorMessage.spec.js > test("Verify Error Message") callback

```
const {test,expect} =require('@playwright/test')


test("Verify Error Message",async function({page})
{

    await page.goto("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login")

    await page.getByPlaceholder("Username").type("Admin")

    await page.getByPlaceholder("Password").type("admiasdas")

    await page.locator("//button[normalize-space()='Login']").click()

    const errorMessage=await page.locator("//p[contains(@class,'alert-content-text')]").textContent()

    console.log("Error message is "+errorMessage);

    expect(errorMessage.includes("Invalid")).toBeTruthy()
```

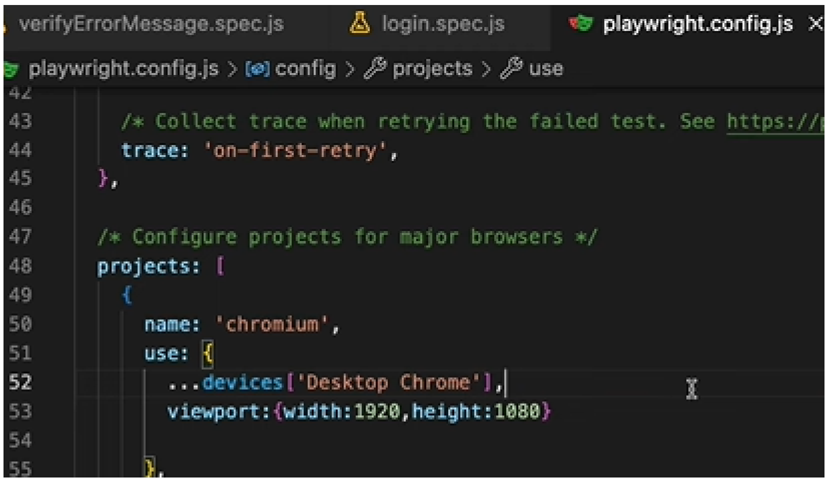
6. How To Maximise Browser Window In playwright

For a particular test



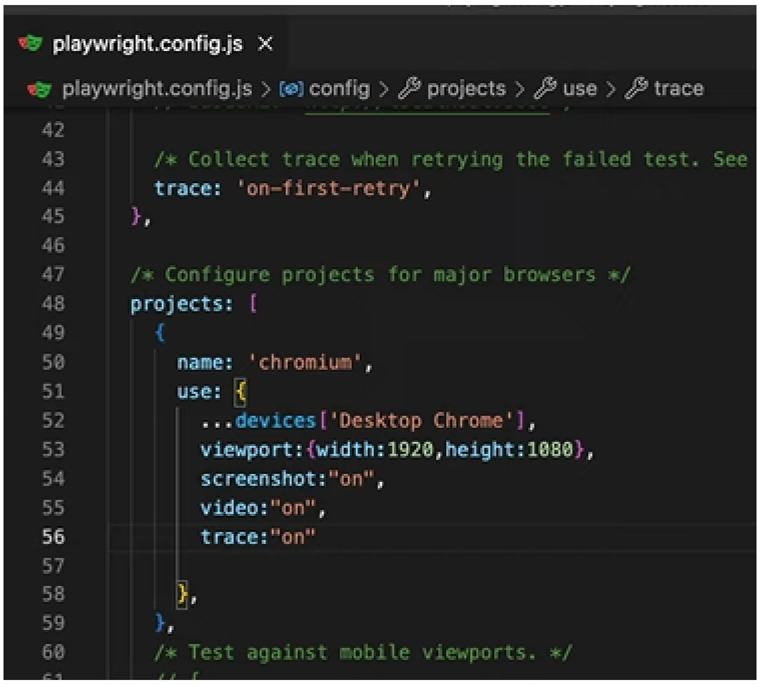
```
const {test,expect}=require("@playwright/test")

test.use({viewport:{width:1500,height:1000}})

test("Valid Login",async function({page}){

    await page.goto("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login")

    console.log(await page.viewportSize().width)
```

For every test:


verifyErrorMessage.spec.js
login.spec.js
playwright.config.js
playwright.config.js > config > projects > use

