WORD FILE GITHUBBED

PLAYWRIGHT JAVASCRIPT

CHAI ASSERTIONS(use only chai assertions)

<https://www.chaijs.com/api/bdd/>

CREATE A NEW WORKSPACE 1)file->NewWindow(new Tab in visual code opens) 2)create a folder in D 3)file->openfolder>select that folder u created

Header File For Running JsProgram const playwright = require('playwright')

Test Function async EnterUserName(Username) {

return await page.locator("//input[@id='username']").fill(Username);}

const { setDefaultTimeout } = require('@cucumber/cucumber'); //**require('@cucumber/cucumber')** is a common way to import the Cucumber library in Node.js applications that use Cucumber for testing.

Run Program npm run test

URL PASS await.page.goto(‘URL’)

GET Current Page URL await page.url();

Variable Declaration const Username=

Int Username=

SendKeys() await page.locator("//input[@class='textField pwdfield']").fill(Password)

await page.locator('#edit-keys').type('Cypress')

KEYBOARD ACTION

await page.keyboard.press('Tab', { count: 3 });//press tab 3 times

TO SEE EXECUTION SLOW

BeforeAll(async()=>

{

global.browser= await playwright['chromium'].launch({ headless: false, slowMo: 1000 });

}

)

AVOID TIMEOUT EXCEPTION

In step definition(Add these lines at the start)

const { setDefaultTimeout } = require('@cucumber/cucumber');

setDefaultTimeout(60 \* 1000);

To get Page Title

const pageTitle = await page.title();

ASSERT URL

const DashBoardUrl=await page.url();//gets dashboard URl

console.log(DashBoardUrl);

expect(DashBoardUrl).contain("dashboard");//check url has dashboard word in it

?

Const myFrame=page.frame(“firstFr”)//only the name of the frame need to be passed

Await myFrame?fill(“input[name=’fname’]”,“sara”)

myFrame?-->if frame is not null then perform the action

HOVER OVER AN ELEMENT CHECK WHILE HOVERING BORDER IS PRESENT OR NOT FOR THAT ELEMENT

const elemHandle = await page.waitForSelector("//div[@id='nav-logo']");

const style = await elemHandle.evaluate((elem) => {

return window.getComputedStyle(elem).getPropertyValue('border');

})

if (style !== 'none') {

console.log('Border present!');

} else {

console.log('No border present.');

}

SCROLL PAGE UPTO A PARTICULAR WEBELEMENT

const signin = await page.$("//label[text()='2022 - All Rights Reserved']");

expect(signin).to.exist;

await signin.scrollIntoViewIfNeeded();

Wait Condition await page.waitForTimeout(5000);

await page.goto('https://playwright.dev', { timeout: 30000 });  
 await page.getByText('Get Started').click({ timeout: 10000 });

waitForSelector this wait for the selector to be appear in page

await page. waitForSelector(“app-gitrepos ol li”)

allInnerTexts() const repos=page.locator(“app-gitrepos ol li”)//contains 10 elememts

concsole.log(await repos. allInnerTexts());//prints all 10 elements

count console.log(await repos. count);//return no of elements

GET SIZE() &GET LOCATION

Const src=await page.locator

Const srcbound=await src.boundingBox()

Console.log(srcbound.x)//get x coordinates

Console.log(srcbound.y)//get y coordinates

Console.log(srcbound.width)//get width

Console.log(srcbound.height)//get height

const usernameField = await page.$('#username');

const passwordField = await page.$('#password');

const usernameBoundingBox = await usernameField.boundingBox();

const passwordBoundingBox = await passwordField.boundingBox();

console.log('Username field width:', usernameBoundingBox.width);

console.log('Username field height:', usernameBoundingBox.height);

console.log('Password field width:', passwordBoundingBox.width);

console.log('Password field height:', passwordBoundingBox.height);

CSS Property like background-colour,text-decoration and cursor-pointer

get CSS Value 🡺 getComputedStyle(element)

Clear()-Clear the input field.

await page.getByRole('textbox').clear();

