|  |  |
| --- | --- |
| Project- | KredSafe |
| Automation QA | Nilesh Shekdar |
| Tool | Playwright |

**Playwright-**

**Playwright** is an open-source **end-to-end testing framework** developed by Microsoft. It allows you to automate browser actions for web applications across **Chromium, Firefox, and WebKit** — using **JavaScript, TypeScript, Python, C#, or Java**.

It’s ideal for testing UI interactions just like a real user would (clicking buttons, filling forms, navigating pages).

**Objective of Playwright-**

The **main objective of Playwright** is to provide a **powerful, modern, and reliable framework** for **automating browser interactions** — enabling developers and QA engineers to:

* **Test web applications end-to-end**, across multiple browsers and devices
* **Ensure consistent behaviour and UI rendering** in different environments
* **Automate repetitive tasks** in the browser (like filling forms, clicking, navigation, etc.)
* **Catch bugs earlier** in the development cycle through **automated testing in CI/CD**

## Role based Use of Playwright-

| **Role** | **Use Case** |
| --- | --- |
| QA Engineer | Automating UI/functional tests |
| Front-End Developer | Testing UI components, forms, interactions |
| Full-Stack Developer | End-to-end user journey validation |
| DevOps Engineer | CI/CD integration and pipeline validation |
| Manual Tester | Starting with simple automated test scenarios |
| Open Source Maintainer | Cross-browser testing for libraries and tools |
|  |  |

**Software Requirements for Playwright-**

| **Component** | **Requirement / Version** |
| --- | --- |
| **Node.js** | Required: **v14+** (LTS recommended – v18 or v20) |
| **npm** or **yarn** | Needed to install Playwright packages |
| **Playwright Library** | npm install --save-dev playwright |
| **Test Runner (optional)** | Built-in @playwright/test OR Jest/Cucumber/etc. |
| **Browsers** | Installed via npx playwright install |
| **OS dependencies** (Linux only) | Use npx playwright install-deps to install required libs |
| **Optional Tools-**   | **Tool** | **Use Case** | | --- | --- | | **VS Code / Any IDE** | Code writing and debugging | | **Allure CLI** | For detailed reports: npm i -g allure-commandline | | **Docker** | Run Playwright in containerized environments | | **CI/CD tool** (GitHub Actions, Jenkins, etc.) | Automate tests on code push | |  |

## Minimum Working Environment -

* OS: Windows 10 / Ubuntu 20.04 / macOS
* Node.js: v18 LTS
* RAM: 4–8 GB
* Disk: 2 GB free
* Installed Browsers: Chromium, Firefox, WebKit via Playwright

Step-by-Step Guide to Playwright-

### **Step 1: Install Node.js**

Make sure you have **Node.js** installed (version 14 or above).

To check:

node -v

npm -v

If not installed, download from: <https://nodejs.org/>

### **Step 2: Set Up a New Project**

mkdir playwright-demo

cd playwright-demo

npm init -y

This creates a basic package.json.

### **Step 3: Install Playwright**

npm install -D @playwright/test

npx playwright install

This installs:

* Playwright test runner
* Browser binaries (Chromium, Firefox, WebKit)

### **Step 4: Create Your First Test File**

📄 Create a file example.spec.js (or .ts if you're using TypeScript):

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

test('homepage has title', async ({ page }) => {

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

await expect(page).toHaveTitle(/Example Domain/);

});

### **Step 5: Run the Test**

npx playwright test

You’ll see output like:

Running 1 test using 1 worker

example.spec.js: homepage has title

### **Step 6: View the HTML Report (Optional)**

Playwright generates test reports:

npx playwright show-report

This opens an HTML dashboard of test results in your browser.

### **Step 7: (Optional) Configure** playwright.config.js

Create a config file for custom settings:

npx playwright test --init

It generates a playwright.config.js with options for:

* Base URL
* Projects (browsers)
* Timeout, retries, etc.

### **Step 8: Add More Tests**

Create more test files in the tests/ folder:

tests/

└── login.spec.js

└── dashboard.spec.js

### **Step 9: Run Tests in All Browsers**

npx playwright test --project=chromium

npx playwright test --project=firefox

npx playwright test --project=webkit

Or all at once:

npx playwright test

### **Step 10: Integrate with CI/CD (GitHub Actions Example)**

Create .github/workflows/playwright.yml:

name: Playwright Tests

on:

push:

branches: [main]

pull\_request:

jobs:

test:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v3

- name: Setup Node.js

uses: actions/setup-node@v3

with:

node-version: '18'

- name: Install dependencies

run: npm ci

- name: Install browsers

run: npx playwright install --with-deps

- name: Run tests

run: npx playwright test

## Summary: Full Workflow-

1. Install Node.js
2. Create project folder
3. Install Playwright
4. Write test file
5. Run tests
6. View reports
7. Configure settings (optional)
8. Add to CI/CD

**Recommended Folder Structure for Playwright-**

my-playwright-project/

├── tests/ # All your test files live here

│ ├── login.spec.ts

│ ├── dashboard.spec.ts

│ └── ... more tests ...

│

├── playwright.config.ts # Main Playwright configuration file

│

├── package.json # Project metadata and dependencies

├── tsconfig.json # (If using TypeScript)

│

├── test-results/ # Test output: screenshots, traces, videos

│

├── allure-results/ # (optional) For Allure report raw files

├── allure-report/ # (optional) Generated Allure HTML report

│

├── node\_modules/ # Installed dependencies

└── .github/

└── workflows/

└── playwright.yml # ✅ GitHub Actions CI/CD config

**Explanation of Important Folders-**

| **Folder/File** | **Purpose** |
| --- | --- |
| tests/ | All your .spec.ts or .spec.js files (test cases) |
| playwright.config.ts | Playwright settings (timeout, browsers, baseURL, etc.) |
| test-results/ | Automatically created by Playwright to store trace, screenshots, and videos |
| .github/workflows/ | Contains your CI/CD pipeline (e.g., GitHub Actions config) |
| allure-results/ | Created if you use Allure for reporting |
| allure-report/ | HTML report generated from Allure data |

## Example Test File Path

tests/login.spec.ts

You can run it like:

npx playwright test tests/login.spec.ts

Write First Test of Login-

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

test('Verify login logout', async ({ page }) => {

      test.setTimeout(60000); // 60 seconds

  await page.goto('https://dev.kredsafe.net/login');

  await page.locator('//input[@name="email"]').fill('hrd786@yopmail.com');

await page.locator('//input[@name="password"]').fill('Nilesh@2025');

await page.locator('//\*[@id="id\_frm\_submit"]').click();

await page.waitForLoadState('networkidle');

 await page.getByRole('button', { name: 'Verify' }).click();

await page.evaluate(() => {

  const link = Array.from(document.querySelectorAll('a'))

    .find(el => el.textContent?.trim() === 'Dams Tom Dams Tom');

  if (link) link.click();

});

await page.evaluate(() => {

  const link = Array.from(document.querySelectorAll('a'))

    .find(el => el.textContent?.trim() === 'Sign out');

  if (link) link.click();

});});

Change Password-

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

test('Verify Change password', async ({ page }) => {

      test.setTimeout(60000); // 60 seconds

  await page.goto('https://dev.kredsafe.net/login');

  await page.locator('//input[@name="email"]').fill('ts1234@yopmail.com');

await page.locator('//input[@name="password"]').fill('Nilesh@2030');

await page.locator('//\*[@id="id\_frm\_submit"]').click();

  await page.waitForTimeout(5000);

await page.goto('https://dev.kredsafe.net/user/change-password');

  await page.waitForTimeout(5000);

await page.locator('//\*[@id="oldpassword"]').fill('Nilesh@2030');

// Fill new password

await page.locator('//\*[@id="passwordch"]').fill('Nilesh@2025');

// Fill confirm new password

await page.locator('//\*[@id="password\_conf"]').fill('Nilesh@2025');

// Click submit button

await page.locator('//\*[@id="id\_frm\_submit"]').click();

});

Change Password-by Data Driven

import { test } from '@playwright/test';

// Define multiple test cases with only password fields

const testData = [

  {

    oldPassword: 'Nilesh@2030',

    newPassword: 'Nilesh@2025',

    confirmPassword: 'Nilesh@2025'

  },

  {

    oldPassword: 'Nilesh@2025',

    newPassword: 'Nilesh@2030',

    confirmPassword: 'Nilesh@2030'

  }

  // Add more test sets if needed

];

test.describe('Data-driven Change Password (with login)', () => {

  for (const data of testData) {

    test(`Change password from ${data.oldPassword} to ${data.newPassword}`, async ({ page }) => {

    test.setTimeout(60000); // 60 seconds

      // Step 1: Login

      await page.goto('https://dev.kredsafe.net/login');

      await page.locator('//input[@name="email"]').fill('ts1234@yopmail.com');

      await page.locator('//input[@name="password"]').fill('Nilesh@2025');

      await page.locator('//\*[@id="id\_frm\_submit"]').click();

      await page.waitForTimeout(5000);

      // Step 2: Navigate to Change Password page

      await page.goto('https://dev.kredsafe.net/user/change-password');

      await page.waitForTimeout(2000);

      // Step 3: Fill password fields

      await page.locator('//\*[@id="oldpassword"]').fill(data.oldPassword);

      await page.locator('//\*[@id="passwordch"]').fill(data.newPassword);

      await page.locator('//\*[@id="password\_conf"]').fill(data.confirmPassword);

      // Step 4: Submit the form

      await page.locator('//\*[@id="id\_frm\_submit"]').click();

      // Optional: Verify success message

      // await expect(page.locator('text=Password changed successfully')).toBeVisible();

    });

  }

});

Forgot Password-

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

test('Verify forgot password and reset flow', async ({ page, context }) => {

      test.setTimeout(60000); // 60 seconds

  // 1. Go to login page

  await page.goto('https://dev.kredsafe.net/login');

  // 2. Scroll down

  await page.evaluate(() => window.scrollBy(0, window.innerHeight));

  // 3. Click "Forgot Password"

  await page.locator('xpath=//\*[@id="loginForm"]/div[4]/div[2]/a').click();

  await page.waitForTimeout(2000);

  // 4. Fill and submit email

  const emailInput = page.locator('xpath=//\*[@id="email"]');

  const submitButton = page.locator('xpath=//\*[@id="id\_frm\_submit"]');

  await emailInput.waitFor({ state: 'visible', timeout: 10000 });

  await emailInput.fill('swapj@yopmail.com');

  await submitButton.click();

  await page.waitForTimeout(3000);

  // 5. Open Yopmail in new tab

  const yopmailTab = await context.newPage();

  await yopmailTab.goto('https://yopmail.com/en/', { waitUntil: 'domcontentloaded' });

  await yopmailTab.fill('#login', 'swapj@yopmail.com');

  await yopmailTab.click('#refreshbut');

  await yopmailTab.waitForTimeout(2000);