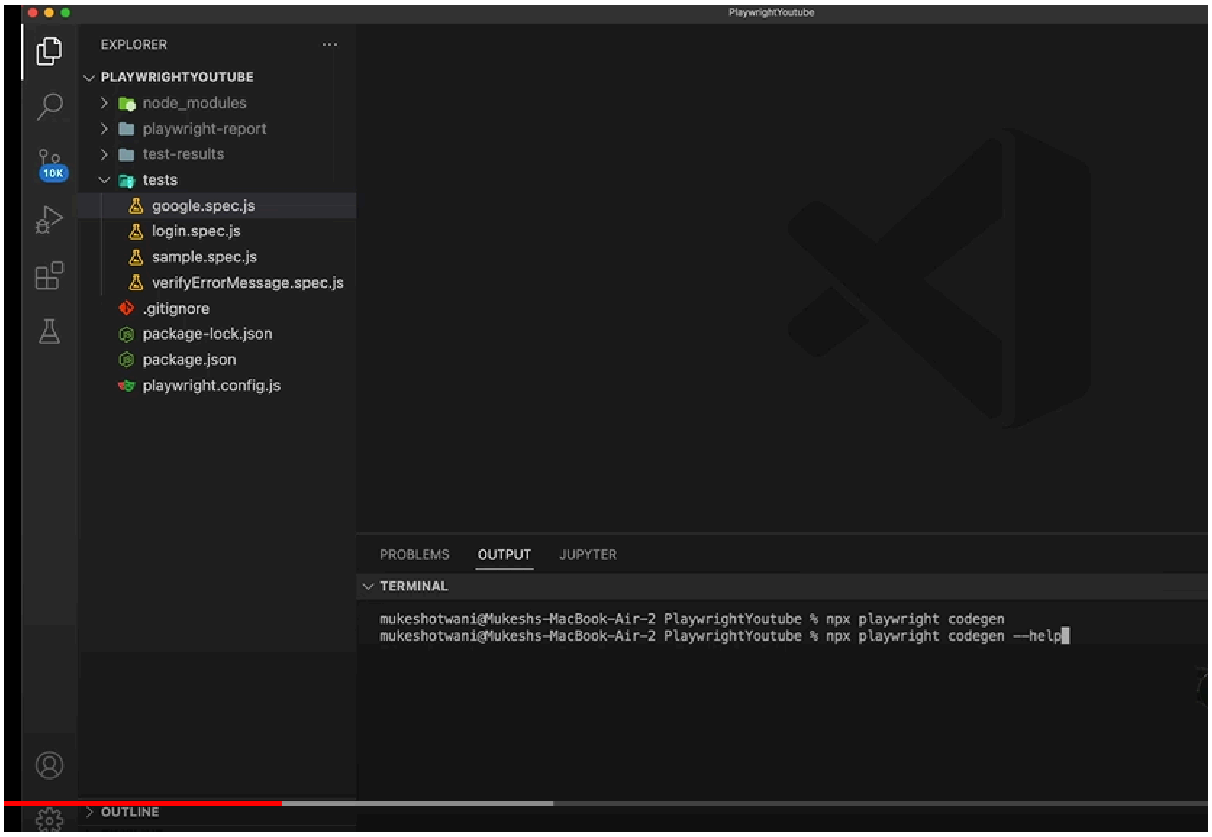
```
      /* Collect trace when retrying the failed test. See https://
      trace: 'on-first-retry',
    },

    /* Configure projects for major browsers */
    projects: [
      {
        name: 'chromium',
        use: {
          ...devices['Desktop Chrome'],
          viewport:{width:1920,height:1080}

        },
```

7. How To Take Screenshots

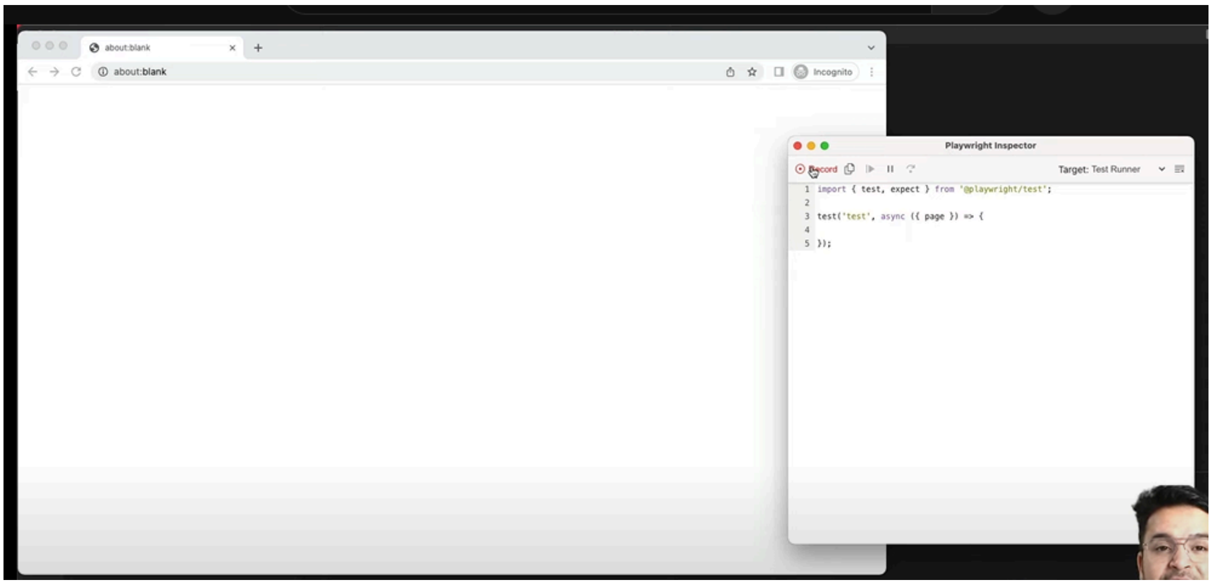


```
      /* Collect trace when retrying the failed test. See
      trace: 'on-first-retry',
    },

    /* Configure projects for major browsers */
    projects: [
      {
        name: 'chromium',
        use: {
          ...devices['Desktop Chrome'],
          viewport:{width:1920,height:1080},
          screenshot:"on",
          video:"on",
          trace:"on"

        },
      },
      /* Test against mobile viewports. */
```