DRAG AND DROP

const sourceLocator = page.locator("//div[@id='form:drag\_content']");

const targetLocator = page.locator("//span[text()='Droppable Target']");

const sourceBoundingBox = await sourceLocator.boundingBox();//gets x,y co-ordinates and width and height

const targetBoundingBox = await targetLocator.boundingBox();

const sourceX = sourceBoundingBox.x + sourceBoundingBox.width / 2;//gets x-coordinate and width/2 means it moves to center

const sourceY = sourceBoundingBox.y + sourceBoundingBox.height / 2;//gets y-coordinate and height/2 means it moves to center

const targetX = targetBoundingBox.x + targetBoundingBox.width / 2;//gets x-coordinate and width/2 means it moves to center

const targetY = targetBoundingBox.y + targetBoundingBox.height / 2;

await page.mouse.move(sourceX, sourceY);//from->moving src

await page.mouse.down();

await page.mouse.move(targetX, targetY);//to target->

await page.mouse.up();

CLICK AN ELEMENT AND MOVE IT TO RIGHT

const elementHandle = await page.$("//div[@id='form:conpnl']");

await elementHandle.evaluate((el) => {

const style = window.getComputedStyle(el);

const left = parseInt(style.left) || 0;

el.style.left = `${left + 700}px`;

});

isVisible()

const iamabouttohide = await page.locator("//span[text()='I am about to hide']");

expect(iamabouttohide).to.exist;

const button = await page.locator("//button[@id='j\_idt87:j\_idt92']").click();

await page.waitForTimeout(10000);

const buttondisappered=await iamabouttohide.isVisible();

expect(buttondisappered).to.be.false;

MULTIWINDOW HANDLING

const [newWindow] = await Promise.all([

context.waitForEvent("page"),

await page.click("//span[text()='Open Multiple']")

])

await newWindow.waitForLoadState();

const PagesOpened=newWindow.context().pages();

console.log("No of tabs Opened are "+PagesOpened.length)

expect(newWindow.url()).contain("table");

const rows = await newWindow.$$("//table[@role='grid']//tr[6]");

for (const row of rows) {

const cells = await row.$$('td');//we r taking the table data

for (const cell of cells) {

const value = await cell.innerText();//and printing the values in table data

console.log(value);

}

}

await newWindow.close();//closes table tab

await page.bringToFront();//bring homepage to front

MOVE SLIDER USING KEYBOARD ACTION&Mouse Action

//if not working for $ then try locator

const elementHandle = await page.locator("(//span[@tabindex=0])[1]").click();

OR

const elementHandle = await page.$("(//span[@tabindex=0])[1]");

await elementHandle.click();

// Move the element to the right using the right arrow key

const keyboard = await page.keyboard;

await keyboard.press('ArrowRight');

await keyboard.press('ArrowRight');

await keyboard.press('ArrowRight');

await keyboard.press('ArrowRight');

const text=await page.locator("(//div[@class='card'])[9]/table/tbody/tr/td/span").textContent();

console.log(text);

OR USING MOUSE ACTION

const slider1 = page.locator("//span[@tabindex='0'][1]");

const targetLocator = page.locator("//span[@tabindex='0'][2]");

const slider1BoundingBox = await slider1.boundingBox();//gets x,y co-ordinates and width and height

const targetBoundingBox = await targetLocator.boundingBox();

const sourceX = slider1BoundingBox.x + slider1BoundingBox.width / 2;//gets x-coordinate and width/2 means it moves to center

const sourceY = slider1BoundingBox.y + slider1BoundingBox.height / 2;//gets y-coordinate and height/2 means it moves to center

const targetX = targetBoundingBox.x + targetBoundingBox.width / 2;//gets x-coordinate and width/2 means it moves to center

const targetY = targetBoundingBox.y + targetBoundingBox.height / 2;

await page.mouse.move(sourceX, sourceY);//from->moving src

await page.mouse.down();

await page.mouse.move(targetX, targetY);//to target->

await page.mouse.up();

PRINT ALL SHOE NAMES IN AMAZON PAGE

const All\_Shoes\_Name=await page.$$("//span[@class='a-size-base-plus a-color-base a-text-normal']");//gets all shoe names

for(const Shoe\_Name of All\_Shoes\_Name)

{

console.log("All ShoeNames are "+await Shoe\_Name.textContent());

}//if u use locator instead of $ and if u don’t use await Shoe\_Name it won’t print

WRITE DATA IN JSON FILE

// Create an object to write to the file

const data = {

name: 'John Doe',

age: 30,

email: 'john.doe@example.com'