  // 6. Click on email from inbox iframe

  const inboxFrame = yopmailTab.frameLocator('#ifinbox');

  // 7. Click "Click here" link in mail iframe

  const mailFrame = yopmailTab.frameLocator('#ifmail');

  const resetLink = mailFrame.locator('xpath=//\*[@id="mail"]/div/div/table/tbody/tr/td/p[2]/a');

  // 8. Wait for reset password tab to open

  const [resetTab] = await Promise.all([

    context.waitForEvent('page'),

    resetLink.click(),

  ]);

  await resetTab.waitForLoadState('domcontentloaded');

  // 9. Fill new password

  await resetTab.locator('xpath=//\*[@id="passwordL"]').fill('Nilesh@2025');

  await resetTab.locator('xpath=//\*[@id="password\_conf"]').fill('Nilesh@2025');

  await resetTab.locator('xpath=//\*[@id="id\_frm\_submit\_reset\_pwd"]/span').click();

  // 10. Optional wait or assertion

  await resetTab.waitForTimeout(3000);

});

Form Payment-

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

test('Form Packet Payment flow', async ({ page }) => {

    test.setTimeout(60000); // 60 seconds

  await page.goto('https://dev.kredsafe.net/login');

  // Login

  await page.getByRole('textbox', { name: 'E-mail Address \*' }).fill('hrd14566@yopmail.com');

  await page.getByRole('textbox', { name: 'Password \*' }).fill('Nilesh@2025');

  await page.getByRole('button', { name: 'Login' }).click();

  await page.waitForLoadState('load');

    // await page.goto('https://dev.kredsafe.net/user/home');

    // await page.waitForLoadState('load');

const home = page.locator("//span[@class='nav-link menuBars' and @data-widget='pushmenu']");

 await home .waitFor({ state: 'visible' });

 await page.evaluate(el => el.click(), await home  .elementHandle());

 const pkt = page.locator(" //p[text()='Packets']");

 await pkt .waitFor({ state: 'visible' });

 await page.evaluate(el => el.click(), await pkt  .elementHandle());

// const cart = page.locator('//\*[@id="cartIcon"]/a/i');

//  await cart .waitFor({ state: 'visible' });

//  await page.evaluate(el => el.click(), await cart  .elementHandle());

const chk = page.locator('//\*[@id="pkt\_id\_700"]');

 await chk .waitFor({ state: 'visible' });

 await page.evaluate(el => el.click(), await chk  .elementHandle());

   // await page.waitForTimeout(2000);

  const addToCartBtn = page.locator("//button[@title='Add to cart']");

 await addToCartBtn .waitFor({ state: 'visible' });

 await addToCartBtn.scrollIntoViewIfNeeded();

   // await page.waitForTimeout(1000);

await addToCartBtn.click();

  await page.waitForLoadState('load');

   await page.waitForTimeout(1000);

const confirmBtn = page.locator("//button[@class='button confirm-btn']");

  await confirmBtn.waitFor({ state: 'visible' });

await confirmBtn.scrollIntoViewIfNeeded();

await confirmBtn.click();

  console.log("Button clicked");

  await page.waitForLoadState('domcontentloaded');

  // Fill card details

   await page.waitForTimeout(5000);

  await page.locator('//\*[@id="cardNumber"]').waitFor({ state: 'visible' });

     await page.locator('//\*[@id="cardNumber"]').fill('4111111111111111');

  await page.locator('//\*[@id="cardExpiry"]').fill('09/25');

  await page.locator('//\*[@id="cardCvc"]').fill('111');

  // Fill billing info

  await page.locator('//input[@id="billingName"]').fill('hch');

  await page.locator('//input[@id="billingAddressLine1"]').fill('Nagpur');

  await page.locator('//input[@id="billingLocality"]').fill('Nagpur');

  await page.locator('//input[@id="billingPostalCode"]').fill('441108');

  // Select state (3rd option)

  const stateDropdown = page.locator('//select[@id="billingAdministrativeArea"]');

  await stateDropdown.selectOption({ index: 2 }); // 0-based index

  // Scroll to Pay button and click via JS click

  const payButton = page.locator('//div[@class="SubmitButton-IconContainer"]');

  await payButton.waitFor({ state: 'visible' });

  await payButton.scrollIntoViewIfNeeded();

await payButton.click();

   await page.waitForTimeout(5000);

   await page.waitForTimeout(1000);

await page.waitForURL('https://dev.kredsafe.net/user/es-packets', { timeout: 40000 });

  await page.waitForLoadState('load');

});

Form Request-

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

test('Fill and submit facility request form', async ({ page }) => {

      test.setTimeout(60000); // 60 seconds

  const fillField = async (selector, value) => {

    const el = page.locator(selector);

    await expect(el).toBeVisible({ timeout: 10000 });

    await el.fill(value);

  };

  const facilityName = 'Roger';

  const contactPerson = 'Damc';

  const email = 'formtest@yopmail.com';

  const phone = '1233217654';

  const address = 'New York US';

  // Login

  await page.goto('https://dev.kredsafe.net/login');

  await fillField('input[name="email"]', 'hrd786@yopmail.com');

  await fillField('input[name="password"]', 'Nilesh@2025');

  await page.locator('#id\_frm\_submit').click();

  // Navigate to request form

  await page.goto('https://dev.kredsafe.net/user/es\_forms/request-form');

  await page.waitForLoadState('networkidle');

  await page.waitForTimeout(1000); // brief pause if needed

  // Fill facility name via JS

  await page.evaluate((value) => {

    const input = document.querySelector('#facility\_name');

    if (input) input.value = value;

  }, facilityName);

  // Fill contact person with dispatching events

  await page.evaluate((value) => {

    const input = document.querySelector('#contact\_person');

    if (input) {

      input.value = value;

      input.dispatchEvent(new Event('input', { bubbles: true }));

      input.dispatchEvent(new Event('change', { bubbles: true }));

    }

  }, contactPerson);

  // Fill email normally

  await page.waitForSelector('#email', { timeout: 10000 });

  await page.locator('#email').fill(email);

  // Fill phone

  await page.evaluate((value) => {

    const input = document.querySelector('#phone');

    if (input) {

      input.value = value;

      input.dispatchEvent(new Event('input', { bubbles: true }));

      input.dispatchEvent(new Event('change', { bubbles: true }));

    }

  }, phone);

  // Fill address textarea

  await page.evaluate((value) => {

    const textarea = document.querySelector('textarea[name="address"]');

    if (textarea) {

      textarea.value = value;

      textarea.dispatchEvent(new Event('input', { bubbles: true }));

      textarea.dispatchEvent(new Event('change', { bubbles: true }));

    }

  }, address);

await page.evaluate(() => {

  const uploadInput = document.querySelector("input[type='file'][id='template\_ducument'][name='file[]'].form-control");

  if (uploadInput) {

    uploadInput.click();

  }

});

  // Scroll down  to element

  await page.keyboard.press('PageDown');

});

Subscription Notes-

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

test('Subscription cancellation request flow', async ({ page }) => {

      test.setTimeout(60000); // 60 seconds

  await page.goto('https://dev.kredsafe.net/login');

  await page.locator('//input[@name="email"]').fill('hrd786@yopmail.com');

await page.locator('//input[@name="password"]').fill('Nilesh@2025');

await page.locator('//\*[@id="id\_frm\_submit"]').click();

  await page.goto('https://dev.kredsafe.net/user/subscription/dashboard');

await page.waitForLoadState('networkidle');

const cancellation = page.locator('xpath=//a[text()="Cancellation Requests"]');

await page.waitForTimeout(5000);

//await cancellation.click();

await page.evaluate(() => {

  const xpath = '//a[text()="Cancellation Requests"]';

  const result = document.evaluate(xpath, document, null, XPathResult.FIRST\_ORDERED\_NODE\_TYPE, null);

  const element = result.singleNodeValue;

  if (element) element.click();

});

await page.evaluate(() => window.scrollBy(0, 300));

const commentIcon = page.locator('xpath=(//i[@title="follow up"])[2]');

await page.waitForTimeout(5000);

await page.evaluate(() => {

  const xpath = '(//i[@title="follow up"])[2]';

  const result = document.evaluate(xpath, document, null, XPathResult.FIRST\_ORDERED\_NODE\_TYPE, null);

  const element = result.singleNodeValue;

  if (element) element.click();

});

});

Login by Data Driven-

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

const testData = [

  { email: 'user1@yopmail.com', password: 'pass1' },

  { email: 'user2@yopmail.com', password: 'pass2' },

  { email: 'hrd786@yopmail.com', password: 'Nilesh@2025' },

];

test.describe('Login tests - Data Driven', () => {

  testData.forEach(({ email, password }) => {

    test(`Login with ${email}`, async ({ page }) => {

          test.setTimeout(60000); // 60 seconds

      await page.goto('https://dev.kredsafe.net/login');

      // Use clear, Playwright-style selectors instead of XPath where possible

      await page.locator('input[name="email"]').fill(email);

      await page.locator('input[name="password"]').fill(password);

      await page.locator('#id\_frm\_submit').click();

      // Example assertion: check if login redirected to dashboard

      await expect(page).toHaveURL(/dashboard|home|profile/); // Adjust regex as needed

    });

  });

});

Make my Primary Email-

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

const testData = [

  { email: 'user1@yopmail.com', password: 'pass1' },

  { email: 'user2@yopmail.com', password: 'pass2' },

  { email: 'hrd786@yopmail.com', password: 'Nilesh@2025' },

];

test.describe('Login tests - Data Driven', () => {

  testData.forEach(({ email, password }) => {

    test(`Login with ${email}`, async ({ page }) => {

          test.setTimeout(60000); // 60 seconds

      await page.goto('https://dev.kredsafe.net/login');

      // Use clear, Playwright-style selectors instead of XPath where possible

      await page.locator('input[name="email"]').fill(email);

      await page.locator('input[name="password"]').fill(password);

      await page.locator('#id\_frm\_submit').click();

      // Example assertion: check if login redirected to dashboard

      await expect(page).toHaveURL(/dashboard|home|profile/); // Adjust regex as needed

    });

  });

});

Notes –

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

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

      test.setTimeout(60000); // 60 seconds

  await page.goto('https://dev.kredsafe.net/');

  await page.getByRole('textbox', { name: 'E-mail Address \*' }).click();

  await page.getByRole('textbox', { name: 'E-mail Address \*' }).fill('ts1234@yopmail.com');

  await page.getByRole('textbox', { name: 'Password \*' }).click();

  await page.getByRole('textbox', { name: 'Password \*' }).fill('Nilesh@2025');

  await page.getByRole('button', { name: 'Login' }).click();

  await page.waitForLoadState('load');

 await page.goto('https://dev.kredsafe.net/user/comments/view/Document/NDkz');

   await page.waitForLoadState('load');