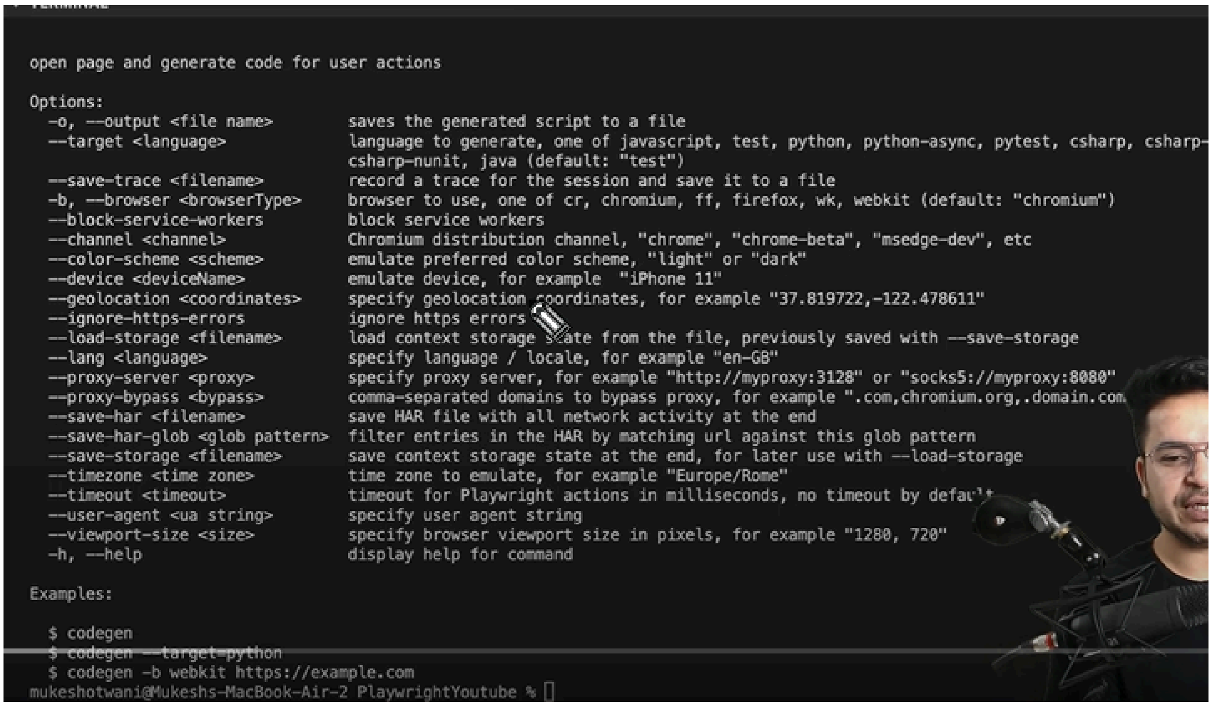
8.How To Record And Play Scripts In Playwright



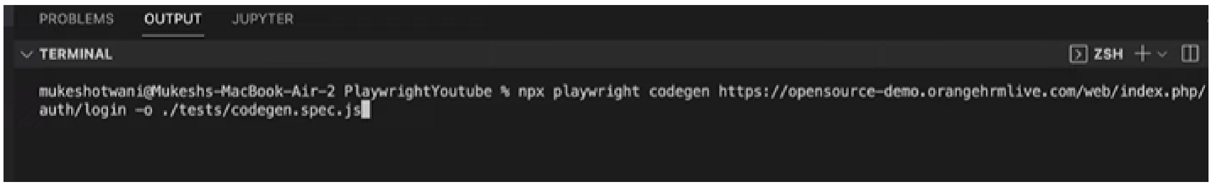
```
mukeshotwani@Mukeshs-MacBook-Air-2 PlaywrightYoutube % npx playwright codegen
mukeshotwani@Mukeshs-MacBook-Air-2 PlaywrightYoutube % npx playwright codegen --help
```