};

// Convert the object to a JSON string

const jsonData = JSON.stringify(data, null, 2);//null->no additional options r specified,2->no of indentataion(white space or tabs)

// Write the JSON data to a file

writeFile('D:/workspace/workspace1/config/data.json', jsonData , (error) =>

{

//if (error) throw error;

console.log('Data written to file!');

});

CHECK WHETHER BUTTON isDisabled(),isEnabled(),isEditable()

async TestLeafCheckDisabled()

{

const btn=await page.$("//input[@id='j\_idt88:j\_idt93']");

const btnn= await btn.isDisabled();or isEnabled() or isEditable()

return expect (btnn).to.be.true;

}

Click await page.locator(‘selector’).click();

Hover await page.locator(“appbhb”).hover();

Set Dimensions for Browser await page.setViewportSize({ width:300, height: 300 });

Get into Frame &Get No Of IFrames const allframes=page.frames();//get into frame

console.log("No of:"+allframes.length);

ScreenShot OfPar const ref= await page.locator('#loginButton').screenshot({ path: 'screenshots/LoginButton.png'});

ScreenShotFullPage const ref=await global.page.screenshot({ path: 'screenshots/Login.png',fullPage:true});

Retry TestCase npx playwright test—retries=2

PATH IN CUCUMBER.JS

stepdefinition/ActitimeStep.step.js --require stepdefinition/\*\*/\*.step.js

features/WFH-001.feature default: `${common} features/\*\*/\*.feature`,

hooks/hooks.js&assertions.js --require hooks/\*\*/\*.js (gets both hooks and assertions present)

TAGS

test(‘ScrollDown @pb’,async({page}=>{

});

test(‘Actitime @auto’,async({page}=>{

});

npx playwright test --grep ‘@pb’ now only scrolldown runs

npx playwright test --grep-invert ‘@pb’ now actitime skipped and scrolldown alone works

FOR GET TEXT (Refer this website)🡪 https://www.programsbuzz.com/article/playwright-get-text-element#element-text

TO PRINT ONLY 1 Element textContent() split

POM FILE

async GetIdentifyyourself()

{return await page.locator("//td[text()='Please identify yourself']").textContent(); }

STEP Definition File

const text= await actitime.GetIdentifyyourself();

const split=text.split(' ')[1];

console.log("splitted text we got is "+split);

UPLOAD FILES

await page.locator("(//input[@type='file'])[1]").setInputFiles(["D:/workspace/workspace1/config/ppple.txt"]);

DOWNLOAD FILES

SELECT DROPDOWN VALUES(selectOption({ label: 'Playwright' });)

const FavouriteUITooldropdown = await page.$('select.ui-selectonemenu');//select option with classname

// // Click on the dropdown to open it

await FavouriteUITooldropdown.click();//click the dropdown 1st

await FavouriteUITooldropdown.selectOption({ label: 'Playwright' });

OR

await FavouriteUITooldropdown.selectOption({ index: 2 });

//sometimes u have click different element 1st and then u have to get the select option where u will get the options. Eg

const All = await page.$("//div[@class='nav-search-scope nav-sprite']");//if u inspect this will show so 1st click this

await All.click();

const SelectAll=await page.$("//select[@id='searchDropdownBox']");//then here only we got the select and options so get options from here

await SelectAll.selectOption({ label: 'Video Games' });

PRINT ALL VALUES FROM DROPDOWN

const DropDownElements=await page.$$("//select[@id='first']/option");

for(const DropDownElement of DropDownElements)

{

console.log(await DropDownElement.textContent());

}

TO PRINT LIST(MORE THAN 1 Elements)

Const allText=await page.locator(‘menu li’).allTextcontent()

TO PRINT ALL URLS IN A PAGE

We can’t call from POM directly write in StepDefinition

Then('User prints all links in Amazon page', async () => {

const links = await page.$$('a');

for (const link of links) {

const href = await link.getAttribute('href');

console.log("List Of All Urls Present Are "+href);

}

});

TO FIND OCCURANCES OR NO OF TIMES PARTICULAR ELEMENT PRESENT IN A PAGE(length)

const Bata\_Shoes = await page.$$("//span[text()='BATA']");//gets all the bata shoes

console.log(`Number of Bata Shoes Present are: ${Bata\_Shoes.length}`);

