//import test from playwright

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

const exp = require('constants');

//write testcase

//test annotation comes from package "playwright/test"

test('Browser context playwright test', async ({ browser }) => {

    //playwright code for UI/API automation

    //to execute script step afer step squentially, have to write 'await' before the line

    //also have to define 'async' before 'function' to make use of 'await'

    //"function()" also known as Anonymous function & can also written as "()=>"

    //as in, async ()=>

    //to invoke browser, pass argument of it in the function()

    //async ({browser})=>

    //curly brace given to browser to make it playwright broweser

    //else it will take it as normal string browser

    //for fresh browser will use .newContext() > will create instance

    const context = await browser.newContext();

    //to create a page from that instance will use page

    const page = await context.newPage();

    //above 2 line> context and page declaration can be removed, by simply passing "page" in function(see next test for ref.)

    //to open url

    await page.goto("https://rahulshettyacademy.com/loginpagePractise/");

    //locators

    //css/xpath/

    //.fill or .type(deprecated)

    //locator by id

    await page.locator("#username").fill("rahulshettyacademys");

    //locator by [attributeName = 'value']

    await page.locator("[type = 'password']").fill("learnings");

    //click on button

    await page.locator("[id = 'signInBtn']").click();

    //to capture a dynamic object and appears only for some second > here its an error msg

    //will use \*

    //will extract the text by ".textContent"

    console.log(await page.locator("[style\*='block']").textContent());

    //in above, it will wait for time thats defined in playwright.config > here its 30 sec

    //assertion for text

    await expect(page.locator("[style \*= 'block']")).toContainText('Incorrect');

});//test case end

//to run below test only> test.only is used

// test.only('Page playwright Test direct call of browser', async ({page})=>

test('Page playwright Test direct call of browser', async ({ page }) => {

    //to open url

    await page.goto("https://google.com");

    //to get the title of the page

    console.log(await page.title());

    //inbuild title assertion

    await expect(page).toHaveTitle("Google");

});

test('TC on type of Locators', async ({ browser }) => {

    const context = await browser.newContext();

    const page = await browser.newPage();

    //defineing locators > so can use multiple times

    const userName = page.locator("#username");

    const passWord = page.locator("[type = 'password']");

    const signinbtn = page.locator("[id = 'signInBtn']");

    const cardtitles = page.locator(".card-body a");

    await page.goto("https://rahulshettyacademy.com/loginpagePractise/");

    await userName.fill("rahulshettyacademysssss");

    await passWord.fill("learningddd");

    await signinbtn.click();

    console.log(await page.locator("[style\*='block']").textContent());

    await expect(page.locator("[style \*= 'block']")).toContainText('Incorrect');

    //fill blank so that it will remove prev strings

    await userName.fill("");

    await userName.fill("rahulshettyacademy");

    await passWord.fill("");

    await passWord.fill("learning");

    await signinbtn.click();

    //traversing PArent > child

    //console.log(await page.locator(".card-body a").textContent());

    //to get first element of the child> nth(index) > it will put child to an array and get the child with matching index

    console.log(await page.locator(".card-body a").nth(0).textContent());

    //or

    //we can use ".first()" to get first element

    console.log(await page.locator(".card-body a").first().textContent());

    //but we only have ".first()" and ".last()"

    //so for more convineient use ".nth()"

    console.log(await cardtitles.nth(1).textContent());

    //to get all child

    console.log(await cardtitles.allTextContents());

});

test('Assignement 1', async ({ page }) => {

    await page.goto("https://rahulshettyacademy.com/client");

    const registerhereProp = page.locator(".text-reset");

    const useremailProp = page.locator("#userEmail");

    const userpasswordProp = page.locator("#userPassword");

    const loginProp = page.locator("[name='login']");

    const fnameProp = page.locator("#firstName");

    const lnameProp = page.locator("#lastName");

    const emailProp = page.locator("#userEmail");

    const phoneProp = page.locator("#userMobile");

    // const dropdownProp = page.locator("[formcontrolname='occupation']");

    // const maleradiobuttonProp = page.locator("[value='Male']");

    const passProp = page.locator("#userPassword");

    const confirmpassProp = page.locator("#confirmPassword");

    const agecheckboxProp = page.locator("[type='checkbox']");

    const registerProp = page.locator("#login");

    // //register

    // await registerhereProp.click();

    // await fnameProp.fill("fname");

    // await lnameProp.fill("lnmae");

    // await emailProp.fill("fname@gmail.com");

    // await phoneProp.fill("1234567890");

    // //dropdown select

    // await page.locator("[formcontrolname='occupation']").selectOption('Student');

    // //radio button

    // await page.locator("[value='Male']").check();

    // await passProp.fill("Fname@1234");

    // await confirmpassProp.fill("Fname@1234");

    // //check box

    // await page.locator("[type='checkbox']").check();

    // await registerProp.click();

    // //clk login

    // await page.locator("[routerlink='/auth']").click();

    //login

    await useremailProp.fill("fname@gmail.com");

    await userpasswordProp.fill("Fname@1234");

    await loginProp.click();