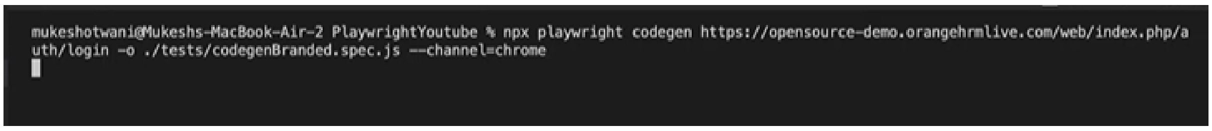
```
import { test, expect } from '@playwright/test';

test('test', async ({ page }) => {

});
```



```
open page and generate code for user actions

Options:
  -o, --output <file name>       saves the generated script to a file
  --target <language>            language to generate, one of javascript, test, python, python-async, pytest, csharp, csharp-
                                 csharp-nunit, java (default: "test")
  --save-trace <filename>        record a trace for the session and save it to a file
  -b, --browser <browserType>    browser to use, one of cr, chromium, ff, firefox, wk, webkit (default: "chromium")
  --block-service-workers        block service workers
  --channel <channel>            Chromium distribution channel, "chrome", "chrome-beta", "msedge-dev", etc
  --color-scheme <scheme>        emulate preferred color scheme, "light" or "dark"
  --device <deviceName>          emulate device, for example  "iPhone 11"
  --geolocation <coordinates>    specify geolocation coordinates, for example "37.819722,-122.478611"
  --ignore-https-errors          ignore https errors
  --load-storage <filename>      load context storage state from the file, previously saved with --save-storage
  --lang <language>              specify language / locale, for example "en-GB"
  --proxy-server <proxy>         specify proxy server, for example "http://myproxy:3128" or "socks5://myproxy:8080"
  --proxy-bypass <bypass>        comma-separated domains to bypass proxy, for example ".com,chromium.org,.domain.com
  --save-har <filename>          save HAR file with all network activity at the end
  --save-har-glob <glob pattern> filter entries in the HAR by matching url against this glob pattern
  --save-storage <filename>      save context storage state at the end, for later use with --load-storage
  --timezone <time zone>         time zone to emulate, for example "Europe/Rome"
  --timeout <timeout>            timeout for Playwright actions in milliseconds, no timeout by default
  --user-agent <ua string>       specify user agent string
  --viewport-size <size>         specify browser viewport size in pixels, for example "1280, 720"
  -h, --help                     display help for command

Examples:

  $ codegen
  $ codegen --target=python
  $ codegen -b webkit https://example.com
mukeshotwani@Mukeshs-MacBook-Air-2 PlaywrightYoutube % 
```

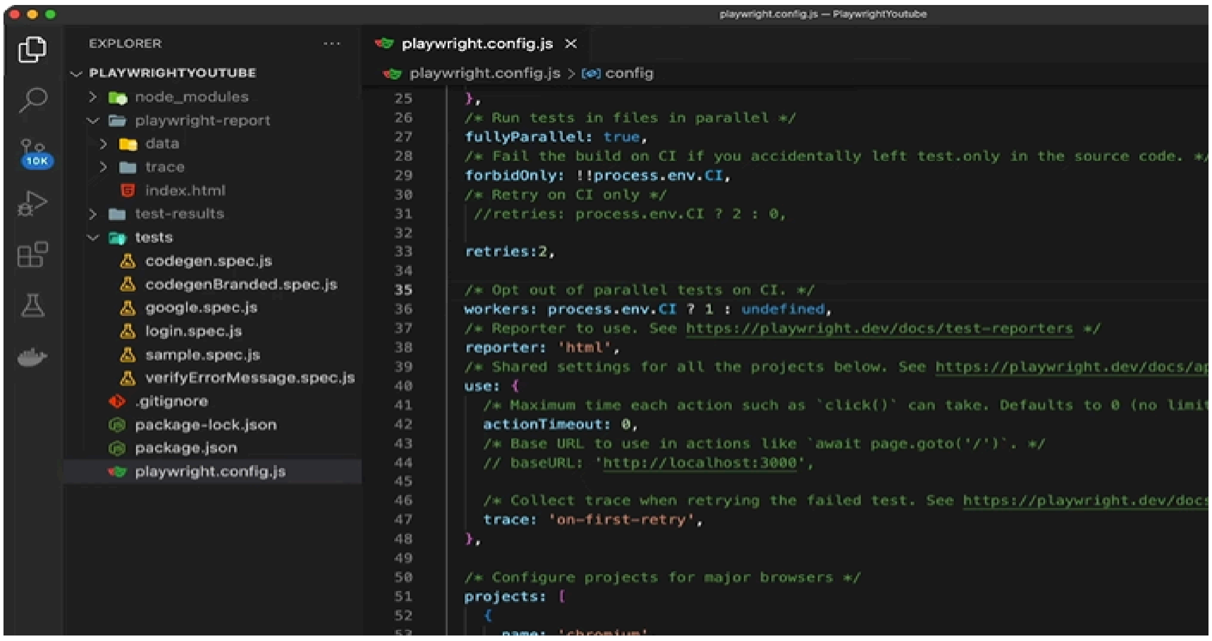


```
mukeshotwani@Mukeshs-MacBook-Air-2 PlaywrightYoutube % npx playwright codegen https://opensource-demo.orangehrmlive.com/web/index.php/
auth/login -o ./tests/codegen.spec.js
```