SELECT THE PRICE BY NAME OF SANDAL FOR MEN IN AMAZON

//span[text()='Men Cedar Fisherman Tan Sandals']//following::div[3]/div[2]/a/span/span[1]

It gets the price of sandal based on name so even if u change the sandal name alsoeg MenSHREK also it gets the price of MenSHERK sandal

DATE(Create a file in config folder and add this )

1. npm i fs //install fs first

const fs = require('fs');

const path = require('path');

const currentDate = new Date().toISOString().replace(/:/g, '-');

const dir = path.join('report\_cucumber', currentDate);

// Create the parent directory recursively

fs.mkdirSync('report\_cucumber', { recursive: true });

// Create the target directory

fs.mkdirSync(dir, { recursive: true });

console.log("Print Date "+currentDate);

console.log("Print Dir "+dir);

// Export the date variable for use in other modules

module.exports = {currentDate };

ADD THESE LINES IN CUCUMBER FILE At the Top

const Date = require('./config/Date');

const Date1=Date.currentDate;

console.log("DDDD is "+Date1);

const common = `

--format json:report\_cucumber/${Date1}/cucumber-html-report.json

--format html:report\_cucumber/${Date1}/cucumber-html-report.html

AT HOOKS FILE(Add these lines at top)

const Date = require('../config/Date');

const Date1=Date.currentDate;

console.log("Hoks js"+Date1);

After()

if (traceRecorder == true) { const trace = await global.context.tracing.stop({ path: `./report\_cucumber/${Date1}/trace/${scenario.pickle.name}.zip` }); }

//so in report\_cucumber folder and inside date Folder trace will be saved

const ref = await global.page.screenshot({ path: `report\_cucumber/${Date1}/failedScreenshot/${scenario.pickle.name}.png`, fullPage: true });

//so in report\_cucumber folder and inside date Folder failed Screenshot will be saved

HANDLING SHADOWDOM ONLY USING CSS SELECTOR NOT XPATH

await page.locator('#searchq').fill("annular");

//passing value to textbox inside shadowdom only use cssSelector don’t use xpath it won’t work

HANDLING SVG ELEMENTS

//Like ShadowDom here also use only CssSelector don’t use Xpath

await page.goto("https://www.google.co.in");

const svgElement = await page.waitForSelector('svg');

await svgElement.click();

// Get the viewBox attribute of an SVG element

const viewBox = await svgElement.getAttribute('viewBox');

console.log('The viewBox attribute is:', viewBox);

// Take a screenshot of an SVG element

await svgElement.screenshot({ path: 'config/google.png' });//this line takes the screenshot of the particular svgelement and store it in the referred path

TO GET THE NO OF SVG ELEMENTS IN A PAGE

const svgElements = await page.$$('svg'); // Select all SVG elements in the page

const numSvgElements = svgElements.length; // Get the number of SVG elements

console.log(`Number of SVG elements: ${numSvgElements}`);

const QuickSettings=await page.$("#tsf > div:nth-child(1) > div.A8SBwf > div.RNNXgb > div > div.dRYYxd > div.XDyW0e > svg");

await QuickSettings.click();

ALERT PROMPT ENTER VALUE IN PROMPT PRINT VALUE IN ALERT

await page.goto(‘https://letcode.in/alert’);

//IF THE PAGE LOADS FOR MORE TIME U CAN’T DIRECTLY CLICK ALERT OR PROMPT YOU HAVE TO USE WAIT AND PASS LOCATOR THERE

const btn=await page.waitForSelector("//button[@id='prompt']")

btn.click();

~~//await page.locator("//button[@id='prompt']").click();~~

const alert=await page.waitForEvent('dialog');

//const message=alert.message();

await alert.accept("Annular");

const result=await page.locator("//p[@id='myName']").textContent();

console.log("Entered Text is "+result);

expect (result).to.include('Annular');

//PROMPT IN W3SCHOOLS

const frame = await page.frame({ name: 'iframeResult' });

//const frame=await page.frame("//iframe[@id='iframeResult']");

//button[text()='Try it']