    //wait function >>

    //to get all text by using all text function , we have to use below function

    await page.waitForLoadState('networkidle');

    //or if above .waitFor.State() not working properly then can use ".waitFor()"

    await page.locator(".card-body b").first().waitFor();

    console.log(await page.locator(".card-body b").allTextContents());

    // console.log("---------------------------------------------");

    // //get text of first item

    // console.log(await page.locator(".card-body b").first().textContent());

});

test('UI controls with dropdown\_radiobutton\_checkbox', async ({ page }) => {

    await page.goto("https://rahulshettyacademy.com/loginpagePractise/");

    //select dropdown

    const dropdownProp = page.locator("select.form-control");

    await dropdownProp.selectOption("consult");//pass the value

    //radio button

    //here the loactor is not unique for individual radio button, this locator returning 2 element

    //so we r using ".first()" / ".last()" / ".nth()" accordingly...

    await page.locator(".radiotextsty").last().click();//can also use check()

    await page.locator("#okayBtn").click();

    //assertion for radio button

    await expect(page.locator(".radiotextsty").last()).toBeChecked();

    //other type of radio btn check validation is ".isChecked()"

    console.log(await page.locator(".radiotextsty").last().isChecked());//return boolean

    //check box

    await page.locator("#terms").click();

    await expect(page.locator("#terms")).toBeChecked();

    //to uncheck, inbuild func.

    await page.locator("#terms").uncheck();

    console.log(await page.locator("#terms").isChecked());

    expect(await page.locator("#terms").isChecked()).toBeFalsy();//expecting false

    //in above "await" is inside cause Action is happening inside only(i.e., isChecked)

    //BLINKING text

    //to check blinking class is available

    const blinkTxt = page.locator("[href \*= 'documents-request']");

    await expect(blinkTxt).toHaveAttribute("class", "blinkingText");

    await page.pause();//it will pause the script and it will open a Inspector of Playwright

});

test('Child Window Handle', async ({ browser }) => {

    const context = await browser.newContext();

    const page = await context.newPage();

    //if we clk on a link and its opening on diff tab then we have to handle it properly

    await page.goto("https://rahulshettyacademy.com/loginpagePractise/");

    const blinkTxt = page.locator("[href \*= 'documents-request']");

    //here after clk a new tab will open, but "page" is only able to hanlde one tab > we have to switch the focus

    //so we can approach below methods to handle it

    //here we need to define 'browser' instead of only 'page'

    //cause we need to deal with another page context

    //const page2 = context.waitForEvent('page');//passing page whch is for new event

    //above waitforevent will look for new when we define it before new event happens

    //so we have to define this before new page or clicked on blinktxt

    //await blinkTxt.click();

    //even with aboove steps defiend it might not get the new tab

    //we need wait for event and click to happen at async but parrallaly

    //in order to listen for new page > it should be either pending/rejected/fulfilled

    //below is an array, which will keep on trying to go from pending to fulfilled at any cost

    //after its fulfilled then only it will go to next step else it will fail

    const [newPage] = await Promise.all(

        [

            context.waitForEvent('page'),

            blinkTxt.click(),

        ]

    )//new page opened

    //to handel new page >

    const txt = await newPage.locator(".red").textContent();

    console.log(txt);

    //Challenge:>  LETS SAY WE WANT TO RETREIVE EMAIL/ADDRESS FROM NEWPAGE AND ENTERS/USE IT IN OLD/PREV. PAGE EMIAL BOX >>

    //first split it by '@' store it in array(split into 2 parts> left and right and stored in 0 and 1 index)

    const arrayText = txt.split("@");

    //now again split right side which is in index 1 of the array by a space

    const domain = arrayText[1].split(" ")[0]

    console.log(domain);

    //NOW go back to prev tab and enter this domain text

    //simply use parent/prev tab page object to do it

    await page.locator("#username").fill(domain);

    //await page.pause();

    //TO HANDLE> 2nd tab or 3rd tab or more, we can simply add new page objcet to the array

    // const [newPage,newPage2,newPage3,...so on] = await Promise.all(

    //     [

    //         context.waitForEvent('page'),

    //         blinkTxt.click(),

    //     ]

    // )

});

////////////////////////////////////////////////////////////////

//to run in DEBUG mode type in termnal>

//npx playwright test tests/UiBasicstest.spec.js --debug

//RECORD feature by PLAYWRIGHT>

//npx playwright codegen URLtypeHere

test('test for RecordNPlayback feature in playwright', async ({ page }) => {

    await page.goto('https://www.google.com/');

    await page.getByLabel('Search', { exact: true }).click();

    await page.getByLabel('Search', { exact: true }).fill('facebook');

    await page.getByText('Facebook', { exact: true }).click();

    await page.getByRole('link', { name: 'Facebook - log in or sign up' }).click();

    await page.getByRole('img', { name: 'Facebook' }).click();

    await page.getByTestId('royal\_email').click();

    await page.getByTestId('royal\_email').fill('1212');

    await page.getByTestId('royal\_email').click();

    await page.getByTestId('royal\_email').fill('username');

    await page.getByTestId('royal\_pass').click();

    await page.getByTestId('royal\_pass').fill('password');

    await page.getByTestId('royal\_login\_button').click();

