We define own config

Ex1. stage.config.js

const { defineConfig } = require('cypress')

module.exports = defineConfig({

    video: true,

    e2e: {

        baseUrl: "https://www.amazon.com"

    }, env: {

        username: "testing stage"

    }

})

Ex2. Prod.config.js

const { defineConfig } = require('cypress')

module.exports = defineConfig({

    video: true,

    e2e: {

        baseUrl: "https://www.google.com"

    },

    env: {

        username: "testing prod"

    }

})

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

,ignore file content

node\_modules

package-lock.json

node\_modules/.package-lock.json

yarn.lock

take 'screenshotWithTimestamp' custom command in support/commands

Cypress.Commands.add('screenshotWithTimestamp', (name) => {

    const timestamp = new Date().toLocaleString();

    const fileName = `${name}-${timestamp}`;

    cy.log(`Taking screenshot: ${fileName}`);

    cy.screenshot(fileName, { capture: 'viewport' });

    cy.get('body').then(($body) => {

        const $dateTime = Cypress.$(`<div>${timestamp}</div>`);

        $body.append($dateTime);

        cy.log(`Timestamp: ${timestamp}`);

    });

});

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

#'dragAndDrop'

Cypress.Commands.add('dragAndDrop', (dragElement, dropElement) => {

    cy.get('div[id^=box]:contains(' + dragElement + ')').trigger('mousedown', { which: 1 })

    cy.get('div[id^=box]:contains(' + dropElement + ')').trigger('mousemove').trigger('mouseup', { force: true })

    cy.wait(2000)

})

/\* Access this command in other tests

In below code, first the value is stored as alias using cy.wrap(value).as('sumOfNumbers').

Then, using cy.get('@sumOfNumbers') => It retrieves the value stored in sumOfNumbers

\*/

Cypress.Commands.add('sumTest', (first, second) => {

    const value = first + second

    cy.wrap(value).as('sumOfNumbers')

    return cy.get('@sumOfNumbers').then((value) => {

        return value;

    })

})

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

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

E2e.js

// Hide fetch/XHR requests

const app = window.top;

if (!app.document.head.querySelector("[data-hide-command-log-request]")) {

  const style = app.document.createElement("style");

  style.innerHTML =

    ".command-name-request, .command-name-xhr { display: none }";

  style.setAttribute("data-hide-command-log-request", "");

  app.document.head.appendChild(style);

}

// Alternatively you can use CommonJS syntax:

// require('./commands')

Cypress.on("uncaught:exception", (err, runnable) => {

  // returning false here prevents Cypress from

  // failing the test

  return false;

});

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

Accessing value from one test to test file 🡪[done]

1. Exporting value and import it when your not using feature file
2. Create fixture file and read :

cy.writeFile('cypress/fixtures/storedEmail.json', { email: value });

    cy.log('Value stored in fixture file')

 cy.readFile('cypress/fixtures/storedEmail.json').then((data) => {

    cy.log('Retrieved email from fixture file: ' + data.email);

    cy.wrap(data.email).as('storedEmail'); // Use alias to access in subsequent steps

  });

1st test

let text

describe('Access common functionality using Cypress commands', () => {

    it('Access function and store text value', () => {

        cy.visit(

            "https://ecommerce-playground.lambdatest.io/index.php?route=account/login"

        );

        cy.sum(10, 12).then((value) => {

          cy.get('[id="input-email"]').type(value);

        });

        cy.get('[id="input-email"]').invoke('val').then((value) => {

            text = value;

            cy.log("\*\*\*\* printed text \*\*\*\* ", text)

            return cy.wrap(text)

        })

    })

export { text };

2ed test

import { text } from './Login.cy'

describe(' Access value from other cypress test', () => {

    it('Access type text from different Cypress test', () => {

        cy.visit(

            "https://ecommerce-playground.lambdatest.io/index.php?route=account/login"

        );

        cy.get('[id="input-email"]').type(text)

    })

})

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

(npm i cypress-plugin-api)🡪[done]

describe("API Test using plugin cypress-plugin-api", () => {

  it("test api - GET ", () => {

    cy.api("GET", "https://reqres.in/api/users?page=2").should((response) => {

      expect(response.status).to.eq(200);

    });

  });

  it("POST API testing Using Cypress API Plugin", () => {

    cy.api("POST", "https://reqres.in/api/users", {

      name: "Anshita",

      job: "QA leader",

    } ).should((response) => {

      expect(response.status).to.eq(201);

    });

  });

  it("DELETE API testing Using Cypress API Plugin", () => {

    cy.api("DELETE", "https://reqres.in/api/users/2").should((response) => {

      expect(response.status).to.eq(204);

    });

  });

});

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

# cypress api[done]

describe('Test API in Cypress', () => {

    it('Testing Bad Request', () => {

        cy.request({

            method: 'GET',

            url: 'https://reqres.in/api/unknown/23',

            failOnStatusCode: false

        }).then((response) => {

            expect(response.status).to.be.eq(404)

        })

    })

})

})

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

describe("Blur an element", () => {

    it(".blur() - blur off a DOM element", () => {

        cy.visit(

            "https://naveenautomationlabs.com/opencart/index.php?route=account/register"

        );

        cy.get("#input-firstname").type("Test AB").blur();

        cy.get("#input-lastname").type("Test").focus();

    });

})