// await frame.locator("//button[text()='Try it']");

const btn=await frame.waitForSelector("//button[text()='Try it']")//we r passing selector for prompt btn inside wait statement bcoz the page takes time while loading

btn.click()

const alert = await page.waitForEvent('dialog');

const message=alert.message();

await alert.accept("Annular");

console.log("Text Present In Alert is "+message);

await page.waitForTimeout(3000)

console.log("hi")

const result1=await frame.locator("//\*[@id='demo']").textContent()

//await page.waitForTimeout(3000)

console.log(result1);

console.log("hello");

MAXIMIZE BROWSER FOR PARTICULAR PROGRAM

// const session = await context.newCDPSession(page);

// const { windowId } = await session.send('Browser.getWindowForTarget');

// await session.send('Browser.setWindowBounds', { windowId: windowId, bounds: { windowState: 'maximized' } });

// await session.detach();

PRINT ALTERNATE Rows IN TABLE

const rows = await newWindow.$$("//table[@role='grid']//tr");

for (let i = 0; i < rows.length; i += 2) {

const rowText = await rows[i].innerText();

console.log(rowText);

}

TRACE RECORDER START AND STOP add screenshot of failed scenario in report

Before(async function (scenario) {

global.context = await global.browser.newContext({ viewport: { width: 1280, height: 720, } })

global.context = await global.browser.newContext({

viewport: {

width: 1280,

height: 720,

},

recordVideo: {

dir: `./report\_cucumber/Videos/${scenario.pickle.name}.webm`

},

})

if (traceRecorder == true) { await global.context.tracing.start({ screenshots: true, snapshots: true }); }

global.page = await global.context.newPage();

// await global.page.viewport({ width: 1280, height: 720 });

})

After(async function (scenario) {

if (scenario.result.status === Status.PASSED) {

// console.log('Passed scenarios name has printed ==> ' + `${scenario.pickle.name}`)

if (traceRecorder == true) { const trace = await global.context.tracing.stop({ path: `./reports/trace/${scenario.pickle.name}.zip` }); }

}

else {

const ref = await global.page.screenshot({ path: `reports/failedScreenshot/${scenario.pickle.name}.png`, fullPage: true });

// console.log('Failed scenarios name has printed and screenShots taken ==> /' + `${scenario.pickle.name}`)

this.attach(ref, 'image/png', scenario.pickle.name); // attach screenshot to report

if (traceRecorder == true) { const trace = await global.context.tracing.stop({ path: `./reports/trace/${scenario.pickle.name}.zip` }); }

}

})

GET **THE TEXT PRESENT IN TEXTBOX**

const TypedText=await page.$("//input[@id='j\_idt88:j\_idt97']");

const TypedTextValue=await TypedText.getAttribute('value');

console.log("Text Present in the TextBox is "+TypedTextValue)

DOWNOAD FILE

//link for demo is https://www.lambdatest.com/selenium-playground/generate-file-to-download-demo

// await page.type("#textbox","Like ,share , Comment ");//fill the textbox

// await page.click("id=create");//click generate button

// const download=await Promise.all([

// page.waitForEvent("download"),

// page.click("id=link-to-download")//click the download button

// ])

//this 2 lines used to download file inside playwright itself

const path = await download[0].path();

console.log(path);

const fileName = download[0].suggestedFilename();

//await download[0].saveAs(fileName);

await download[0].saveAs('D:/workspace/workspace1/reports/Downloads/TestLeaf Logo.png');

GET THE COLOUR OF THE BUTTON(CSS BACKGROUND COLOR)

const btn=page.locator("//button[@id='j\_idt88:j\_idt96']");

const color=await btn.evaluate((ele)=>

{

return window.getComputedStyle(ele).getPropertyValue("background-color")

})

console.log("Color of the button is "+color);

FIND THE DEFAULT SELECTED RADIOBUTTON

// Assuming you have a list of radio buttons with 'input' tag name and 'type' attribute as 'radio'

const radioButtons = await page.$$('input[type="radio"]');

let defaultSelectedButton;

// Iterate through each radio button element

for (const radioButton of radioButtons) {

// Check if the current radio button is selected

const isSelected = await radioButton.isSelected();

// If the radio button is selected, assign it to the defaultSelectedButton variable and break the loop

if (isSelected) {

defaultSelectedButton = radioButton;

break;