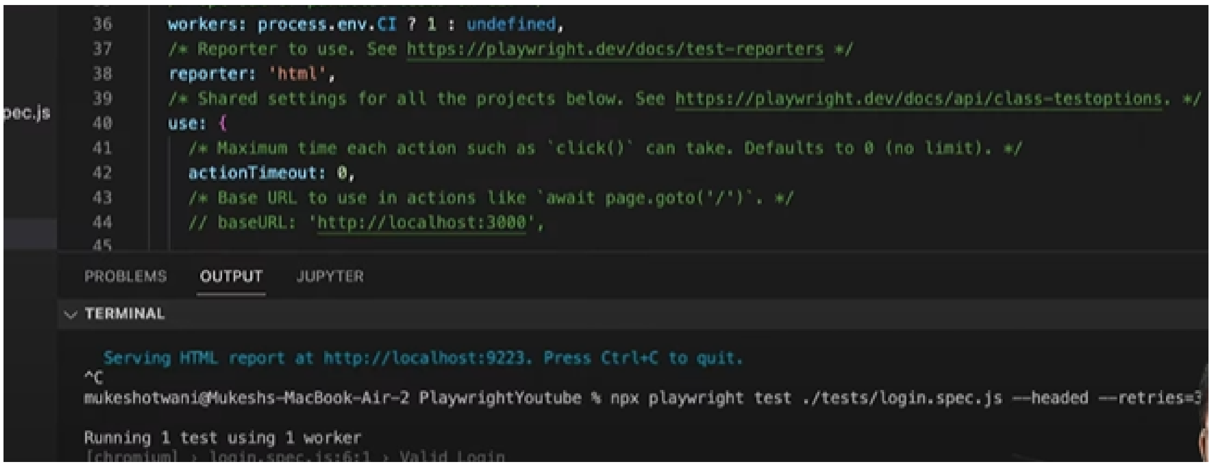
```
mukeshotwani@Mukeshs-MacBook-Air-2 PlaywrightYoutube % npx playwright codegen https://opensource-demo.orangehrmlive.com/web/index.php/a
uth/login -o ./tests/codegenBranded.spec.js --channel=chrome
```

9.How To Retry Failed Test Cases in Playwright



```
  },
  /* Run tests in files in parallel */
  fullyParallel: true,
  /* Fail the build on CI if you accidentally left test.only in the source code. */
  forbidOnly: !!process.env.CI,
  /* Retry on CI only */
   //retries: process.env.CI ? 2 : 0,

  retries:2,

  /* Opt out of parallel tests on CI. */
  workers: process.env.CI ? 1 : undefined,
  /* Reporter to use. See https://playwright.dev/docs/test-reporters */
  reporter: 'html',
  /* Shared settings for all the projects below. See https://playwright.dev/docs/ap
  use: {
    /* Maximum time each action such as `click()` can take. Defaults to 0 (no limit
    actionTimeout: 0,
    /* Base URL to use in actions like `await page.goto('/')`. */
    // baseURL: 'http://localhost:3000',

    /* Collect trace when retrying the failed test. See https://playwright.dev/docs
    trace: 'on-first-retry',
  },

  /* Configure projects for major browsers */
  projects: [
    {
      name: 'chromium'
```

Second approach



```
workers: process.env.CI ? 1 : undefined,
/* Reporter to use. See https://playwright.dev/docs/test-reporters */
reporter: 'html',
/* Shared settings for all the projects below. See https://playwright.dev/docs/api/class-testoptions. */
use: {
  /* Maximum time each action such as `click()` can take. Defaults to 0 (no limit). */
  actionTimeout: 0,
  /* Base URL to use in actions like `await page.goto('/')`. */
  // baseURL: 'http://localhost:3000',
```

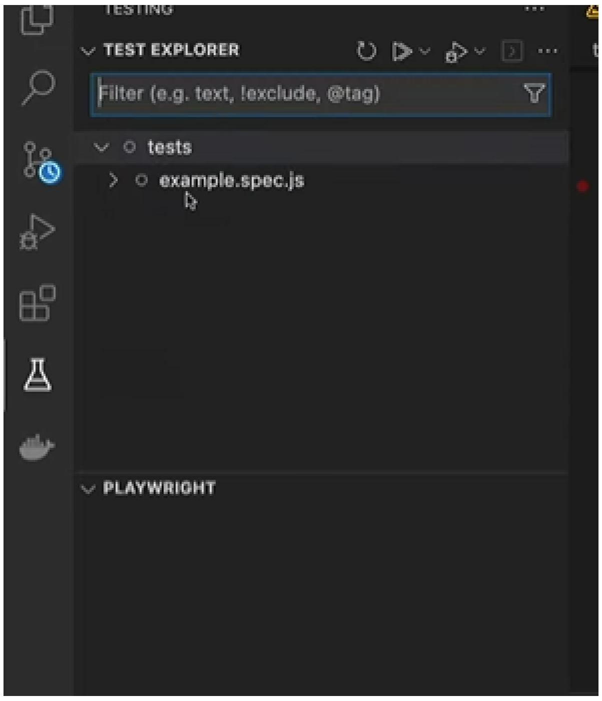
```
  Serving HTML report at http://localhost:9223. Press Ctrl+C to quit.
^C
mukeshotwani@Mukeshs-MacBook-Air-2 PlaywrightYoutube % npx playwright test ./tests/login.spec.js --headed --retries=3

Running 1 test using 1 worker
[chromium] › login.spec.js:6:1 › Valid Login
```