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

[done]

describe('Find broken links on the page', () => {

    it('verify navigation across the pages', () => {

        cy.visit('https://ecommerce-playground.lambdatest.io/index.php?route=common/home')

        let brokenLinks = 0

        let activeLinks = 0

        cy.get('a').each(($link, index) => {

            const href = $link.attr('href')

            if (href) {

                cy.request({ url: href, failOnStatusCode: false }).then((response) => {

                    if (response.status >= 400) {

                        cy.log(`\*\*\*\* link  ${index + 1} is Broken Link \*\*\* ${href} `)

                        brokenLinks++

                    }

                    else {

                        cy.log(`\*\*\* link  ${index + 1} is Active Link \*\*\*`)

                        activeLinks++

                    }

                })

            }

        }).then(($links) => {

            const totalLinks = $links.length

            cy.log(`\*\*\*\* total links \*\*\*\* ${totalLinks}`)

            cy.log(`\*\*\*\* broken links \*\*\*\* ${brokenLinks}`)

            cy.log(`\*\*\*\* active links \*\*\*\* ${activeLinks}`)

        })

    })

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

describe("Cypress Exception Handling", () => {

  it("Fail on status code by opening url in browser", () => {

    cy.visit(

      "https://ecommerce-playground.lambdatest.io/index.php?route=account/login/1",

      { failOnStatusCode: false }

    );

  });

  it("Fail on status code by calling api", () => {

    cy.request({

      url: "https://ecommerce-playground.lambdatest.io/index.php?route=account/login/1",

      failOnStatusCode: false,

    });

  });

  it("Test Failure when trying to find incorrect locator- error Message", () => {

    cy.on("fail", (err, runnable) => {

      console.log(err.message);

      return false;

    });

    cy.visit(

      "https://ecommerce-playground.lambdatest.io/index.php?route=account/login"

    );

    cy.get('[id="input-email"]').type("lambdatest.Cypress@disposable.com");

    cy.get('[id="input-password"]').type("Cypress1234!!");

    cy.get('[value="Login"]').click();

    cy.get(".error-message").should("be.visible");

    cy.log("test");

    cy.get('[name="search"]').eq(0).type("Macbook");

  });

  it("Test Failure when trying to find incorrect locator - Password", () => {

    cy.on("fail", (err, runnable) => {

      console.log(err.message);

      return false;

    });

    cy.visit(

      "https://ecommerce-playground.lambdatest.io/index.php?route=account/login"

    );

    cy.get('[id="input-email"]').type("lambdatest.Cypress@disposable.com");

    cy.get('[id="input-password1"]').type("Cypress1234!!");

  });

})

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

Disable test in cypress 🡪 .skip

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

describe(' Drag and Drop', () => {

    const capitals = ["Stockholm", "Oslo", "Seoul", "Rome", "Madrid", "Copenhagen", "Washington"];

    const countries = ["Sweden", "Norway", "South Korea", "Italy", "Spain", "Denmark", "United States"];

    beforeEach(() => {

        cy.visit('http://dhtmlgoodies.com/scripts/drag-drop-custom/demo-drag-drop-3.html')

    })

    function moveCapitalToCountry(dragElement, dropElement) {

        cy.get('div[id^=box]:contains(' + dragElement + ')').trigger('mousedown', { which: 1 })

        cy.wait(100)

        cy.get('div[id^=box]:contains(' + dropElement + ')').trigger('mousemove').trigger('mouseup', { force: true })

    }

    function moveBackCapital(dragElement) {

        cy.get('div[id^=box].dragableBox:contains(' + dragElement + ')').trigger('mousedown', { which: 1 })

        cy.wait(100)

        cy.get('div#dropContent').invoke('css', 'height', '200px').trigger('mousemove').trigger('mouseup', { force: true })

    }

    function verifyColor(dragElement) {

        cy.get('div[id^=box].dragableBox:contains(' + dragElement + ')').should('be.visible')

            .should('have.css', 'background-color', 'rgb(0, 255, 0)')

    }

    it('TC1 || Positive Use Case|| Drag all the capitals to the respective country section', () => {

        countries.forEach((country, index) => {

            const capital = capitals[index];

            moveCapitalToCountry(capital, country);

        });

        capitals.forEach(capital => {

            verifyColor(capital)

            moveBackCapital(capital);

        })

    })

    it('TC2 || Negative Use Case|| Drag all the capitals to the wrong countries', () => {

        moveCapitalToCountry("Stockholm", "Denmark")

        moveCapitalToCountry("Rome", "Norway")

        moveCapitalToCountry("Oslo", "Italy")

        moveCapitalToCountry("Copenhagen", "United States")

        moveCapitalToCountry("Madrid", "South Korea")

        moveCapitalToCountry("Washington", "Spain")

        moveCapitalToCountry("Seoul", "Sweden")

        capitals.forEach(capital => {

            moveBackCapital(capital);

        })

    })

    it('TC3 || Combination of Positive and Negative Use Case', () => {

        moveCapitalToCountry("Stockholm", "Sweden")

        moveCapitalToCountry("Rome", "Italy")

        moveCapitalToCountry("Oslo", "Norway")

        moveCapitalToCountry("Copenhagen", "United States")

        moveCapitalToCountry("Madrid", "Spain")

        moveCapitalToCountry("Washington", "South Korea")

        moveCapitalToCountry("Seoul", "Denmark")

        capitals.forEach(capital => {

            moveBackCapital(capital);

        })

    })

})/////////////////////////////////////////////////////

[done]

describe(" Read csv file using cy.readFile", () => {

    it('Read csv', () => {

        cy.readFile('cypress/fixtures/test.csv').then((data) => {

            cy.log("data === " + data + " rows" + data[0][1])

        })

    })

})