//await page.getByRole('textbox', { name: 'Message' }).scrollIntoViewIfNeeded();

await page.evaluate(() => {

  const textarea = document.evaluate(

    "//textarea[@id='message\_info']",

    document,

    null,

    XPathResult.FIRST\_ORDERED\_NODE\_TYPE,

    null

  ).singleNodeValue;

  if (textarea) textarea.value = 'Test Notes';

});

await page.evaluate(() => {

  const submitBtn = document.evaluate(

    "//span[normalize-space(text())='Submit']",

    document,

    null,

    XPathResult.FIRST\_ORDERED\_NODE\_TYPE,

    null

  ).singleNodeValue;

  if (submitBtn) submitBtn.click();

});

  });

Online Signup-

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

test('Your test name', async ({ page, context }) => {

      test.setTimeout(60000); // 60 seconds

  await page.goto('https://dev.kredsafe.net');

  // 1. Click on "Register Here" using JS click

  const registerHereLink = page.locator('xpath=//a[@href="https://dev.kredsafe.net/users/register/options"]');

  await page.evaluate((el) => el.click(), await registerHereLink.elementHandle());

  // 2. Wait and click "Register Using Online Application"

  const registerOnlineLink = page.locator('xpath=//a[@href="https://dev.kredsafe.net/guest/verify"]');

  await page.waitForTimeout(5000);

  await registerOnlineLink.click();

  // 3. Fill email field

  const emailField = page.locator('xpath=//input[@class="emailInputText"]');

  await emailField.waitFor({ state: 'visible', timeout: 10000 });

  await emailField.fill('ts097110@yopmail.com');

  // 4. Wait and click submit

  await page.waitForTimeout(3000);

  const submitButton = page.locator('xpath=//\*[@id="id\_frm\_submit\_reg"]');

  await submitButton.scrollIntoViewIfNeeded();

  await submitButton.click();

  // Wait for confirmation or redirection

  await page.waitForTimeout(1000); // ⛔️ FIXED: was `await page.waitForTimeout2000);`

  // 5. Go to Yopmail

  await page.goto('https://yopmail.com', {

    waitUntil: 'domcontentloaded',

    timeout: 30000,

  });

  await page.fill('#login', 'ts097110@yopmail.com');

  await page.click('#refreshbut');

  // 6. Find the iframe containing the email

  const mailFrame = page.frameLocator('#ifmail');

  const registerYopmailButton = mailFrame.locator(

    'xpath=//a[contains(@href, "https://dev.kredsafe.net/register") and @title="Register Now"]'

  );

  await registerYopmailButton.waitFor({ state: 'visible', timeout: 15000 });

  // 7. Click the email link which opens a new tab

const [newPage] = await Promise.all([

  context.waitForEvent('page'),

  registerYopmailButton.evaluate((el) => el.click()),

]);

await newPage.waitForLoadState('domcontentloaded');

console.log('New tab URL before load wait:', await newPage.url());

//await newPage.waitForLoadState('domcontentloaded', { timeout: 15000 });

console.log('domcontentloaded event passed');

  // 8. Fill the form fields via JS evaluate

  await newPage.evaluate(() => {

    const setValue = (selector, value) => {

      const el = document.querySelector(selector);

      if (el) {

        el.value = value;

        el.dispatchEvent(new Event('input', { bubbles: true }));

        el.dispatchEvent(new Event('change', { bubbles: true }));

      }

    };

    setValue('#first\_name', 'Niraj');

    setValue('#last\_name', 'User');

    setValue('#passwordL', 'Nilesh@2025');

    setValue('#password\_conf', 'Nilesh@2025');

    setValue('#mobilenumber', '2400399345');

    setValue('#zip', '111111');

  });

  // 9. Click "Submit" via JS

await newPage.evaluate(() => {

  document.querySelector('#id\_frm\_submit')?.click();

});

  // Optional: wait for result/confirmation

  //await newPage.waitForTimeout(3000);

});

//await page.locator('xpath=//\*[@id="first\_name"]').fill('Niraj');

//await page.locator('xpath=//\*[@id="last\_name"]').fill('User');

//await page.locator('xpath=//\*[@id="passwordL"]').fill('Nilesh@2025');

//await page.locator('xpath=//\*[@id="password\_conf"]').fill('Nilesh@2025');

//await page.locator('xpath=//\*[@id="mobilenumber"]').fill('2344322345');

//await page.locator('xpath=//\*[@id="zip"]').fill('111111');

// Click the account register button

//await page.locator('xpath=//\*[@id="id\_frm\_submit"]').click();

OTP-

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

test('Verify OTP via YOPMail', async ({ page, context }) => {

    test.setTimeout(60000); // 60 seconds

  // 1. Go to login page

  await page.goto('https://dev.kredsafe.net/login');

  // 2. Login with credentials

  await page.locator('input[name="email"]').fill('ts1234@yopmail.com');

  await page.locator('input[name="password"]').fill('Nilesh@2030');

  await page.locator('#id\_frm\_submit').click();

  // 3. Open new tab for YOPMail

  const yopmailTab = await context.newPage();

  await yopmailTab.goto('https://yopmail.com/en/');

await yopmailTab.waitForLoadState('networkidle');

  // 4. Enter email and open inbox

  await yopmailTab.fill('#login', 'ts1234@yopmail.com');

  await yopmailTab.click('#refreshbut');

  await yopmailTab.waitForTimeout(3000); // wait for iframe to load

  // 5. Get iframe named 'ifmail'

  const emailFrame = yopmailTab.frame({ name: 'ifmail' });

  if (!emailFrame) throw new Error('❌ Iframe "ifmail" not found');

  // 6. Get the OTP text content

  const otpElement = await emailFrame.locator('xpath=//\*[@id="mail"]/div/div/table/tbody/tr/td/p[2]');

  await otpElement.waitFor({ state: 'visible', timeout: 10000 });

  const otpText = await otpElement.textContent();

  console.log('📨 Full email content:', otpText?.trim());

  // 7. Extract OTP using regex

  const otpCodeMatch = otpText?.match(/\b\d{6}\b/);

  if (!otpCodeMatch) throw new Error('❌ OTP code not found in email');

  const otpCode = otpCodeMatch[0];

  console.log('✅ Extracted OTP:', otpCode);

await page.bringToFront();

await page.goto('https://dev.kredsafe.net/accept-otp');

await page.waitForTimeout(2000);

await page.evaluate((otp) => {

  const input = document.querySelector('#idOTP');

  input.value = otp;

  // Dispatch input event in case the site listens to it

  input.dispatchEvent(new Event('input', { bubbles: true }));

}, otpCode);

await page.evaluate(() => {

  const btn = document.querySelector('#btnSubmit');

  if (btn) btn.click();

  else console.warn('Button not found');

});  });

Pending activity-

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

test('Verify Pending Activity', async ({ page }) => {

      test.setTimeout(60000); // 60 seconds

  await page.goto('https://dev.kredsafe.net/login');

  await page.locator('//input[@name="email"]').fill('hrd786@yopmail.com');

await page.locator('//input[@name="password"]').fill('Nilesh@2025');

await page.locator('//\*[@id="id\_frm\_submit"]').click();

await page.waitForLoadState('networkidle');

const pendingTab = page.locator('#pendingActivityTab');

  await pendingTab.waitFor({ state: 'visible' });

  await pendingTab.click();

  await page.waitForTimeout(5000);

  const activityElements = page.locator('.pending-activity-class'); // Replace with actual locator

  const count = await activityElements.count();

  for (let i = 0; i < count; i++) {

    const element = activityElements.nth(i);

    const activityName = await element.innerText();

    console.log(`The Activity Name is ${activityName}`);

    if (

      activityName.includes('Overview') ||

      activityName.includes('Work Experience') ||

      activityName.includes('Education') ||

      activityName.includes('Internship') ||

      activityName.includes('Residency') ||

      activityName.includes('Fellowship') ||

      activityName.includes('Board Certification / Eligibility') ||

      activityName.includes('State Licenses or Certificates') ||

      activityName.includes('Pending Forms') ||

      activityName.includes('Pending Packets')

    ) {

      // Open link in a new tab (simulate Ctrl + Click)

      const [newPage] = await Promise.all([

        context.waitForEvent('page'),

        element.click({ modifiers: ['Control'] })

      ]);

      await newPage.waitForLoadState();

      if (

        ['Work Experience', 'Education', 'Internship', 'Residency', 'Fellowship', 'Board Certification / Eligibility'].some(section =>

          activityName.includes(section)

        )

      ) {

        const sectionTitle = newPage.locator('.section-title-class'); // replace with actual

        await sectionTitle.waitFor();

        const sectionText = await sectionTitle.innerText();

        console.log(`The section name is ${sectionText}`);

        expect(sectionText).toBe(activityName);

      } else if (activityName.includes('State Licenses or Certificates')) {

        const sectionTitle = newPage.locator('.card-title-ses'); // replace with actual

        await sectionTitle.waitFor();

        const sectionText = await sectionTitle.innerText();

        expect(sectionText).toBe(activityName);

      } else if (

        activityName.includes('Pending Forms') ||

        activityName.includes('Pending Packets')

      ) {

        const errorMsg = newPage.locator('.error-message'); // replace with actual

        await errorMsg.waitFor();

        const errorText = await errorMsg.innerText();

        const expectedText = 'Please complete your profile before accessing the forms and packets area.';

        console.log(`Error Message: ${errorText}`);

        expect(errorText).toBe(expectedText);

      }

      await newPage.close();

    } else {

      console.log('No matching pending activities found.');

    }

  }

});

Profile flow-

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

test('Verify login and complete all user profile sections', async ({ page }) => {

      test.setTimeout(60000); // 60 seconds

  // -------- LOGIN --------

  await page.goto('https://dev.kredsafe.net/login');

  await page.locator('input[name="email"]').fill('hrd29@mailinator.com');

  await page.locator('input[name="password"]').fill('Nilesh@2025');

  await page.locator('#id\_frm\_submit').click();

  await page.waitForLoadState('networkidle');

  await page.goto('https://dev.kredsafe.net/user/overview'),

  await page.waitForLoadState('networkidle');

await page.waitForTimeout(1000);

  // Fill DOB

  await page.locator('//input[@name="dob"]').fill('11/11/1990');

  // Fill SSN

  await page.locator('//\*[@id="ssn-field"]').fill('128006739'); // Replace with your SSN variable

  //await page.keyboard.press('Tab');

  await page.waitForTimeout(1000); // wait 15 seconds

  // Fill ZIP code

  await page.locator('//\*[@id="zip"]').fill('07086');

  // Select Country (if required — assuming default selected)

  // Wait for State dropdown and select

  await page.waitForSelector('//select[@id="state-list"]');

  await page.locator('//select[@id="state-list"]').selectOption({ value: 'AL' });

  // Fill City

  await page.locator('//input[@id="city"]').fill('NY');

  // Select Industry

  await page.locator('//select[@id="industry\_id"]').selectOption({ value: '255' });

  // Select Category

  await page.locator('//select[@id="category"]').selectOption({ value: 'Allied Health' });