}

}

HOVER OVER A BUTTON AND CHECK WHETHER THE BACKGROUNDCOLOR OF BUTTON CHANGED OR NOT

const button = await page.$("(//span[text()='Success'])[1]"); // Select the button element

const buttonBgColorBeforeHover = await button.evaluate(button => getComputedStyle(button).getPropertyValue('background-color')); // Get the background-color //property before hover

await button.hover(); // Hover over the button using the built-in hover function

const buttonBgColorAfterHover = await button.evaluate(button => getComputedStyle(button).getPropertyValue('background-color')); // Get the background-color

//property after hover

console.log(`Button background color before hover: ${buttonBgColorBeforeHover}`);

console.log(`Button background color after hover: ${buttonBgColorAfterHover}`);

if(`Button background color before hover: ${buttonBgColorBeforeHover}`==`Button background color after hover: ${buttonBgColorAfterHover}`)

{

console.log("Color not changed")

}

else

{

console.log("Color changed")

}

CHECK WHETHER BUTTON IS ROUND OR NOT

const button = await page.$("//button[@id='j\_idt88:j\_idt108']"); // Select the button element

const isButtonRound = await button.evaluate(button => {

const borderRadius = getComputedStyle(button).getPropertyValue('border-radius'); // Get the value of the border-radius CSS property

return borderRadius.endsWith('50%') || borderRadius.endsWith('999em'); // Check if the value ends with '50%' or '999em', which indicates a round button

});

if (isButtonRound) {

console.log('Button is round');

} else {

console.log('Button is not round');

}

CLICK BUTTON INSIDE A FRAME USING SRC OF FRAME

const frame=page.frameLocator("iframe[src='default.xhtml']")

// Click on the button inside the frame

await frame.locator("(//button[@id='Click'])[1]").click();

FRAME CAN BE ACCESSED ONLY USING NAME

SWITCH TO FRAME AND PRINT ALL VALUES IN TABLE INSIDE THE FRAME

Then('User prints all links in Amazon page', async () => {

const AllFrames=page.frames();

console.log("No Of Frames are "+AllFrames.length);//return no of frames in page

const frame = await page.frame({ name: 'iframeResult' });//name of the frame

const rows = await frame.$$('table tr');

for (const row of rows) {

const cells = await row.$$('td');//we r taking the table data

for (const cell of cells) {

const value = await cell.innerText();//and printing the values in table data

console.log(value);

}}}

);

ASSERT WHETHER VERTICAL or HORIZONTAL SCROLLBAR IS PRESENT OR NOT

const hasVerticalScrollbar = await page.evaluate(() => {

const { scrollHeight, clientHeight } = document.documentElement;

return scrollHeight > clientHeight;

})

if(!hasVerticalScrollbar)

{

console.log("ScrollBar is Present");

}

else

{

console.log("ScrollBar is not Present");

}

Promise.all([])

When performing 2 actions simultaneously eg:click and waitForEvent put it inside promise

await Promise.all([

page.waitForEvent(“popup”),

page.click(“FOLLLII”)

]);

Await page.waitForLoadState(); if u r opening 2 pages and 1 page loads before another page use this

SCREENSHOT OF FAILED SCENARIO WITH NAME OF SCEANRIO

After(async function(scenario) {

if(scenario.result.status===Status.PASSED)

{

const ref=await global.page.screenshot({path:`passedscenario/${scenario.pickle.name}.png`,fullpage:true});

console.log('Scenario Passed' + `${scenario.pickle.name}`)

}

EVALUATE USERID STATUS

async userIDstatus() {

return await page.$eval('#usernameStatus', element => element.textContent);

}

SELECT ALTERNATE CHECKBOXES IN LIST OF CHECKBOXES

const CheckBoxes = await page.$$("//div[@class='ui-chkbox-box ui-widget ui-corner-all ui-state-default']");

for (let i = 0; i < CheckBoxes.length; i += 2) { // iterate over every other checkbox

await CheckBoxes[i].click(); // click the checkbox

}

READ VALUES FROM EXCEL SHEET

1.npm i xlsx//in terminal for reading excel

// Load the Excel workbook and worksheet

const workbook = new ExcelJS.Workbook();

await workbook.xlsx.readFile('config/datasheet.xlsx');//give the location of excel sheet

const worksheet = workbook.getWorksheet(1);

// Loop through the rows in the worksheet

for (let i = 1; i <= worksheet.rowCount; i++) {

const row = worksheet.getRow(i);