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

//Ref : https://docs.cypress.io/api/commands/selectfile

describe("Upload File Request", () => {

  it("TC1 - No Input Field || upload File by changing action", () => {

    cy.visit("https://www.ilovepdf.com/pdf\_to\_word");

    cy.get("a#pickfiles").selectFile("test.pdf", { action: "drag-drop" });

  });

  it.skip("TC2- Error Case- No Input Field|| upload File without changing action", () => {

    cy.visit("https://www.ilovepdf.com/pdf\_to\_word");

    cy.contains("Select PDF file").selectFile("test.pdf");

  });

  it.skip("TC3- Upload File with input field", () => {

    cy.visit("https://demoqa.com/upload-download");

    cy.get("input#uploadFile").selectFile("test.pdf");

  });

});

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

Monky testing

import {

  createHorde,

  gremlins,

  species,

  strategies,

  mogwais,

  randomizer,

} from "gremlins.js";

const seed = "formFiller";

describe("Form filler", () => {

  let horde;

  beforeEach(() => {

    return cy.visit("https://classic.freecrm.com/register/").then(() =>

      cy.window().then((pageWindow) => {

        horde = createHorde({

          window: pageWindow,

          species: [

            species.clicker(),

            species.toucher(),

            species.formFiller(),

            species.scroller(),

            species.typer(),

          ],

          mogwais: [mogwais.alert(), mogwais.fps(), mogwais.gizmo()],

          strategies: [

            strategies.distribution(),

            strategies.allTogether({ nb: 10000 }),

            strategies.bySpecies({ nb: 10 }),

          ],

          randomizer: new Chance(seed),

        });

      })

    );

  });

  it("should fill text input with seeded value", () => {

    return cy.wrap(horde.unleash());

  });

});

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

[doone]

describe(" Mouse over", () => {

  /\*

  To use realHiver Command, install below plugin

  npm install cypress-real-events

  Add import "cypress-real-events" in support>e2e.js (cypress above 10)

  \*/

  it("TC1 - Mouse Over: Using Plugin realEvent ", () => {

    cy.visit("https://www.bigbasket.com/");

    cy.get(".icon.svg-header.svg-arrow-down-hover.svg-arrow-down-hover-dim"

    ).realHover("mouse");

    cy.get("a:contains('Beverages')").realHover("mouse");

  });

  it(" TC2 - Mouse Over: Using plugin realEvent", () => {

    cy.visit("https://naveenautomationlabs.com/opencart/");

    cy.get("li.dropdown>a:contains('Desktops')").realHover("mouse");

    cy.get("a:contains('PC (0)')").click();

  });

  it("TC3 - Mouse Over: Using Cypress inbuilt Command", () => {

    cy.visit("https://www.tradeling.com/ae-en");

    cy.get("[data-testid='all\_categories\_nav\_bar']").trigger("mouseover");

  });

});

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

[done ]

describe("Multi Origin in Cypress", () => {

  it("Multi Origin test using cy.origin() ", () => {

    cy.visit('https://www.magicbricks.com/')

    cy.get('a[title="Twitter"]').invoke('attr', 'target', '\_self').click({ force: true })

    cy.origin('https://twitter.com', () => {

      cy.contains('comJoined September 2009').should('be.visible')

    })

  });

  it("Error Case: When cy.origin is not passed", () => {

    cy.visit('https://www.magicbricks.com/')

    cy.get('a[title="Twitter"]').invoke('attr', 'target', '\_self').click({ force: true })

    cy.contains('comJoined September 2009').should('be.visible')

  });

})

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

[done]

describe("Pagination Test", () => {

  let numberOfPages;

  it("Pagination Example ", () => {

    cy.visit(

      "https://www.lambdatest.com/selenium-playground/table-pagination-demo"

    );

    cy.get("li#prev").as("next");

    cy.get("div.pagination-container>nav>ul")

      .find(">li")

      .then(($value) => {

        numberOfPages = $value.length;

        cy.log("page count" + numberOfPages);

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

          cy.get("table#table-id>tbody>tr>td")

            .eq(1)

            .then(($value) => {

              if ($value.text == "Stuart Johnston") {

                cy.log(" Name Found ");

                cy.log("page is " + i);

              } else {

                cy.log(" Name not Found ");

                cy.log(" Moving to the next page");

                cy.get("@next").click({ force: true });

              }

            });

        }

      });

  });

});

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

describe("Enter the text without passing .click()", () => {

  beforeEach(() => {

    cy.visit(

      "https://naveenautomationlabs.com/opencart/index.php?route=account/register"

    );

  });

  const dataJson = require("../fixtures/testData.json");

  const input\_firstName = dataJson.firstName;

  const input\_lastName = dataJson.lastName;

  const input\_mobileNumber = dataJson.mobileNumber;

  const input\_password = dataJson.password;

  /\*

  Passing the input value from json file,

   passing email as random generator ,

   used .check() to select the checkbox

   used .tab() to enter tab ..