–retries=3

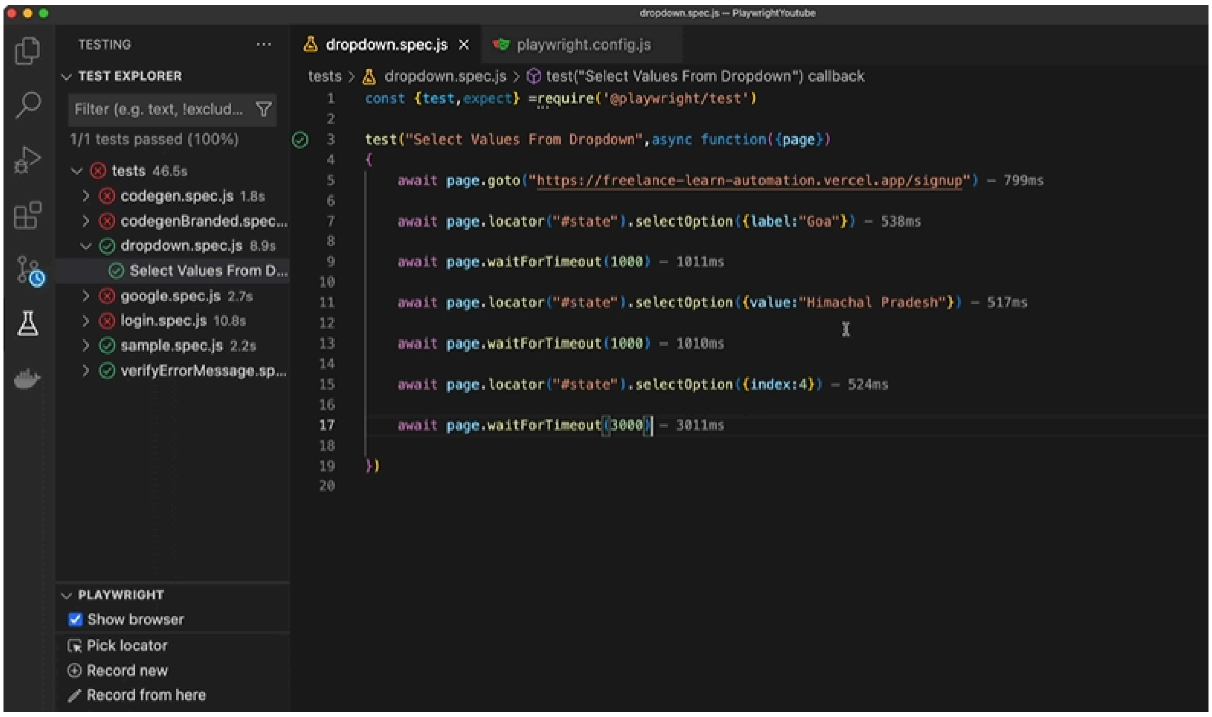
If the test is passed in retry it is called as flaky test