// Get the values from the current row

const values = row.values;

for(const value of values)

{

console.log(value);

}

}

WRITE VALUES IN EXCEL FILE

const workbook = new ExcelJS.Workbook();

const worksheet = workbook.addWorksheet('Sheet2');

worksheet.addRow([1, 'Annular', 'MANAGING DIRECTOR']);

worksheet.addRow([2, 'Annular', 'SelvaKugan']);

worksheet.addRow([3, 'Annular', 'sellamuthu']);

worksheet.addRow([4, 'Annular', 'Ajith']);

worksheet.addRow([5, 'Annular', 'Krishna']);

worksheet.addRow([6, 'Annular', 'Francis']);

worksheet.addRow([7, 'Annular', 'Kalai']);

worksheet.addRow([8, 'Annular', 'Kanishka']);

await workbook.xlsx.writeFile('config/datasheet.xlsx');

PAGINATION Print All Values From Page1 and click Next and Print All Values From Page2 and SoOn

let currentPageNum = 1;

while (true) {

const allValues = await page.$$eval("//span[@class='a-size-base-plus a-color-base a-text-normal']", values => values.map(value => value.textContent));

for(const value of allValues)

{

console.log("the current Page Values are "+value);//prints all values in a page

}

console.log("The Next Page Values ");

// Check if there is a "Next" button

const nextButton = await page.$("//a[text()='Next']");

if (!nextButton) {

break; // Exit the loop if there is no "Next" button

}

// Click the "Next" button and wait for navigation

await Promise.all([

page.waitForNavigation(),

nextButton.click(),

]);

currentPageNum++; // Increment the current page number

}

SEARCH FOR A PARTICULAR ELEMENT FROM PAGE TO PAGE IN PAGINATION

// Loop through all the pages

while (true) {

// Search for the element on the current page

const element = await page.$("//span[contains(text(),' Ripley Sandal Brown\_4 7 Kids UK (8614413)')]");

if (element) {

// If found, get its value and print it

const value = await element.textContent();

console.log(`Found the element with value: ${value}`);

// const Price=await page.$("(//span[text()='MenSHREK TR E Slipper UK 9 Color Black (8716538)']/following::div[3]/div/a/span/span/span)[1]");

// const tt=Price.textContent();

// console.log(tt);

break; // Stop the loop

}

// If not found, check if there is a next page

const nextButton = await page.$("//a[text()='Next']");

if (!nextButton) {

console.log('Element not found');

break; // Stop the loop

}

// If there is a next page, click the next button and wait for the page to load

await Promise.all([

page.waitForNavigation(),

nextButton.click(),

]);

}

COMMIT CODE IN GITHUB

1.create empty folder named Pract in D:

2.create repository named Prac in github

3.IN Vscode-🡪file🡪newwindow🡪open the pract folder

o/p

File

OPEN EDITORS

PRACT

//like above pract folder will be inside vscode

4.copy URL from git Repository

5.ctrl+J in vscode🡪git clone <https://github.com/Saravanan128/prac.git>

O/P will be like below

PS D:\pract> git clone https://github.com/Saravanan128/prac.git

Cloning into 'prac'...

remote: Enumerating objects: 3, done.

remote: Counting objects: 100% (3/3), done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

Receiving objects: 100% (3/3), done.

6.now in vscode

PRACT

Prac

README.md

//like above README.md which present in the remote repository wil be present in ur PRACT folder in vscode

7.Now Create a file named cucumber.js inside prac folder before README.md

8.as soon as u added cucumber file it turns green and U symbol will appear

9.now u can click on source ctrl button and there this cucumber file will be added and u can click plus and stage changes and commit and commit and sync it

10.now cucumber file u created will be commited to git repository and if u add new lines or make change to cucumber file then it will be commited in git

11.Likewise add files and folders 1 by 1

12.

IMPORTANT POINT

Always add 1 by 1 eg 1st create cucumber file ,then create feature file ,then step def,ten config,then

npm install chai,

npm install playwright,

npm install prettier//see pdf file

NEVER COPY PASTE ENTIRE FOLDER,SUBFOLDERS AT ONCE DO 1 BY 1

13.check package.json file whether every datas are present or not if not copy code from annuar package file and paste it here