     Installed plugin => npm install -D cypress-plugin-tab

     for using tab as cypress does not have direct support for tab

  \*/

  it("Input values in the textbox and click on submit ", () => {

    cy.get("input#input-firstname")

      .type(input\_firstName)

      .tab()

      .type(input\_lastName, { log: false })

      .tab()

      .type("abc123" + Math.floor(Math.random() \* 10 + 1) + "@disposable.com")

      .tab()

      .type(input\_mobileNumber)

      .tab()

      .type(input\_password)

      .tab()

      .type(input\_password)

      .tab()

      .tab()

      .tab()

      .tab()

      .check()

      .tab()

      .click();

    cy.log("browser version ", Cypress.browser.version);

  });

});

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

describe("Scroll into view", () => {

  it("Infinite - scroll into view", () => {

    cy.visit("https://www.swiggy.com/");

    cy.get("#location").type("Gurgaon");

    cy.contains("Gurgaon, Haryana, India").click();

    cy.scrollTo("bottom");

    cy.contains("Pune").should("be.visible");

  });

});

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

import { onlyOn, skipOn } from '@cypress/skip-test'

describe("Conditionally disable test in Cypress using plugin", () => {

    it("Only on Chrome, Open URL and add product to cart", () => {

        cy.onlyOn('chrome')

        cy.visit(

            "https://bstackdemo.com/")

        cy.get('.shelf-item\_\_buy-btn').first().click();

    });

    it("Only on Firefox, Open URL and add product to cart", () => {

        cy.onlyOn('firefox')

        cy.visit(

            "https://bstackdemo.com/")

        cy.get('.shelf-item\_\_buy-btn').first().click();

    });

    it("Skip on Window, Open URL and add product to cart", () => {

        cy.skipOn('windows')

        cy.visit(

            "https://bstackdemo.com/")

        cy.get('.shelf-item\_\_buy-btn').first().click();

    });

    it("Skip on MAC, Open URL and add product to cart", () => {

        cy.skipOn('mac')

        cy.visit(

            "https://bstackdemo.com/")

        cy.get('.shelf-item\_\_buy-btn').first().click();

    });

    it("Skip on Windows, Run only on Mac and Browser Chrome", () => {

        skipOn('windows')

        //  cy.onlyOn("mac").onlyOn('chrome')

        cy.visit(

            "https://bstackdemo.com/")

        cy.get('.shelf-item\_\_buy-btn').first().click();

    });

    it("Run on Firefox", () => {

        onlyOn("firefox")

        cy.visit(

            "https://bstackdemo.com/")

        cy.get('.shelf-item\_\_buy-btn').first().click();

    });

    it("Skip on Firefox, Run only on Mac and Browser Chrome", () => {

        skipOn('firefox')

        cy.onlyOn("mac").onlyOn('chrome')

        cy.visit(

            "https://bstackdemo.com/")

        cy.get('.shelf-item\_\_buy-btn').first().click();

    });

});

describe('Verify Tool Tip text in Cypress', () => {

    it('Test Case 1 - Tool Tip Text', () => {

        cy.visit('https://practice.expandtesting.com/tooltips')

        cy.get('#btn1').trigger('mouseover')

        cy.get('.tooltip-inner').should('be.visible').and('have.text', 'Tooltip on top')

1

    })

    it('TC2 - Tool Tip Text', () => {

        cy.visit('https://open.spotify.com/')

        cy.get('[aria-label="Collapse Your Library"]').trigger('mouseover')

        cy.get('#hover-or-focus-tooltip').should('be.visible')

            .and('have.text', "Collapse Your Library")

    })

})

Api Testing

Custom commands

Cypress.Commands.add('getAPI', (pathParam) => {

    cy.request({

        method: 'GET',

        url: '/' + pathParam,

        headers: {

            Authorization: 'Bearer 96096fd0358d8c24b76ba3e3e415aae7102a4452fdf0ad819c070462c474b70b'

        }

    })

})

Cypress.Commands.add('postAPI', (payload) => {

    cy.request({

        method: 'POST',

        url: '/',

        headers: {

            Authorization: 'Bearer 96096fd0358d8c24b76ba3e3e415aae7102a4452fdf0ad819c070462c474b70b'

        },

        body: payload

    })

})

Cypress.Commands.add('putAPI', (pathParam) => {

    cy.request({

        method: 'PUT',

        url: '/' + pathParam,

        headers: {

            Authorization: 'Bearer 96096fd0358d8c24b76ba3e3e415aae7102a4452fdf0ad819c070462c474b70b'

        },

    })

})

Cypress.Commands.add('deleteAPI', (pathParam) => {

    cy.request({

        method: 'DELETE',

        url: '/' + pathParam,

        headers: {

            Authorization: 'Bearer 96096fd0358d8c24b76ba3e3e415aae7102a4452fdf0ad819c070462c474b70b'

        },

describe(' API Automation In Cypress', () => {

    it(' Using Custom Commands', () => {

        let payload = {

            "name": "AB Automation Hub",

            "email": "ABAutomationHub1021@dispostable.com",

            "gender": "female",

            "status": "active"

        }

        cy.postAPI(payload).then((response) => {

            expect(response.status).to.be.equal(201)

            let userId = response.body.id

            cy.getAPI(userId).then((response) => {

                expect(response.status).to.be.equal(200)

            })

            cy.putAPI(userId).then((response) => {

                expect(response.status).to.be.equal(200)

            })

            cy.deleteAPI(userId).then((response) => {

                expect(response.status).to.be.equal(204)

            })

        })

    })

})

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

Examples

1.

 it('Get User1', () => {

        cy.request({

            method: 'GET',

            url: 'https://gorest.co.in/public/v2/users/4420383',

            headers: {

                Authorization: "Bearer 96096fd0358d8c24b76ba3e3e415aae7102a4452fdf0ad819c070462c474b70b"

            }

        })

            .then((response) => {

                cy.log(JSON.stringify(response))

                expect(response.status).to.equal(200)

                expect(response.body.id).to.equal(4420383)

            })

    })

2.