10.Install Playwright In Visual Studio Code

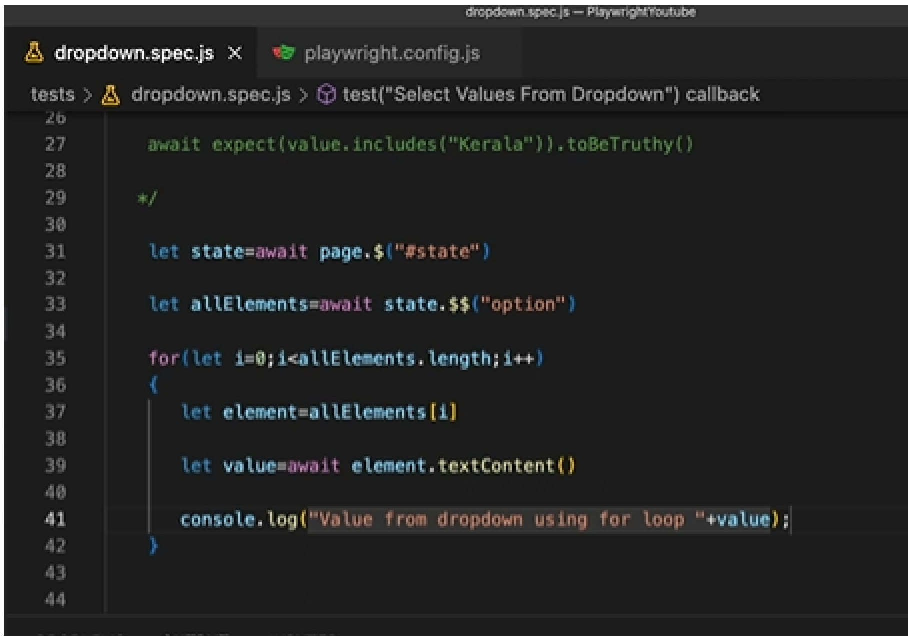
To run tests using testing option



11.Handle Dropdown In Playwright And Verify Dropdown Values



```javascript
const {test,expect} =require('@playwright/test')

test("Select Values From Dropdown",async function({page})
{
    await page.goto("https://freelance-learn-automation.vercel.app/signup") — 799ms

    await page.locator("#state").selectOption({label:"Goa"}) — 538ms

    await page.waitForTimeout(1000) — 1011ms

    await page.locator("#state").selectOption({value:"Himachal Pradesh"}) — 517ms

    await page.waitForTimeout(1000) — 1010ms

    await page.locator("#state").selectOption({index:4}) — 524ms

    await page.waitForTimeout(3000) — 3011ms

})
```

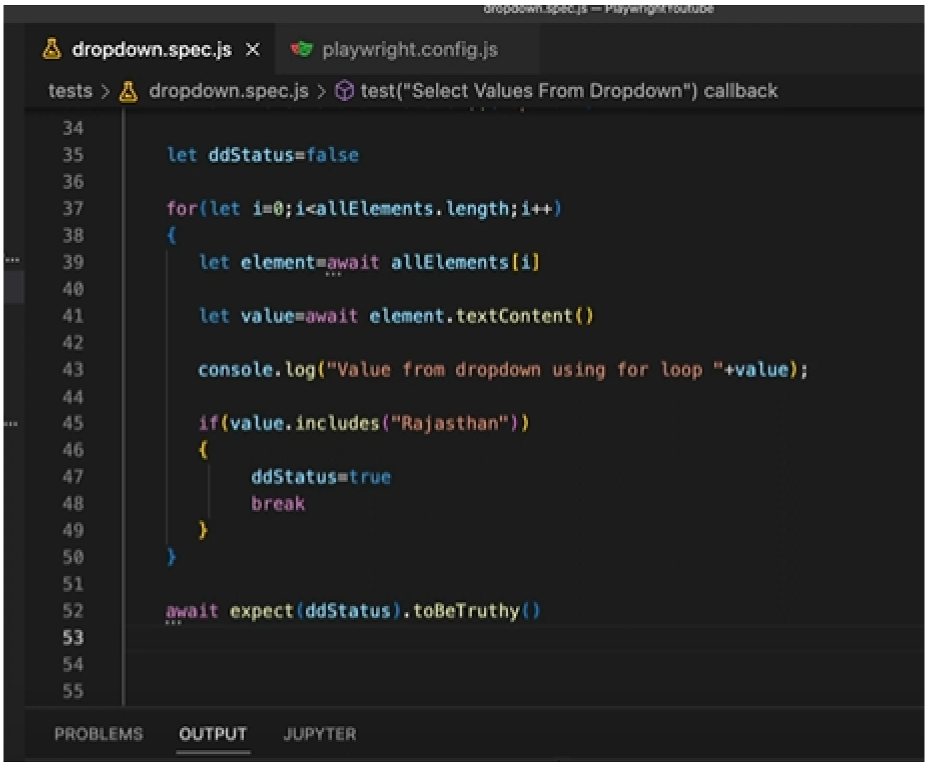


dropdown.spec.js — PlaywrightYoutube

dropdown.spec.js ✕ | playwright.config.js

tests > dropdown.spec.js > test("Select Values From Dropdown") callback

```
     await expect(value.includes("Kerala")).toBeTruthy()

    */

     let state=await page.$("#state")

     let allElements=await state.$$("option")

     for(let i=0;i<allElements.length;i++)
     {
        let element=allElements[i]

        let value=await element.textContent()

        console.log("Value from dropdown using for loop "+value);
     }