  // Wait, Scroll, and Select Speciality

  await page.waitForTimeout(2000);

  await page.locator('//span[@class="select2-search select2-search--inline"]').scrollIntoViewIfNeeded();

  await page.locator('//span[@class="select2-search select2-search--inline"]').click();

  // Select from dropdown (highlighted)

  await page.locator('//li[contains(@class,"select2-results\_\_option--highlighted")]').click();

  // Scroll to and Click Submit Button

  const submitBtnSpan = page.locator('//\*[@id="id\_frm\_submit"]/span');

await submitBtnSpan.scrollIntoViewIfNeeded();

await submitBtnSpan.waitFor({ state: 'visible' });

await page.evaluate(el => el.click(), await submitBtnSpan.elementHandle());

  await page.waitForTimeout(2000);

  await page.waitForURL('https://dev.kredsafe.net/user/work-experiences', { timeout: 15000 });

await page.waitForLoadState('networkidle');

//   // -------- WORK EXPERIENCE --------

//  await page.goto('https://dev.kredsafe.net/user/work-experiences');

  await page.waitForLoadState('networkidle');

  await page.locator('input[name="workexperience[0][company\_name]"]').fill('Tiu');

  await page.locator('input[name="workexperience[0][job\_title]"]').fill('Sr Doctor');

  await page.locator('select#country').selectOption({ label: 'United States' });

  const workStateDropdown = page.locator('select#state-list\_0');

  await workStateDropdown.waitFor({ state: 'visible' });

  await workStateDropdown.selectOption('AL');

  await page.locator('input[name="workexperience[0][city]"]').fill('NY');

 await page.evaluate(() => {

    document.querySelector('#currently\_working\_status')?.click();

  });

  await page.locator('select#start\_month\_0').selectOption({ label: 'Jan' });

  await page.locator('select#we\_start\_year\_identity\_0').selectOption({ label: '2010' });

  const workSubmitBtn = page.locator('#id\_frm\_submit > span');

  await workSubmitBtn.scrollIntoViewIfNeeded();

  await workSubmitBtn.click();

  await page.waitForURL('https://dev.kredsafe.net/user/education', { timeout: 15000 });

  await page.waitForLoadState('networkidle');

///-------- EDUCATION --------

//  await page.goto('https://dev.kredsafe.net/user/education');

  await page.waitForLoadState('networkidle');

  await page.locator('input[name="education[0][institution\_name]"]').fill('Test');

  await page.locator('input[name="education[0][stream]"]').fill('Test');

  const eduCountryDropdown = page.locator('select#country');

  await eduCountryDropdown.waitFor({ state: 'visible' });

  await eduCountryDropdown.selectOption({ label: 'United States' });

  const eduStateDropdown = page.locator('select#state-list\_0');

  await eduStateDropdown.waitFor({ state: 'visible' });

  await expect(eduStateDropdown).toBeEnabled({ timeout: 10000 });

  await eduStateDropdown.selectOption({ label: 'Arizona' });

  await page.locator('input#city').fill('NY');

  // Click "Yes" for education confirmation using JS

  const eduYesRadio = page.locator('//\*[@id="work\_experience\_items"]/div[1]/div/div[2]/div/div[6]/div/label[1]/input[1]');

  await eduYesRadio.waitFor({ state: 'attached' });

  await page.evaluate(el => el.click(), await eduYesRadio.elementHandle());

  await page.locator('select#start\_month\_0').selectOption({ label: 'Jan' });

  await page.locator('select#we\_start\_year\_identity\_0').selectOption({ label: '2010' });

  const eduSubmitBtn = page.locator('#id\_frm\_submit > span');

  await eduSubmitBtn.scrollIntoViewIfNeeded();

  await eduSubmitBtn.waitFor({ state: 'visible' });

  await eduSubmitBtn.click();

  await page.waitForURL('https://dev.kredsafe.net/user/professional-education', { timeout: 15000 });

  await page.waitForLoadState('networkidle');

//   // -------- PROFESSIONAL EDUCATION --------

  //await page.goto('https://dev.kredsafe.net/user/professional-education');

  await page.waitForLoadState('networkidle');

  await page.locator('input[name="education[0][institution\_name]"]').fill('TIU');

  await page.locator('input[name="education[0][stream]"]').fill('sos');

  const profEduCountry = page.locator('select.autoupdate.country');

  await profEduCountry.waitFor({ state: 'visible' });

  await profEduCountry.selectOption({ label: 'United States' });

  const profEduState = page.locator('select#state-list\_0');

  await profEduState.waitFor({ state: 'visible' });

  await profEduState.selectOption({ label: 'Arizona' });

  await page.locator('input[name="education[0][city]"]').fill('Thanda');

  // Click "Yes" radio button

  const profEduYesRadio = page.locator('//\*[@id="work\_experience\_items"]/div[1]/div/div[2]/div/div[6]/div/label[1]/input[1]');

  await profEduYesRadio.waitFor({ state: 'attached' });

  await page.evaluate(el => el.click(), await profEduYesRadio.elementHandle());

  await page.locator('select#start\_month\_0').selectOption({ label: 'Jan' });

  await page.locator('select#we\_start\_year\_identity\_0').selectOption({ label: '2010' });

  const profEduSubmitBtn = page.locator('#id\_frm\_submit');

  await profEduSubmitBtn.scrollIntoViewIfNeeded();

  await profEduSubmitBtn.click();

  await page.waitForURL('https://dev.kredsafe.net/user/internship', { timeout: 15000 });

  await page.waitForLoadState('networkidle');

//  -------- INTERNSHIP --------

//  await page.goto('https://dev.kredsafe.net/user/internship');

  await page.waitForLoadState('networkidle');

  await page.locator('input[name="internship[0][job\_title]"]').fill('Lead');

  await page.locator('input[name="internship[0][employer]"]').fill('TIU');

  const internshipCountry = page.locator('select.autoupdate.country');

  await internshipCountry.waitFor({ state: 'visible' });

  await internshipCountry.selectOption({ label: 'United States' });

  const internshipState = page.locator('select#state-list\_0');

  await internshipState.waitFor({ state: 'visible' });

  await expect(internshipState).toBeEnabled({ timeout: 15000 });

  await internshipState.selectOption({ label: 'Alabama' });

  await page.locator('input[name="internship[0][city]"]').fill('Thanda');

  await page.waitForTimeout(3000); // avoid if possible

  await page.locator('select#start\_month\_0').selectOption({ label: 'Jan' });

  await page.locator('select#we\_start\_year\_identity\_0').selectOption({ label: '2010' });

  await page.locator('select#end\_month\_0').selectOption({ label: 'Jan' });

  await page.locator('select#we\_end\_year\_identity\_0').selectOption({ label: '2019' });

  const internshipSubmitBtn = page.locator('#id\_frm\_submit');

  await internshipSubmitBtn.scrollIntoViewIfNeeded();

  await internshipSubmitBtn.click();

  await page.waitForURL('https://dev.kredsafe.net/user/residencies', { timeout: 15000 });

  await page.waitForLoadState('networkidle');

//   // -------- RESIDENCIES --------

  //await page.goto('https://dev.kredsafe.net/user/residencies');

  await page.waitForLoadState('networkidle');

  await page.locator('//\*[@id="work\_experience\_items"]/div[1]/div/div[2]/div/div[1]/div/input').fill('AIIMS');

  await page.locator('input[name="residency[0][job\_title]"]').fill('QWERTY');

  await page.locator('input[name="residency[0][employer]"]').fill('ASD');

  const residencyCountry = page.locator('select.autoupdate.country');

  await residencyCountry.selectOption({ label: 'United States' });

  const residencyState = page.locator('select#state-list\_0');

  await residencyState.waitFor({ state: 'visible' });

  await residencyState.selectOption({ label: 'Arizona' });

  await page.locator('input[name="residency[0][city]"]').fill('Thanda');

  await page.locator('select#start\_month\_0').selectOption({ label: 'Jan' });

  await page.locator('select#we\_start\_year\_identity\_0').selectOption({ label: '2010' });

  await page.locator('select#end\_month\_0').selectOption({ label: 'Jan' });

  await page.locator('select#we\_end\_year\_identity\_0').selectOption({ label: '2019' });

  const residencySubmitBtn = page.locator('#id\_frm\_submit');

  await residencySubmitBtn.scrollIntoViewIfNeeded();

  await residencySubmitBtn.click();

  await page.waitForURL('https://dev.kredsafe.net/user/fellowships', { timeout: 15000 });

  await page.waitForLoadState('networkidle');

  //-------- FELLOWSHIPS --------

  //await page.goto('https://dev.kredsafe.net/user/fellowships');

  await page.waitForLoadState('networkidle');

  await page.locator('//\*[@id="work\_experience\_items"]/div[1]/div/div[2]/div/div[2]/div/input').fill('Test');

  await page.locator('input[name="fellowship[0][job\_title]"]').fill('QWERTY');

  await page.locator('input[name="fellowship[0][employer]"]').fill('ASD');

  const fellowshipCountry = page.locator('select.autoupdate.country');

  await fellowshipCountry.selectOption({ label: 'United States' });

  const fellowshipState = page.locator('select#state-list\_0');

  await fellowshipState.waitFor({ state: 'visible' });

  await fellowshipState.selectOption({ label: 'Arizona' });

  await page.locator('input[name="fellowship[0][city]"]').fill('Thanda');

  await page.locator('select#start\_month\_0').selectOption({ label: 'Jan' });

  await page.locator('select#we\_start\_year\_identity\_0').selectOption({ label: '2010' });

  await page.locator('select#end\_month\_0').selectOption({ label: 'Jan' });

  await page.locator('select#we\_end\_year\_identity\_0').selectOption({ label: '2019' });

  const fellowshipSubmitBtn = page.locator('#id\_frm\_submit');

  await fellowshipSubmitBtn.scrollIntoViewIfNeeded();

  await fellowshipSubmitBtn.click();

  await page.waitForURL('https://dev.kredsafe.net/user/certificates', { timeout: 15000 });

  await page.waitForLoadState('networkidle');

  // -------- CERTIFICATES --------

 // await page.goto('https://dev.kredsafe.net/user/certificates', { waitUntil: 'domcontentloaded', timeout: 60000 });

//  await page.goto('https://dev.kredsafe.net/user/certificates');

await page.waitForLoadState('networkidle');

await page.waitForSelector('input[name="certificate[0][certification\_name]"]', { timeout: 20000 });

  await page.locator('input[name="certificate[0][certification\_name]"]').fill('asd');

const authorityDropdown = page.locator('//select[@id="authority\_<?php echo e(0); ?>"]');

await authorityDropdown.selectOption({ value: 'AL' });

  await page.locator('input[name="certificate[0][certification\_date]"]').fill('02/02/2025');

  await page.waitForTimeout(2000);

  const certificateSubmitBtn = page.locator('#id\_frm\_submit');

  await certificateSubmitBtn.scrollIntoViewIfNeeded();

  await certificateSubmitBtn.click();

  await page.waitForURL('https://dev.kredsafe.net/user/board-certifications', { timeout: 15000 });

  await page.waitForLoadState('networkidle');