describe(' POST CALL In Cypress', () => {

    function generateRandomEmail() {

        const randomString = Math.random().toString(36).substring(2, 10)

        const email = randomString + "@dispostable.com"

        return email

    }

   it(' POST CALL - Fixtures', () => {

        cy.fixture('users').then((responseObject) => {

            responseObject.email = generateRandomEmail()

            cy.request({

                method: 'POST',

                url: 'https://gorest.co.in/public/v2/users',

                headers: {

                    Authorization: 'Bearer 96096fd0358d8c24b76ba3e3e415aae7102a4452fdf0ad819c070462c474b70b'

                },

                body: responseObject

            }).then((response) => {

                expect(response.status).to.equal(201)

                expect(response.body).has.property("name", "AB Test 01")

                expect(response.body).has.property("gender", "female")

                expect(response.body).has.property("status", "active")

                expect(response.body.id).to.not.be.null

            })

        })

    })

3. Api chaining

    it(' End to End Flow', () => {

        users.email = "randoomm123ab@dispostable.com"

        cy.request({

            method: 'POST',

            url: 'https://gorest.co.in/public/v2/users',

            headers: {

                Authorization: 'Bearer 96096fd0358d8c24b76ba3e3e415aae7102a4452fdf0ad819c070462c474b70b'

            },

            body: users

        }).then((response) => {

            let id = response.body.id

            cy.request({

                method: 'PUT',

                url: 'https://gorest.co.in/public/v2/users/' + id,

                headers: {

                    Authorization: 'Bearer 96096fd0358d8c24b76ba3e3e415aae7102a4452fdf0ad819c070462c474b70b'

                },

                body: updateUSers

            })

            .then((response)=>{

                expect(response.status).to.be.equal(200)

            })

            cy.request({

                method:'GET',

                url: 'https://gorest.co.in/public/v2/users/' + id,

                headers: {

                    Authorization: 'Bearer 96096fd0358d8c24b76ba3e3e415aae7102a4452fdf0ad819c070462c474b70b'

                },

            })

            .then((response)=>{

                expect(response.status).to.be.equal(200)

                expect(response.body).has.property('name', updateUSers.name)

            })

        })

**When to Use failOnStatusCode: false:**

handles error responses

errors are handled gracefully by the application.

1. **Testing Error Scenarios:**
   * If you are intentionally testing how your application handles error responses (like 404 Not Found, 500 Internal Server Error, etc.), you'll need to set failOnStatusCode: false to avoid the test from failing due to the non-2xx status code.
2. **Handling Non-Critical Errors:**
   * In some scenarios, a request might fail (e.g., return a 404 or 500), but the overall test should continue or assert based on that failure. This is common in testing fallback mechanisms or ensuring that certain errors are handled gracefully by the application.
3. **API Testing with Multiple Expected Status Codes:**
   * When testing an API endpoint that could legitimately return different status codes based on different conditions (e.g., 200 for success, 404 for not found, etc.), you might use failOnStatusCode: false to check the status code dynamically in your assertions.

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

**# Cypress\_CrossBrowserTesting**

 "lambdatest-cypress-cli": "^3.0.10"

npm install -g lambdatest-cypress-cli

The lambdatest-cypress-cli is a command-line interface (CLI) tool provided by LambdaTest, which allows you to run your Cypress tests on the LambdaTest cloud platform. LambdaTest is a cloud-based cross-browser testing platform that provides a scalable and reliable infrastructure to run automated tests on various browser and OS combinations.

**Key Features of lambdatest-cypress-cli:**

1. **Cross-Browser Testing:**
   * Run your Cypress tests across multiple browsers and operating systems available on the LambdaTest platform.
2. **Parallel Test Execution:**
   * Execute your Cypress tests in parallel across multiple browser and OS combinations, speeding up your test suite's execution time.
3. **Real-Time Logs and Reports:**
   * Get real-time test execution logs and access detailed test reports on the LambdaTest dashboard.
4. **Integrations:**
   * Integrate with various CI/CD tools like Jenkins, GitHub Actions, CircleCI, etc., to run Cypress tests as part of your continuous integration and delivery pipelines.
5. **Easy Configuration:**
   * Configure your test runs by specifying browser configurations, test specs, and other options through a simple YAML configuration file or directly via the CLI.

Lambdatest-config.json file:

{

   "lambdatest\_auth": {

   "username": "anshita.bhasin",

   "access\_key": '<key>'

   },

   "browsers": [

      {

         "browser": "Chrome",

         "platform": "Windows 10",

         "versions": [

            "latest-1"

         ]

      },

      {

         "browser": "Firefox",

         "platform": "Windows 10",

         "versions": [

            "latest-1"

         ]

      }

   ],

   "run\_settings": {

      "config\_file": "cypress.config.js",

      "reporter\_config\_file": "base\_reporter\_config.json",

      "build\_name": "Cypress cross browser testing using lambdatest",

      "parallels": 1,

      "specs": "./cypress/e2e/homePage.cy.js",

      "ignore\_files": "",

      "network": false,

      "headless": false,

      "npm\_dependencies": {

         "cypress": "12.6.0"

      }

   },

   "tunnel\_settings": {

      "tunnel": false,

      "tunnel\_name": null

   }

}

Lambda-run.json file:

{

   "build\_id": "11569451",

   "session\_id": "edfff083-40a5-46c6-b88c-34321326b34e"

}

Package.json

 "scripts": {

    "LT-test": "lambdatest-cypress run"

  },

Custom-support-js file:

const addContext = require('mochawesome/addContext')

// NOTE: import this file in cypress/support/e2e.js

// import '../../custom\_support\_file';

// require('../../custom\_support\_file');

Cypress.on('test:after:run', (test, runnable) => {

    if (test.state === 'failed') {

      let item = runnable

      const nameParts = [runnable.title]