});//recorded by playwright

////////////////////////////////////////////////////////////////

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

test('Assignement 1', async ({ page }) => {

    await page.goto("https://rahulshettyacademy.com/client");

    const registerhereProp = page.locator(".text-reset");

    const useremailProp = page.locator("#userEmail");

    const userpasswordProp = page.locator("#userPassword");

    const loginProp = page.locator("[name='login']");

    const fnameProp = page.locator("#firstName");

    const lnameProp = page.locator("#lastName");

    const emailProp = page.locator("#userEmail");

    const phoneProp = page.locator("#userMobile");

    // const dropdownProp = page.locator("[formcontrolname='occupation']");

    // const maleradiobuttonProp = page.locator("[value='Male']");

    const passProp = page.locator("#userPassword");

    const confirmpassProp = page.locator("#confirmPassword");

    const agecheckboxProp = page.locator("[type='checkbox']");

    const registerProp = page.locator("#login");

    //e2e

    const productNameToValidate = 'IPHONE 13 PRO';//we r looking for this prod. and will add this to cart. This is we r going to deal with in this test

    const products = page.locator(".card-body");

    //login

    const emailId = "fname@gmail.com";

    await useremailProp.fill(emailId);

    await userpasswordProp.fill("Fname@1234");

    await loginProp.click();

    await page.waitForLoadState('networkidle');

    await page.locator(".card-body b").first().waitFor();

    console.log(await page.locator(".card-body b").allTextContents());

    //get the product count so that we can travers thru it and look for particular product

    const count = await products.count();

    for (let i = 0; i < count; ++i) {

        //now look for product namme wise

        //we can use .locator again like chaining

        //to look for can use ''nth()' for it

        const pname = await products.nth(i).locator("b").textContent(); //it will start looking from only here> page.locator(".card-body");

        if (pname === productNameToValidate) {

            //if matched then > add to cart

            //again we can use same parent locator and use chain method

            await products.nth(i).locator("text = Add To Cart").click();//locator by text

            break;//product found so ne need to go further

        }

    }

    //click on CART

    await page.locator("[routerlink \*= 'cart']").click();

    //wait function,to be loaded>wait for

    await page.locator("div li").first().waitFor();

    //we r using above wait cause > for "isVisible()" method play wright dont give auto wait

    //validate added prod is in CART

    //search for product name with locator dynamically

    const bool = await page.locator("h3:has-text('IPHONE 13 PRO')").isVisible();

    expect(bool).toBeTruthy();

    //click Checkout

    await page.locator("text = Checkout").click();

    //DropDown> after typing something, text appers and we select that text

    // await page.locator("[placeholder\*='Country']").fill("ind");//after type ind , suggestion will show up

    //but above aproach will just paste it> suggestion wont appear

    //Only way is to type it slowly with letter by letter >

    await page.locator("[placeholder\*='Country']").pressSequentially("ind", { delay: 100 });

    //after suggestion appears >

    //get the locator of the tab where suggestions r apeearing

    //then will wait fot it

    //then with Chaining method > travers thru it and look for the Text that we want to select

    const dropDownOption = page.locator(".ta-results");

    await dropDownOption.waitFor();

    const optionCount = await dropDownOption.locator("button").count();

    //look for desired text

    for (let i = 0; i < optionCount; i++) {

        const dText = await dropDownOption.locator("button").nth(i).textContent();

        if (dText === " India") {

            //matches >?> clk

            await dropDownOption.locator("button").nth(i).click();

            break;

        }

    }

    //validate email which is in gray color

    await expect(page.locator(".user\_\_name [type='text']").first()).toHaveText(emailId);

    //in above locator we r traversing from parent (.user\_\_name) to child ([type='text']) locator

    //if class : value is having space then we can use any one like above case

    //class:a b >>> here we can use like> (.a) or (.b) to locate the elem. thru class prop

    //clk on place order

    await page.locator(".action\_\_submit").click();

    //validate Thank you for the order text

    await expect(page.locator(".hero-primary")).toHaveText(" Thankyou for the order. ");

    //grab the order id

    const orderIdText = await page.locator(".em-spacer-1 .ng-star-inserted").textContent();

    console.log("Order Id: " + orderIdText);

    //clik on my order

    await page.locator("[routerlink = '/dashboard/myorders'] i").click();

    await page.locator("tbody").waitFor();

    //will validate the order id in my order section

    const orderTabs = page.locator("tbody tr");//parent child > common locator

    const orderCount = await orderTabs.count();

    for (let i = 0; i < orderCount; i++) {

        const oId = await orderTabs.nth(i).locator("th").textContent();

        //console.log("order id in my order: "+oId);

        if (orderIdText.includes(oId)) {

            await orderTabs.nth(i).locator("button").first().click();

            break;

        }

    }

    //await page.locator(".col-text").waitFor();

    const oidInView = await page.locator(".col-text").textContent();

    expect(orderIdText.includes(oidInView)).toBeTruthy();

    //await page.pause();

});