  //-------- BOARD CERTIFICATIONS --------

  //await page.goto('https://dev.kredsafe.net/user/board-certifications');

  await page.waitForLoadState('networkidle');

  await page.locator('input[name="certificate[0][certification\_name]"]').fill('asd');

  await page.locator('input[name="certificate[0][authority]"]').fill('ASDF');

  await page.locator('input[name="certificate[0][certification\_date]"]').fill('02/02/2025');

  await page.waitForTimeout(2000);

const boardCertSubmitBtn = page.locator('//\*[@id="id\_frm\_submit"]');

await boardCertSubmitBtn.scrollIntoViewIfNeeded();

await boardCertSubmitBtn.waitFor({ state: 'attached' });

await page.evaluate(el => el.click(), await boardCertSubmitBtn.elementHandle());

  await page.waitForTimeout(2000);

  // Click "click here" link

const clickHereLink = page.locator('//a[text()="click here"]');

await clickHereLink.waitFor({ state: 'visible' });

await clickHereLink.scrollIntoViewIfNeeded();

await page.evaluate((el) => el.click(), await clickHereLink.elementHandle());

});

Profile not completed –

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

test('Profile not completed', async ({ page }) => {

  await page.goto('https://dev.kredsafe.net/login');

  // Fill login form

  await page.locator('//input[@name="email"]').fill('ts1234@yopmail.com');

  await page.locator('//input[@name="password"]').fill('Nilesh@2030');

  await page.locator('//\*[@id="id\_frm\_submit"]').click();

  // Wait for page to load after login

  await page.waitForLoadState('networkidle');

  // Click the bell icon using JavaScript

  await page.evaluate(() => {

    const bellIcon = document.evaluate(

      '//\*[@id="notification-bell"]/a/i',

      document,

      null,

      XPathResult.FIRST\_ORDERED\_NODE\_TYPE,

      null

    ).singleNodeValue;

    if (bellIcon) bellIcon.click();

  });

  // Wait for UI update

  await page.waitForTimeout(2000);

  // Click on 'Form' using JavaScript

  await page.evaluate(() => {

    const form = document.evaluate(

      '/html/body/div[3]/aside/div/div[2]/nav/ul/li[6]/a/p',

      document,

      null,

      XPathResult.FIRST\_ORDERED\_NODE\_TYPE,

      null

    ).singleNodeValue;

    if (form) form.click();

  });

  // Wait for error message to appear

  await page.waitForTimeout(2000);

  // Get error message using XPath

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

    const el = document.evaluate(

      "//div[contains(text(),'Please complete your profile')]",

      document,

      null,

      XPathResult.FIRST\_ORDERED\_NODE\_TYPE,

      null

    ).singleNodeValue;

    return el ? el.textContent.trim() : null;

  });

  console.log("Error text:", errorMessage);

  // Assert error message

  const expected = "Please complete your profile before accessing the forms and packets area.";

//  if (errorMessage === expected) {

    //console.log("✅ Error message is correct:", errorMessage);

//  }

if (errorMessage?.trim() === expected.trim()) {

  console.log("✅ Error message is correct:", errorMessage);

} else {

  console.error("❌ Error message did not match!");

  console.error("Expected:", expected);

  console.error("Actual:  ", errorMessage);

}

});

Recent activity-

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

test('Verify Recent Activity', async ({ page }) => {

      test.setTimeout(60000); // 60 seconds

  await page.goto('https://dev.kredsafe.net/login');

  // Login

  await page.locator('input[name="email"]').fill('ts1234@yopmail.com');

  await page.locator('input[name="password"]').fill('Nilesh@2030');

  await page.locator('#id\_frm\_submit').click();

  // Wait for navigation to complete

  await page.waitForURL('\*\*/user/home');

  await page.waitForTimeout(5000); // Let the activities table render

  // Define locators

  const activityRows = page.locator('#activities > tbody > tr');

  const scrollContainer = page.locator('.dataTables\_scrollBody'); // Confirm this selector in dev tools

  let previousCount = 0;

  let currentCount = await activityRows.count();

  // Keep scrolling until all rows are loaded

  while (currentCount > previousCount) {

    previousCount = currentCount;

    // Scroll to bottom

    await scrollContainer.evaluate(el => el.scrollTop = el.scrollHeight);

    // Wait for new data to load

    await page.waitForTimeout(1000);

    // Recalculate count

    currentCount = await activityRows.count();

  }

  console.log('✅ Total Recent Activities Loaded:', currentCount);

  // Optional: assert at least 1,000+ entries (if required)

  expect(currentCount).toBeGreaterThan(1000);

});

Restore profile-

import { test } from '@playwright/test';

test('Restore flow with visible JS clicks and robust handling', async ({ page }) => {

  // 1. Zoom in for clarity (optional)

      test.setTimeout(60000); // 60 seconds

   await page.goto('https://dev.kredsafe.net/login');

  await page.locator('input[name="email"]').fill('ts050176@yopmail.com');

  await page.locator('input[name="password"]').fill('Nilesh@2025');

  await page.locator('#id\_frm\_submit').click();

  await page.waitForLoadState('networkidle');

  /// 1. Navigate to the Internship page

await page.goto('https://dev.kredsafe.net/user/internship');

await page.waitForLoadState('networkidle');

// 2. Click the "View Archive of Internship" button

const archiveBtn = page.locator("//button[@title='View Archive of Internship']");

await archiveBtn.waitFor({ state: 'attached', timeout: 10000 }); // Safe wait

await page.evaluate(el => el.click(), await archiveBtn.elementHandle());

await page.waitForLoadState('networkidle');

// 3. Select the "Check All" checkbox

const checkAllBox = page.locator('//input[@id="checkAll"]');

await checkAllBox.waitFor({ state: 'visible', timeout: 10000 });

await checkAllBox.click();

// 4. Click the "Restore" button

const restoreBtn = page.locator("//a[@title='Restore' and contains(@class, 'btn\_restore\_arch\_itms')]");

await restoreBtn.waitFor({ state: 'visible', timeout: 10000 });

await restoreBtn.click();

  // // 4. Helper: JS click by XPath with pause

  // async function clickByXPath(xpath, pauseMs = 1500) {

  //   await page.evaluate((xp) => {

  //     const el = document.evaluate(xp, document, null, XPathResult.FIRST\_ORDERED\_NODE\_TYPE, null).singleNodeValue;

  //     el?.scrollIntoView({ behavior: 'smooth', block: 'center' });

  //     el?.click();

  //   }, xpath);

  //   console.log(`Clicked element via XPath: ${xpath}`);

  //   await page.waitForTimeout(pauseMs);

  // }

  // // 5. Perform clicks in the correct sequence

  // await clickByXPath("//a[@class='remove\_me' and @rel='1']");

  // await clickByXPath("/html/body/div[12]/div/div/div[3]/button[2]/span");

  // await clickByXPath('/html/body/div[3]/div[1]/section[2]/div/div/div[1]/div/div[2]/button/span/i');

  // await clickByXPath('//input[@id="checkAll"]');

  // await clickByXPath('/html/body/div[3]/div[4]/div/div/div[2]/a/span');

  // await clickByXPath('/html/body/div[3]/div[1]/section[2]/div/div/div[1]/div/div[2]/button/span/i');

  // await clickByXPath("//\*[@id='recycleInfo']//label[contains(text(),'Close')]");

  // // 6. Final pause to observe the result

  // await page.waitForTimeout(2000);

});

Subscription-

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

test('complete subscription and payment flow', async ({ page }) => {

test.setTimeout(60000);

  await page.goto('https://dev.kredsafe.net/login');

  // Login

  await page.getByRole('textbox', { name: 'E-mail Address \*' }).fill('ts1409@yopmail.com');

  await page.getByRole('textbox', { name: 'Password \*' }).fill('Nilesh@2025');

  await page.getByRole('button', { name: 'Login' }).click();

  await page.waitForLoadState('load');

  // Click subscribe button

  const subscribeBtn = page.locator('//button[@id="subscribe2"]');

      await page.waitForTimeout(10000);

  await subscribeBtn.waitFor({ state: 'visible' });

  await subscribeBtn.scrollIntoViewIfNeeded();

  await subscribeBtn.click();

  await page.waitForLoadState('load');

  // Click home tab button

  // const homeTabButton = page.locator('//button[@id="home-tab\_1"]');

  // await homeTabButton.waitFor({ state: 'visible' });

  // await homeTabButton.scrollIntoViewIfNeeded();

  // await homeTabButton.click();

  // console.log("Credi Button clicked");

const confirmBtn = page.locator("//button[@class='button confirm-btn']");

  await confirmBtn.waitFor({ state: 'visible' });

await confirmBtn.scrollIntoViewIfNeeded();

await confirmBtn.click();

  console.log("Button clicked");

 // await page.waitForLoadState('load');

  await page.waitForLoadState('domcontentloaded');

  // Fill card details

   await page.waitForTimeout(5000);

  await page.locator('//\*[@id="cardNumber"]').waitFor({ state: 'visible' });

     await page.locator('//\*[@id="cardNumber"]').fill('4111111111111111');

  await page.locator('//\*[@id="cardExpiry"]').fill('09/25');

  await page.locator('//\*[@id="cardCvc"]').fill('111');

  // Fill billing info

  await page.locator('//input[@id="billingName"]').fill('hcheeti');

  await page.locator('//input[@id="billingAddressLine1"]').fill('Nagpur');

  await page.locator('//input[@id="billingLocality"]').fill('Nagpur');

  await page.locator('//input[@id="billingPostalCode"]').fill('441108');

  // Select state (3rd option)

  const stateDropdown = page.locator('//select[@id="billingAdministrativeArea"]');

  await stateDropdown.selectOption({ index: 2 }); // 0-based index

  // Scroll to Pay button and click via JS click

  const payButton = page.locator('//div[@class="SubmitButton-IconContainer"]');

  await payButton.waitFor({ state: 'visible' });

  await payButton.scrollIntoViewIfNeeded();

await payButton.click();

await page.waitForURL('https://dev.kredsafe.net/user/subscription/dashboard', { timeout: 30000 });

  await page.waitForLoadState('load');

});

Subscription Payment Premium-

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

test('complete subscription and payment flow', async ({ page }) => {

test.setTimeout(60000);

  await page.goto('https://dev.kredsafe.net/login');

  // Login

  await page.getByRole('textbox', { name: 'E-mail Address \*' }).fill('hrd14566@yopmail.com');

  await page.getByRole('textbox', { name: 'Password \*' }).fill('Nilesh@2025');

  await page.getByRole('button', { name: 'Login' }).click();

  await page.waitForLoadState('load');

  // Click subscribe button

// const poneyear = page.locator("//a[@href='#tab\_premium\_res\_50']");