      // Iterate through all parents and grab the titles

      while (item.parent) {

        nameParts.unshift(item.parent.title)

        item = item.parent

      }

      if(runnable.hookName) {

        nameParts.push(`${runnable.hookName} hook`)

      }

      const fullTestName = nameParts

              .filter(Boolean)

              .join(' -- ')           // this is how cypress joins the test title fragments

      const imageUrl = `${fullTestName} (failed).png`

      addContext({ test }, imageUrl)

    }

  })

Base-report-config json file:

{

   "reporterEnabled": "mochawesome",

   "mochawesomeReporterOptions": {

      "reportDir": "cypress/results/mochawesome",

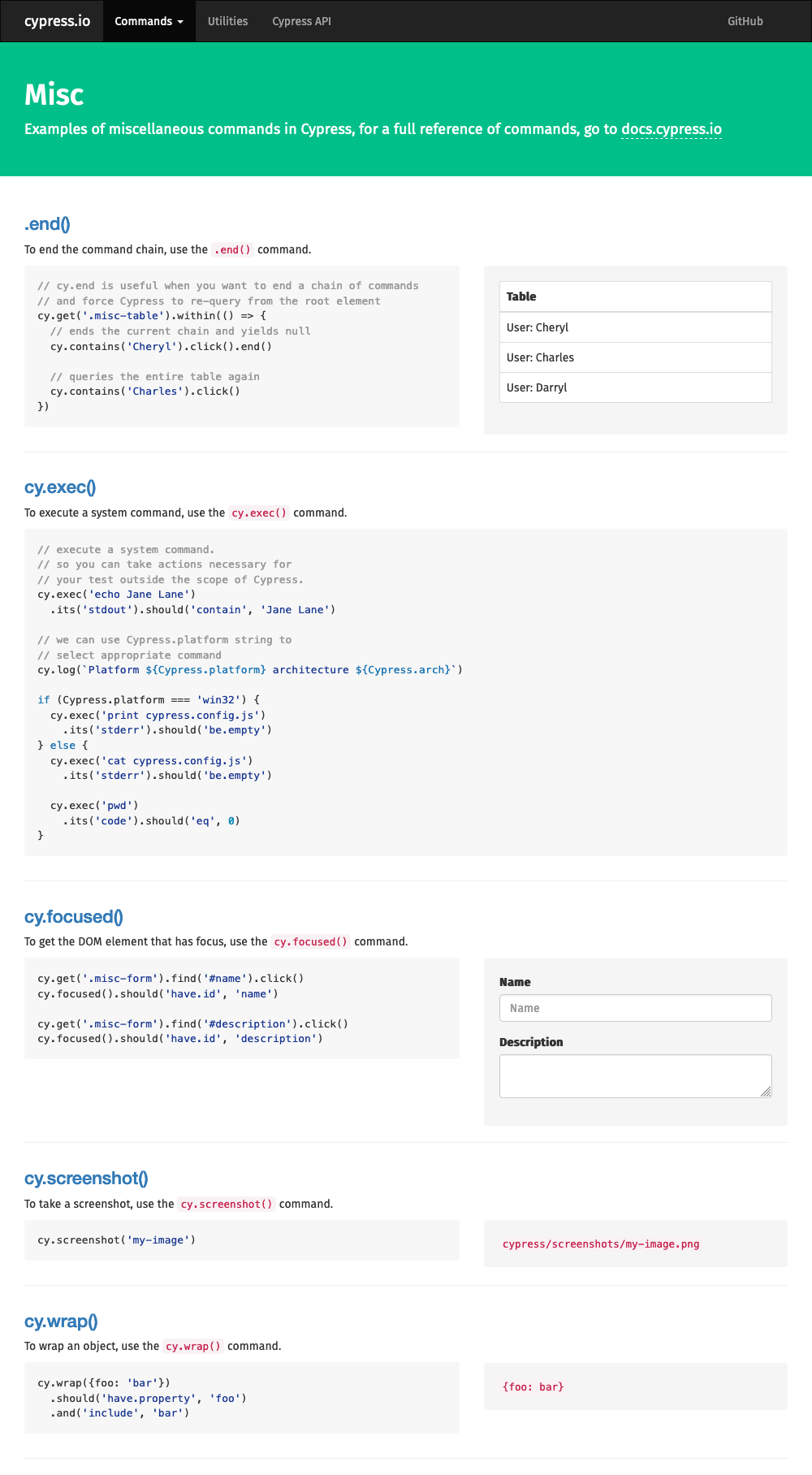
      "overwrite": true,

      "html": true,

      "json": true

   }

}



describe("Testing on different browsers", () => {

  it("Open test in Chrome browser", { browser: "chrome" }, () => {

    cy.visit("https://www.lambdatest.com/");

    cy.log("-- Chrome Browser---");

  });

  it("Open test in Firefox browser", { browser: "firefox" }, () => {

    cy.visit("https://www.lambdatest.com/");

    cy.log("-- Firefox Browser---");

  });

  it("Open test in Electron browser", { browser: "electron" }, () => {

    cy.visit("https://www.lambdatest.com/");

    cy.log("-- Electron Browser---");

  });

});

#################### github action

**# Install Cypress (Node version should be 12 and above)**

1. npm init

2. npm install cypress --save-dev

**# Run Cypress Test Locally**

1. npx cypress open

**# Configure and Run Cypress Test Case On LambdaTest Platform**

1. npm install -g lambdatest-cypress-cli

2. lambdatest-cypress init

3. lambdatest-cypress run

Package.json

{

  "name": "cypress\_github-actions-example",

  "version": "1.0.0",

  "description": "test automation using cypress",

  "main": "index.js",

  "scripts": {

   "test:firefox": "npx cypress run --browser firefox",

    "test-browser-chrome": "npx cypress run --browser chrome",

    "test": "npx cypress run",

    "LT-test": "lambdatest-cypress run"