```



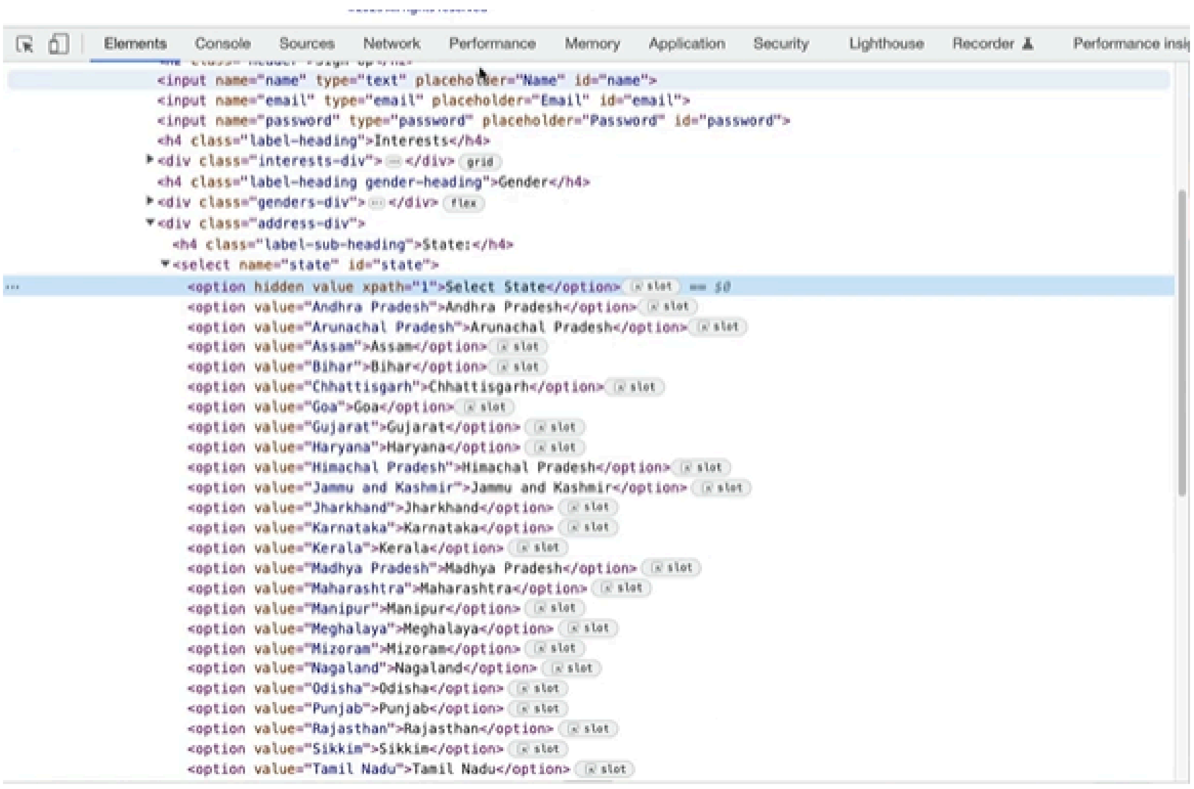
dropdown.spec.js — PlaywrightYoutube

dropdown.spec.js ✕ | playwright.config.js

tests > dropdown.spec.js > test("Select Values From Dropdown") callback

```

     let ddStatus=false

     for(let i=0;i<allElements.length;i++)
     {
        let element=await allElements[i]

        let value=await element.textContent()

        console.log("Value from dropdown using for loop "+value);

        if(value.includes("Rajasthan"))
        {
             ddStatus=true
             break
        }
     }

     await expect(ddStatus).toBeTruthy()



```

PROBLEMS   OUTPUT   JUPYTER



```
<input name="name" type="text" placeholder="Name" id="name">
<input name="email" type="email" placeholder="Email" id="email">
<input name="password" type="password" placeholder="Password" id="password">
<h4 class="label-heading">Interests</h4>
▶<div class="interests-div"> ⋯ </div> grid
<h4 class="label-heading gender-heading">Gender</h4>
▶<div class="genders-div"> ⋯ </div> flex
▼<div class="address-div">
  <h4 class="label-sub-heading">State:</h4>
 ▼<select name="state" id="state">
    <option hidden value xpath="1">Select State</option> slot == $0
    <option value="Andhra Pradesh">Andhra Pradesh</option> slot
    <option value="Arunachal Pradesh">Arunachal Pradesh</option> slot
    <option value="Assam">Assam</option> slot
    <option value="Bihar">Bihar</option> slot
    <option value="Chhattisgarh">Chhattisgarh</option> slot
    <option value="Goa">Goa</option> slot
    <option value="Gujarat">Gujarat</option> slot
    <option value="Haryana">Haryana</option> slot
    <option value="Himachal Pradesh">Himachal Pradesh</option> slot
    <option value="Jammu and Kashmir">Jammu and Kashmir</option> slot
    <option value="Jharkhand">Jharkhand</option> slot
    <option value="Karnataka">Karnataka</option> slot
    <option value="Kerala">Kerala</option> slot
    <option value="Madhya Pradesh">Madhya Pradesh</option> slot
    <option value="Maharashtra">Maharashtra</option> slot
    <option value="Manipur">Manipur</option> slot
    <option value="Meghalaya">Meghalaya</option> slot
    <option value="Mizoram">Mizoram</option> slot
    <option value="Nagaland">Nagaland</option> slot
    <option value="Odisha">Odisha</option> slot
    <option value="Punjab">Punjab</option> slot
    <option value="Rajasthan">Rajasthan</option> slot
    <option value="Sikkim">Sikkim</option> slot
    <option value="Tamil Nadu">Tamil Nadu</option> slot
```

[Kaniel Outis]

Test.describe.parallel is used to write a test to run parallel, to make it fast