// await poneyear.waitFor({ state: 'visible' });

// await page.evaluate(el => el.click(), await poneyear.elementHandle());

// Similarly for 'psixmonth'

// const psixmonth = page.locator("//input[@type='radio' and @name='prem\_installments' and @data-month='6' and @value='2']");

// await psixmonth.waitFor({ state: 'visible' });

// await page.evaluate(el => el.click(), await psixmonth.elementHandle());

// Similarly for '12xmonth'

//   const tmonth = page.locator("//input[contains(@class, 'prem\_radio\_12') and contains(@class, 'prem\_radio\_50') and @value='1']");

// await tmonth.waitFor({ state: 'visible' });

// await page.evaluate(el => el.click(), await tmonth.elementHandle());

const pthreeyear  = page.locator("//a[@href='#tab\_premium\_res\_48']");

 await pthreeyear .waitFor({ state: 'visible' });

 await page.evaluate(el => el.click(), await pthreeyear .elementHandle());

//3 Yera -full payment

// const pthreefullpay  = page.locator("//input[@name='prem\_installments' and @data-month='3' and @value='1']");

// await pthreefullpay .waitFor({ state: 'visible' });

//  await page.evaluate(el => el.click(), await pthreefullpay .elementHandle());

// const pthreetwelvwmonth  = page.locator("//input[@type='radio' and @name='prem\_installments' and @data-month='1' and @value='3']");

// await pthreetwelvwmonth .waitFor({ state: 'visible' });

// await page.evaluate(el => el.click(), await pthreetwelvwmonth .elementHandle());

 const pthreesixmonthr  = page.locator("//input[@type='radio' and @name='prem\_installments' and @data-month='6' and @value='6']");

 await pthreesixmonthr .waitFor({ state: 'visible' });

await page.evaluate(el => el.click(), await pthreesixmonthr .elementHandle());

const subscribeBtn = page.locator("//button[@id='subscribe5']");

      await page.waitForTimeout(10000);

  await subscribeBtn.waitFor({ state: 'visible' });

  await subscribeBtn.scrollIntoViewIfNeeded();

  await subscribeBtn.click();

  await page.waitForLoadState('load');

  // Click home tab button

  // const homeTabButton = page.locator('//button[@id="home-tab\_1"]');

  // await homeTabButton.waitFor({ state: 'visible' });

  // await homeTabButton.scrollIntoViewIfNeeded();

  // await homeTabButton.click();

  // console.log("Credi Button clicked");

const confirmBtn = page.locator("//button[@class='button confirm-btn']");

  await confirmBtn.waitFor({ state: 'visible' });

await confirmBtn.scrollIntoViewIfNeeded();

await confirmBtn.click();

  console.log("Button clicked");

 // await page.waitForLoadState('load');

  await page.waitForLoadState('domcontentloaded');

  // Fill card details

   await page.waitForTimeout(5000);

  await page.locator('//\*[@id="cardNumber"]').waitFor({ state: 'visible' });

     await page.locator('//\*[@id="cardNumber"]').fill('4111111111111111');

  await page.locator('//\*[@id="cardExpiry"]').fill('09/25');

  await page.locator('//\*[@id="cardCvc"]').fill('111');

  // Fill billing info

  await page.locator('//input[@id="billingName"]').fill('hcheeti');

  await page.locator('//input[@id="billingAddressLine1"]').fill('Nagpur');

  await page.locator('//input[@id="billingLocality"]').fill('Nagpur');

  await page.locator('//input[@id="billingPostalCode"]').fill('441108');

  // Select state (3rd option)

  const stateDropdown = page.locator('//select[@id="billingAdministrativeArea"]');

  await stateDropdown.selectOption({ index: 2 }); // 0-based index

  // Scroll to Pay button and click via JS click

  const payButton = page.locator('//div[@class="SubmitButton-IconContainer"]');

  await payButton.waitFor({ state: 'visible' });

  await payButton.scrollIntoViewIfNeeded();

await payButton.click();

   await page.waitForTimeout(5000);

await page.waitForURL('https://dev.kredsafe.net/user/subscription/dashboard', { timeout: 30000 });

  await page.waitForLoadState('load');

});

Subscription Standard –

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

test('complete subscription and payment flow', async ({ page }) => {

test.setTimeout(60000);

  await page.goto('https://dev.kredsafe.net/login');

  // Login

  await page.getByRole('textbox', { name: 'E-mail Address \*' }).fill('hrd14566@yopmail.com');

  await page.getByRole('textbox', { name: 'Password \*' }).fill('Nilesh@2025');

  await page.getByRole('button', { name: 'Login' }).click();

  await page.waitForLoadState('load');

      const oneyear = page.locator("//a[@href='#tab\_std\_res\_30']");

      await oneyear.waitFor({ state: 'visible' });

      await page.evaluate((el) => el.click(), await oneyear.elementHandle());

// const sixmonth = page.locator("//input[@name='installments' and @value='2']");

// await sixmonth.waitFor({ state: 'visible' });

// await page.evaluate((el) => el.click(), await sixmonth.elementHandle());

const twelvemonth = page.locator("//input[@type='radio' and @name='installments' and @value='1']");

await twelvemonth.waitFor({ state: 'visible' });

await page.evaluate((el) => el.click(), await twelvemonth.elementHandle());

  // Click subscribe button

  const subscribeBtn = page.locator('//button[@id="subscribe2"]');

      await page.waitForTimeout(10000);

  await subscribeBtn.waitFor({ state: 'visible' });

  await subscribeBtn.scrollIntoViewIfNeeded();

  await subscribeBtn.click();

  await page.waitForLoadState('load');

  // Click home tab button

  // const homeTabButton = page.locator('//button[@id="home-tab\_1"]');

  // await homeTabButton.waitFor({ state: 'visible' });

  // await homeTabButton.scrollIntoViewIfNeeded();

  // await homeTabButton.click();

  // console.log("Credi Button clicked");

const confirmBtn = page.locator("//button[@class='button confirm-btn']");

  await confirmBtn.waitFor({ state: 'visible' });

await confirmBtn.scrollIntoViewIfNeeded();

await confirmBtn.click();

  console.log("Button clicked");

  await page.waitForLoadState('domcontentloaded');

  // Fill card details

   await page.waitForTimeout(5000);

  await page.locator('//\*[@id="cardNumber"]').waitFor({ state: 'visible' });

     await page.locator('//\*[@id="cardNumber"]').fill('4111111111111111');

  await page.locator('//\*[@id="cardExpiry"]').fill('09/25');

  await page.locator('//\*[@id="cardCvc"]').fill('111');

  // Fill billing info

  await page.locator('//input[@id="billingName"]').fill('hch');

  await page.locator('//input[@id="billingAddressLine1"]').fill('Nagpur');

  await page.locator('//input[@id="billingLocality"]').fill('Nagpur');

  await page.locator('//input[@id="billingPostalCode"]').fill('441108');

  // Select state (3rd option)

  const stateDropdown = page.locator('//select[@id="billingAdministrativeArea"]');

  await stateDropdown.selectOption({ index: 2 }); // 0-based index

  // Scroll to Pay button and click via JS click

  const payButton = page.locator('//div[@class="SubmitButton-IconContainer"]');

  await payButton.waitFor({ state: 'visible' });

  await payButton.scrollIntoViewIfNeeded();

await payButton.click();

   await page.waitForTimeout(5000);

   await page.waitForTimeout(1000);

await page.waitForURL('https://dev.kredsafe.net/user/subscription/dashboard', { timeout: 40000 });

  await page.waitForLoadState('load');

});

Upload Profile Picture-

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

test('Verify Upload profile picture', async ({ page }) => {

  test.setTimeout(60000);

  await page.goto('https://dev.kredsafe.net/login');

  await page.locator('//input[@name="email"]').fill('ts1234@yopmail.com');

await page.locator('//input[@name="password"]').fill('Nilesh@2030');

await page.locator('//\*[@id="id\_frm\_submit"]').click();

  await page.waitForTimeout(5000);

await page.goto('https://dev.kredsafe.net/user/overview');

  await page.waitForTimeout(3000);

const upload = page.locator('#profile\_img');

// Move to the element and click it using JS click

await upload.evaluate(el => el.click());

await upload.setInputFiles('C:\\Users\\Admin\\Desktop\\Tesst Data\\surecafetest.jpg');

  console.log('✅ Profile image uploaded successfully.');

});

Upload Resume-

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

import path from 'path';

test('Upload resume and submit form', async ({ page }) => {

 test.setTimeout(60000);

  await page.goto('https://dev.kredsafe.net');

  await page.waitForTimeout(5000);

  // Scroll and click 'Register Here' using JS

  const registerHereLink = page.locator('xpath=//a[@href="https://dev.kredsafe.net/users/register/options"]');

  await registerHereLink.scrollIntoViewIfNeeded();

  await page.evaluate(el => el.click(), await registerHereLink.elementHandle());

  await page.waitForTimeout(5000);

  // Click on Upload Resume button

  const accoutRegisterUploadButton = page.locator('xpath=/html/body/div[1]/div[1]/div/div/div[2]/div/div/div[2]/a');

  await accoutRegisterUploadButton.click();

  await page.waitForTimeout(3000);

  // Refresh the page and re-select upload input

//await page.goto(page.url(), { timeout: 4000, waitUntil: 'load' });

//await page.waitForTimeout(2000);

await page.waitForLoadState('load');

  // Upload the file

  const fileInput = page.locator('xpath=//\*[@id="resume\_file\_upload"]');

    await fileInput.waitFor({ state: 'visible', timeout: 10000 });

  const filePath = path.resolve('C:/Users/Admin/Downloads/DerekSmith(1).doc');

  await fileInput.setInputFiles(filePath);

  // Click on resume submit button

  const uploadResumeSubmitButton = page.locator('xpath=//\*[@id="id\_frm\_submit\_resume"]');

  await uploadResumeSubmitButton.click();

console.log('Clicked submit, waiting now');

//await page.waitForTimeout(1000);