  },

  "repository": {

    "type": "git",

    "url": "https://github.com/Anshita-Bhasin/Cypress\_Github-Actions.git"

  },

  "author": "Anshita Bhasin",

  "license": "ISC",

  "devDependencies": {

    "cypress": "^12.5.1"

  },

  "dependencies": {

    "lambdatest-cypress-cli": "^3.0.7"

  }

Cypress.on("fail", (e, runnable) => {

  console.log("error", e);

  console.log("runnable", runnable);

  console.log("message", e.message);

  if (

    e.name === "AssertionError" &&

    !e.message.includes(

      "Timed out retrying after 4000ms: Expected to find element: `.error-message`, but never found it."

    )

  ) {

    throw e;

  }

});

Cypress.on("uncaught:exception", (e, runnable) => {

  console.log("error", e);

  console.log("error message", e.message);

  console.log("runnable", runnable);

  if (e.message.includes("Things went bad")) {

    return false;

  }

});

//Hide XHR Request

const app = window.top;

if (!app.document.head.querySelector("[data-hide-command-log-request]")) {

  const style = app.document.createElement("style");

  style.innerHTML =

    ".command-name-request, .command-name-xhr { display: none }";

  style.setAttribute("data-hide-command-log-request", "");

  app.document.head.appendChild(style);

}

////# cypress exception handlling

describe("Cypress Exception Handling", () => {

  it("Fail on status code by opening url in browser", () => {

    cy.visit(

      "https://ecommerce-playground.lambdatest.io/index.php?route=account/login/1",

      { failOnStatusCode: false }

    );

  });

  it("Fail on status code by calling api", () => {

    cy.request({

      url: "https://ecommerce-playground.lambdatest.io/index.php?route=account/login/1",

      failOnStatusCode: false,

    });

  });

  it("Test Failure when trying to find incorrect locator- error Message", () => {

    cy.visit(

      "https://ecommerce-playground.lambdatest.io/index.php?route=account/login"

    );

    cy.get('[id="input-email"]').type("lambdatest.Cypress@disposable.com");

    cy.get('[id="input-password"]').type("Cypress1234!!");

    cy.get('[value="Login"]').click();

    cy.get(".error-message").should("be.visible");

    cy.log("test");

    cy.get('[name="search"]').eq(0).type("Macbook");

  });

  it("Test Failure when trying to find incorrect locator - Password", () => {

    cy.visit(

      "https://ecommerce-playground.lambdatest.io/index.php?route=account/login"

    );

    cy.get('[id="input-email"]').type("lambdatest.Cypress@disposable.com");

    //cy.get('[id="input-password1"]').type("Cypress1234!!");

  });

  it("Uncaught Exception - Due to application error", () => {

    cy.visit("index.html");

    cy.get("button#error").click();

    cy.wait(1000);

  });

});

View port :

Lambdatest-config json

{

  "lambdatest\_auth": {

    "username": "<user.name>",

    "access\_key": "<password>"

  },

  "browsers": [

    {

      "browser": "Chrome",

      "platform": "Windows 10",

      "versions": ["latest-1"]

    },

    {

      "browser": "Firefox",

      "platform": "Windows 10",

      "versions": ["latest-1"]

    }

  ],

  "run\_settings": {

    "config\_file": "cypress.config.js",

    "reporter\_config\_file": "base\_reporter\_config.json",

    "build\_name": "Cypress Viewport Testing",

    "parallels": 1,

    "specs": "./cypress/e2e/loginTest.cy.js",

    "ignore\_files": "",

    "network": false,

    "headless": false,

    "npm\_dependencies": {

      "cypress": "12.16.0"

    },

    "cypress\_settings": "--config viewportWidth=660,viewportHeight=880;"

  }

}

Ex

describe("Viewport Testing using defined presets", () => {

    const presets =

        ['macbook-15', 'samsung-note9', 'ipad-mini', 'iphone-xr']

        ;

    it("Passing single preset", () => {

        cy.visit("https://ecommerce-playground.lambdatest.io/index.php?route=account/login");

        cy.viewport('ipad-2');

        cy.get('[id="input-email"]').type("lambdatest.Cypress@disposable.com");

        cy.get('[id="input-password"]').type("Cypress123!!");

        cy.get('[type="submit"]').eq(0).click({ force: true });

    });

    it(`Passing multiple presets`, () => {

        cy.visit("https://ecommerce-playground.lambdatest.io/index.php?route=common/home");

        presets.forEach((preset) => {

            cy.viewport(preset)

            cy.get('input[name="search"]').eq(0).type("Macbook");

        })

    });

});

Ex

describe("Viewport Testing using customized Widhth And Height", () => {

    const viewports = [

        { width: 1280, height: 720 }, // Desktop

        { width: 320, height: 480 }, // Mobile portrait

        { width: 768, height: 1024 },// Tablet

    ];

    it("Passing single width and height", () => {

        cy.visit("https://ecommerce-playground.lambdatest.io/index.php?route=account/login");

        cy.viewport(360, 890);

        cy.get('[id="input-email"]').type("lambdatest.Cypress@disposable.com");

        cy.get('[id="input-password"]').type("Cypress123!!");

        cy.get('[type="submit"]').eq(0).click({ force: true });

    });

    it(`Passing multiple width and height`, () => {

        cy.visit("https://ecommerce-playground.lambdatest.io/index.php?route=common/home");

        viewports.forEach((viewport) => {

            cy.viewport(viewport.width, viewport.height);

            cy.get('input[name="search"]').eq(0).type("Macbook");

        })

    });

});

Ex cy.viewport('iphone-xr', 'landscape');

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

  "dependencies": {

    "cypress-parallel": "^0.13.0"

  }

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

Multi reports config file

{

  "reporterEnabled": "cypress-parallel/json-stream.reporter.js, cypress-parallel/simple-spec.reporter.js"

}

///// page in pom

export class registerPage {

    weblocators = {

        firstName: '#input-firstname',

        lastName: '#input-lastname',

        email: '#input-email',

        telephone: '#input-telephone',

        password: '#input-password',

        passwordConfirm: '#input-confirm',

        policyCheckbox: 'input[type="checkbox"]',

        continue: '.btn.btn-primary'

    }

    openURL() {

        cy.visit(Cypress.env('URL'))

    }

    enterFirstName(FName) {

        cy.get(this.weblocators.firstName).type(FName)

    }

    enterlastName(LName) {

        cy.get(this.weblocators.lastName).type(LName)

    }

    enterEmail(email) {

        cy.get(this.weblocators.email).type(email)

    }

    enterTelephone(phoneNo) {

        cy.get(this.weblocators.telephone).type(phoneNo)

    }

    enterPassword(password) {

        cy.get(this.weblocators.password).type(password)

        cy.get(this.weblocators.passwordConfirm).type(password)

    }

    selectCheckbox() {

        cy.get(this.weblocators.policyCheckbox).check()

    }

    clickOnContinue() {

        cy.get(this.weblocators.continue).click()

    }

}

/////////////// testcase

import { registerPage } from "../../pages/registerPage"

const registerObj = new registerPage()

import registerData from '../../fixtures/registerData.json'

describe(' test automation', () => {

    it('register flow', () => {

        registerObj.openURL()

        registerObj.enterFirstName(registerData.firstName)

        registerObj.enterlastName(registerData.lastName)

        registerObj.enterEmail(registerData.email)

        registerObj.enterTelephone(registerData.telephone)

        registerObj.enterPassword(registerData.password)

        registerObj.selectCheckbox()

        registerObj.clickOnContinue()

    })

})

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

import { faker } from '@faker-js/faker';

const csv = require('neat-csv')

let regData

describe('Registration Form', () => {

    before(() => {

        cy.fixture("users.csv")

            .then(csv)

            .then((data) => {

                regData = data

            })

    })

    it(' Fill Reg Form for different users ', () => {

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

            cy.visit('')

            cy.get('#input-firstname').type(regData[i]['firstName']);

            cy.get('#input-lastname').type(regData[i]['lastName']);

            cy.get('#input-email').type(faker.datatype.uuid() + regData[i]['Email']);

            cy.get('#input-telephone').type(regData[i]['Telephone']);

            cy.get('#input-password').type(regData[i]['Password']);

            cy.get('#input-confirm').type(regData[i]['Password']);

            cy.get('input[type="checkbox"]').check()

            cy.get('input.btn.btn-primary').click()

            cy.get('a[data-toggle="dropdown"]').first().click()

            cy.contains('Logout').click()

        }

    })

})

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

The "cypress-split": "^1.3.5" is a versioned dependency in your Cypress project that allows you to split your test suite into smaller chunks and run them in parallel across multiple CI jobs. This is particularly useful for speeding up test execution in continuous integration (CI) environments, where you may want to distribute tests across several machines or containers to optimize build times.

**Key Features of cypress-split:**

1. **Parallel Test Execution:**
   * Automatically splits your Cypress test suite into multiple groups, allowing them to be executed in parallel across different CI jobs.
2. **Dynamic Splitting:**
   * The splitting is dynamic, meaning the tool can balance the number of tests in each group to ensure that the execution time is evenly distributed, avoiding one group taking significantly longer than the others.
3. **CI Integration:**
   * Works well with popular CI tools like GitHub Actions, CircleCI, Travis CI, and others, making it easier to implement parallel test execution within your existing CI pipeline.
4. **Easy Configuration:**
   * Minimal setup is required. You can configure the split strategy via environment variables or directly within your CI configuration files.

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

Lambda config file

{

  "lambdatest\_auth": {

    "username": "anshita.bhasin@gmail.com",

    "access\_key": "yxu3wRJAzjY2rpgWjla3S5kBqwWTryAlfZIWNbdSGURPoXyoxR"

  },

  "browsers": [

    {

      "browser": "Chrome",

      "platform": "Windows 10",

      "versions": ["latest-1"]

    },

    {

      "browser": "Firefox",

      "platform": "Windows 10",

      "versions": ["latest-1"]

    },

    {

        "browser": "Edge",

        "platform": "Windows 10",

        "versions": [

           "latest"

        ]

     },

      {

       "browser": "Firefox",

       "platform": "MacOS Monterey",

       "versions": [

          "latest"

       ]

    },

 {

       "browser": "Chrome",

       "platform": "MacOS Big sur",

       "versions": [

          "latest"

       ]

    }

  ],

  "run\_settings": {

    "config\_file": "cypress.config.js",

    "reporter\_config\_file": "base\_reporter\_config.json",

    "build\_name": "Cypress-Test-With-Docker",

    "parallels": 1,

    "specs": "./cypress/e2e/\*.cy.js",

    "ignore\_files": "",

    "network": false,

    "headless": false,

    "npm\_dependencies": {

      "cypress": "12.9.0"

    }

  }

}