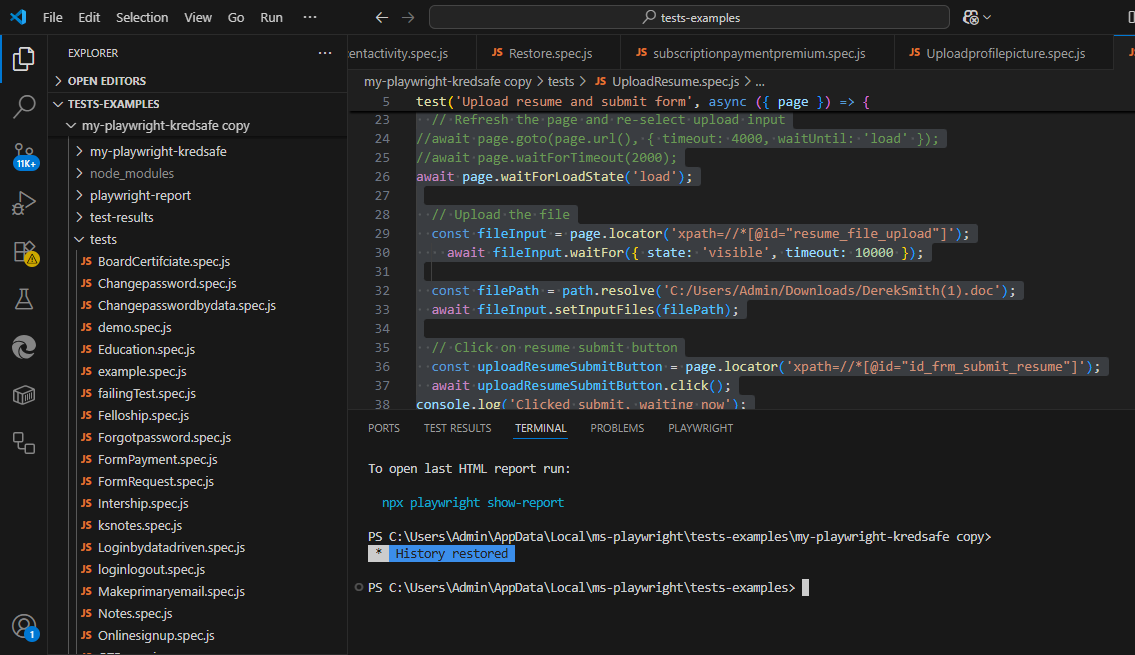
console.log('Wait finished');

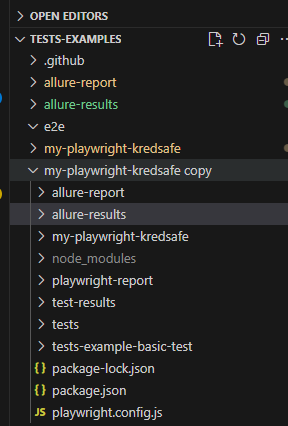
  // Scroll and submit the full form

await page.evaluate(() => document.querySelector('#id\_frm\_submit\_resume\_data')?.click());

  console.log('Form submitted successfully');

});





Playwright .cofig.js

import { defineConfig, devices } from '@playwright/test';

const LT\_USERNAME = 'nshekdar';

const LT\_ACCESS\_KEY = 'LT\_6VFot7JxPlDmRO9p8mFz8W2HfdUTKedygGUXxh0bKM4gH0m';

export default defineConfig({

  testDir: './tests',

  fullyParallel: true,

  forbidOnly: !!process.env.CI,

  retries: process.env.CI ? 2 : 0,

  workers: process.env.CI ? 1 : undefined,

  reporter: [

    ['list'],

    ['html'],

    ['allure-playwright', { outputFolder: 'allure-results' }],

  ],

  use: {

    trace: 'on-first-retry',

  },

  projects: [

    {

      name: 'chrome',

      use: { ...devices['Desktop Chrome'], channel: 'chrome' },

    },

    {

      name: 'webkit',

      use: { ...devices['Desktop Safari'] },

    },

    {

      name: 'firefox',

      use: { ...devices['Desktop Firefox'] },

    },

    {

      name: 'chrome-lambdatest',

      use: {

        browserName: 'chromium',

        connectOptions: {

          wsEndpoint: `wss://cdp.lambdatest.com/playwright?capabilities=${encodeURIComponent(

            JSON.stringify({

              browserName: 'Chrome',

              browserVersion: 'latest',

              'LT:Options': {

                platform: 'Windows 11',

                build: 'Playwright LambdaTest Build',

                name: 'Chrome Test on LambdaTest',

                user: LT\_USERNAME,

                accessKey: LT\_ACCESS\_KEY,

                console: true,

                network: true,

              },

            })

          )}`,

        },

      },

    },

  ],

});

Package.json

{

  "name": "my-playwright-kredsafe",

  "version": "1.0.0",

  "description": "",

  "main": "index.js",

  "scripts": {

    "test": "npx playwright test",

    "allure:generate": "allure generate allure-results --clean -o allure-report",

    "allure:open": "allure open allure-report"

  },

  "keywords": [],

  "author": "",

  "license": "ISC",

  "type": "commonjs",

  "devDependencies": {

    "@playwright/test": "^1.54.2",

    "allure-commandline": "^2.34.1",

    "allure-playwright": "^3.3.3",

    "ts-node": "^10.9.2",

    "typescript": "^5.9.2"

  }

}

**Playwright Codegen**

* Open a browser
* Interact with the UI manually
* Automatically generate a test script (JavaScript, TypeScript, Python, etc.)

## Steps to Record a Script with Playwright

Prerequisite: Make sure Playwright is installed  
If not installed:

npm install -D @playwright/test

npx playwright install

### Step 1: Run Codegen Command

npx playwright codegen https://example.com

This will:

* Launch a browser window
* Open the URL (https://example.com)
* Open a side panel with live-generated code

### Step 2: Interact with the Website

As you:

* Click buttons
* Fill out forms
* Navigate pages

Playwright will automatically record your actions and generate the corresponding code.

### Step 3: Copy the Script

Once you're done:

* Copy the script from the Codegen window
* Paste it into a test file (e.g., tests/example.spec.ts)

### Step 4: Run Your Test

Save the test file and run it using:

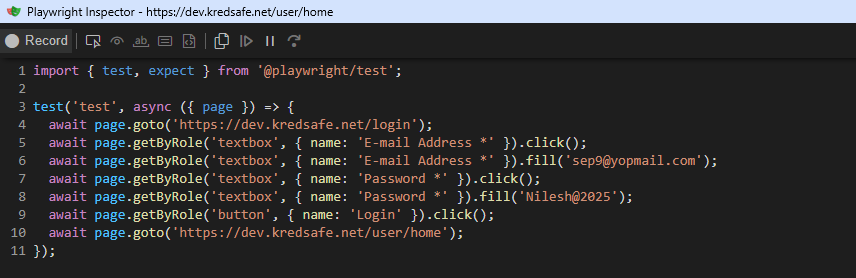
npx playwright test tests/example.spec.ts

## Optional: Specify Language or Browser

### Record using a specific browser (e.g., Firefox):

npx playwright codegen --browser=firefox <https://example.com>

npx playwright codegen <https://dev.kredsafe.net/login>  
  
Recorded login scripts in recorder .



### Record in a specific language:

| **Language** | **Command Example** |
| --- | --- |
| JavaScript | Default |
| TypeScript | --target=typescript |
| Python | --target=python |
| Java | --target=java |
| C# (.NET) | --target=csharp |

Example:

npx playwright codegen --target=python https://example.com

## Sample Output (TypeScript)

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

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

await page.goto('https://dev.kredsafe.net/login');

await page.getByRole('textbox', { name: 'E-mail Address \*' }).click();

await page.getByRole('textbox', { name: 'E-mail Address \*' }).fill('sep9@yopmail.com');

await page.getByRole('textbox', { name: 'Password \*' }).click();

await page.getByRole('textbox', { name: 'Password \*' }).fill('Nilesh@2025');

await page.getByRole('button', { name: 'Login' }).click();

await page.goto('https://dev.kredsafe.net/user/home');

});

## Tips

* Use **selectors** wisely: Playwright tries to use accessible and reliable ones (like text= or role=).
* Recorded scripts may need **cleanup** (removing unnecessary waits or clicks).
* You can chain assertions using expect().

## ****Allure Report****

**Allure Report** is a flexible and powerful **test report framework** that provides:

* **Beautiful and interactive HTML reports**
* Clear display of passed/failed/skipped tests
* Support for **screenshots**, **logs**, **videos**, and **attachments**
* Integrates with **Playwright**, **Selenium**, **Cypress**, **JUnit**, **TestNG**, and more
* Useful for **manual** and **automated test results**

### Key Features:

| **Feature** | **Description** |
| --- | --- |
| Organized Results | Shows test suite > test case > step level |
| ⏱️Timeline View | Shows test durations |
| Step-by-step logs | You can view actions inside a test |
| Screenshots | Auto-included on failure (with setup) |
| Trend/History | See test trends across builds |
| CI Integration | Can be added to Jenkins, GitHub Actions, etc. |

## ****Set Up Allure Report**** for Playwright (Step-by-Step)

Let’s assume you’re using **Playwright Test Runner** (not Jest or Mocha).

### Step 1: Install Allure Packages

npm i -D allure-playwright

allure-playwright is a plugin that hooks into Playwright.

### Step 2: Configure playwright.config.ts

Add this to your config to enable the reporter:

import { defineConfig } from '@playwright/test';

export default defineConfig({

reporter: [

['list'],

['allure-playwright']

],

});

### Step 3: Add Allure Command Line Tool

You can install globally:

npm i -g allure-commandline

Or add as a dev dependency:

npm i -D allure-commandline

### Step 4: Run Your Tests

npx playwright test

After running, the raw report will be in:

./allure-results/

### Step 5: Generate Allure HTML Report

allure generate allure-results --clean -o allure-report

This creates an HTML report in allure-report/.

### Step 6: Open the Report in Browser

allure open allure-report

## 🧪 Optional: GitHub Actions Integration

You can add a step in your CI:

- name: Generate Allure Report

run: |

npm install -g allure-commandline

allure generate allure-results --clean -o allure-report

## Directory Summary

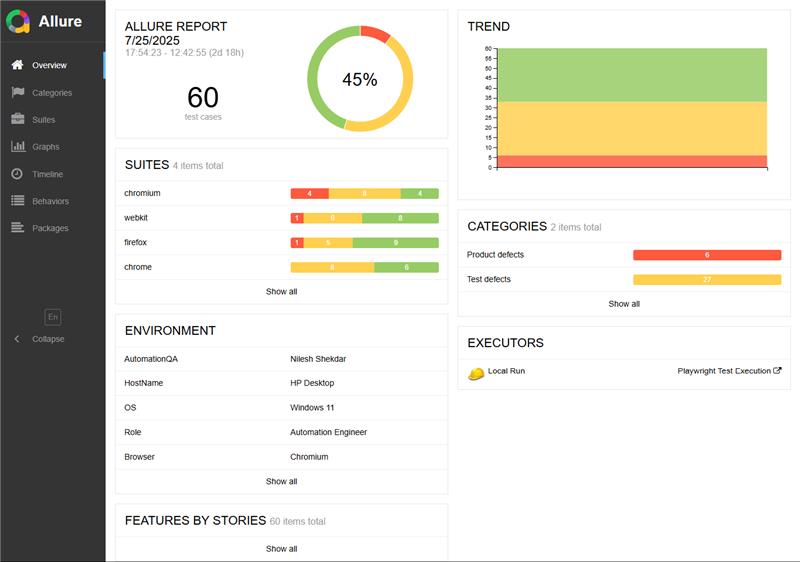
project/

├── tests/

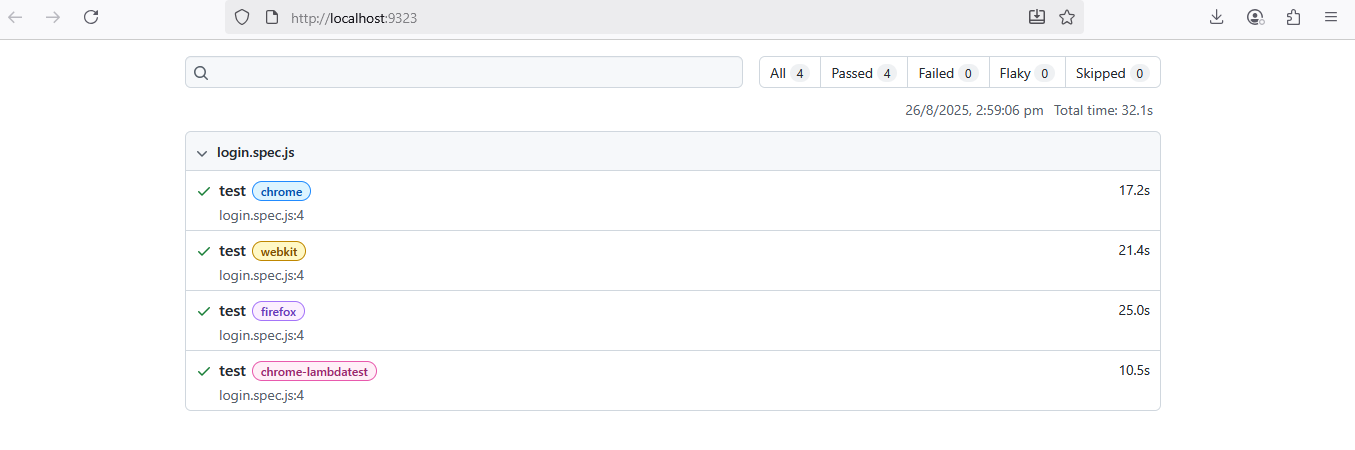
├── allure-results/ # Raw results generated after test run

├── allure-report/ # HTML report generated here

├── playwright.config.ts



npx playwright show-report



Playwright CI/CD-

* **CI (Continuous Integration):** Automatically builds and tests code every time it's committed to the repository.
* **CD (Continuous Deployment):** Automatically deploys the application to production or staging environments after passing CI steps.

### Use of Playwright in CI/CD

* Ensures every code change is tested before deployment.
* Detects issues early, reducing bugs in production.
* Supports **headless** and **headed** browser testing in CI environments.

## Requirements for CI/CD Setup with Playwright

### Software / Tools Needed:

| **Tool** | **Purpose** |
| --- | --- |
| **Git** | Version control |
| **Node.js & npm** | Run Playwright scripts |
| **Playwright** | E2E testing framework |
| **CI/CD Platform** (Choose one) | Run automated pipelines (e.g., GitHub Actions, GitLab CI, Jenkins, CircleCI) |
| **Optional: Docker** | For containerized test runs |
| **Browsers** | Installed by Playwright or OS-level (Chrome, Firefox, etc.) |

## Step-by-Step CI/CD Setup with Playwright

### 1. Set up Playwright Project

# 1. Create a new project

mkdir playwright-tests && cd playwright-tests

npm init -y

# 2. Install Playwright

npm i -D @playwright/test

# 3. Install browser binaries

npx playwright install

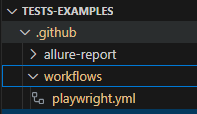
# 4. Create a sample test

npx playwright codegen https://example.com

### 2. Choose and Set Up a CI/CD Platform

Let’s use **GitHub Actions** as an example (others like GitLab CI or Jenkins are similar in structure).

https://github.com/nilesh1710/KredSafe



**playwright.yml- Most Updated**

name: Playwright Tests

on:

push:

branches: [ "master" ]

pull\_request:

branches: [ "master" ]

jobs:

test-and-push:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v3

- name: Setup Node.js

uses: actions/setup-node@v3

with:

node-version: 18

- name: Install dependencies

run: npm ci

- name: Install Playwright Browsers

run: npx playwright install --with-deps

- name: Run specific Playwright test (Restorevalidation)

run: npx playwright test tests/Restorevalidation.spec.js

- name: Commit and push changes if test passes

if: success()

run: |

git config user.name "github-actions[bot]"

git config user.email "github-actions[bot]@users.noreply.github.com"

# Check for file changes (optional: adjust paths)

git add .

if git diff --cached --quiet; then

echo "✅ No changes to commit"

else

git commit -m "Auto-push: Tests passed for Restorevalidation"

git push origin master

fi

- name: Upload Playwright report (always runs)

if: always()

uses: actions/upload-artifact@v4

with:

name: playwright-report

path: playwright-report

3. Create GitHub Actions Workflow-Updated

PS C:\Users\Admin\AppData\Local\ms-playwright\tests-examples>  
C:\Users\Admin\AppData\Local\ms-playwright\tests-examples\my-playwright-kredsafe copy\tests\Notes.spec.js

C:\Users\Admin\AppData\Local\ms-playwright\tests-examples\my-playwright-kredsafe copy\playwright.config.js

name: Playwright Tests

on: workflow\_dispatch:

push:

branches:

- master

jobs:

run-tests:

runs-on: ubuntu-latest

steps:

- name: Checkout repository without submodules

uses: actions/checkout@v3

with:

submodules: false

fetch-depth: 0 # fetch full history, optional but useful

- name: Remove leftover submodule config if any

run: |

git submodule deinit -f --all || echo "No submodules to deinit"

git rm --cached -r KredSafe || echo "No submodule KredSafe to remove"

git config --remove-section submodule.KredSafe || echo "No submodule section KredSafe to remove"

- name: Cache Node.js modules

uses: actions/cache@v3

with:

path: ~/.npm

key: ${{ runner.os }}-node-${{ hashFiles('package-lock.json') }}

restore-keys: |

${{ runner.os }}-node-

- name: Set up Node.js

uses: actions/setup-node@v3

with:

node-version: 18

- name: Install dependencies

run: npm ci

- name: Install Playwright Browsers

run: npx playwright install --with-deps

- name: Run Notes.spec.js on Chrome only

run: npx playwright test tests/Notes.spec.js --project=chrome --reporter=html --config=playwright.config.js - name: Upload Playwright HTML Report

if: always()

uses: actions/upload-artifact@v4

with:

name: playwright-report

path: playwright-report/

Update CICD Flow –  
Terminal Url-

C:\Users\Admin\AppData\Local\ms-playwright\tests-examples\my-playwright-kredsafe copy>

C:\Users\Admin\AppData\Local\ms-playwright\tests-examples\my-playwright-kredsafe copy\tests

C:\Users\Admin\AppData\Local\ms-playwright\tests-examples\my-playwright-kredsafe copy\tests\Notes.spec.js

C:\Users\Admin\AppData\Local\ms-playwright\tests-examples\my-playwright-kredsafe copy\playwright.config.js

name: Playwright Tests

on:

workflow\_dispatch:

push:

branches:

- master

jobs:

run-tests:

runs-on: ubuntu-latest

steps:

- name: Checkout repository without submodules

uses: actions/checkout@v3

with:

submodules: false

fetch-depth: 0 # fetch full history, optional but useful

- name: Remove leftover submodule config if any

run: |

git submodule deinit -f --all || echo "No submodules to deinit"

git rm --cached -r KredSafe || echo "No submodule KredSafe to remove"

git config --remove-section submodule.KredSafe || echo "No submodule section KredSafe to remove"

- name: Cache Node.js modules

uses: actions/cache@v3

with:

path: ~/.npm

key: ${{ runner.os }}-node-${{ hashFiles('package-lock.json') }}

restore-keys: |

${{ runner.os }}-node-

- name: Set up Node.js

uses: actions/setup-node@v3

with:

node-version: 18

- name: Install dependencies

run: npm ci

- name: Install Playwright Browsers

run: npx playwright install --with-deps

- name: Run Notes.spec.js on Chrome only

run: npx playwright test tests/Notes.spec.js --project=chrome --reporter=html --config=playwright.config.js

- name: Upload Playwright HTML Report

if: always()

uses: actions/upload-artifact@v4

with:

name: playwright-report

path: playwright-report/

name: Playwright Tests

on:

workflow\_dispatch:

push:

branches:

- master

jobs:

run-tests:

runs-on: ubuntu-latest

steps:

- name: Checkout repository

uses: actions/checkout@v3

- name: Cache Node.js modules

uses: actions/cache@v3

with:

path: ~/.npm

key: ${{ runner.os }}-node-${{ hashFiles('package-lock.json') }}

restore-keys: |

${{ runner.os }}-node-

- name: Set up Node.js

uses: actions/setup-node@v3

with:

node-version: 18

- name: Install dependencies

run: npm ci

- name: Install Playwright Browsers

run: npx playwright install --with-deps

- name: Run Notes.spec.js on Chrome only

run: npx playwright test tests/Notes.spec.js --project=chrome --reporter=html --config=playwright.config.js

- name: Upload Playwright HTML Report

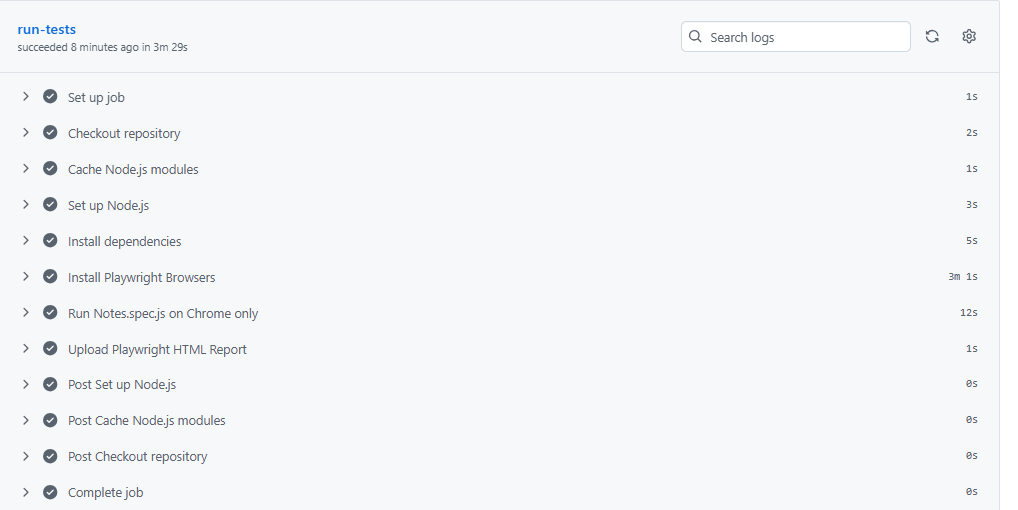
if: always()

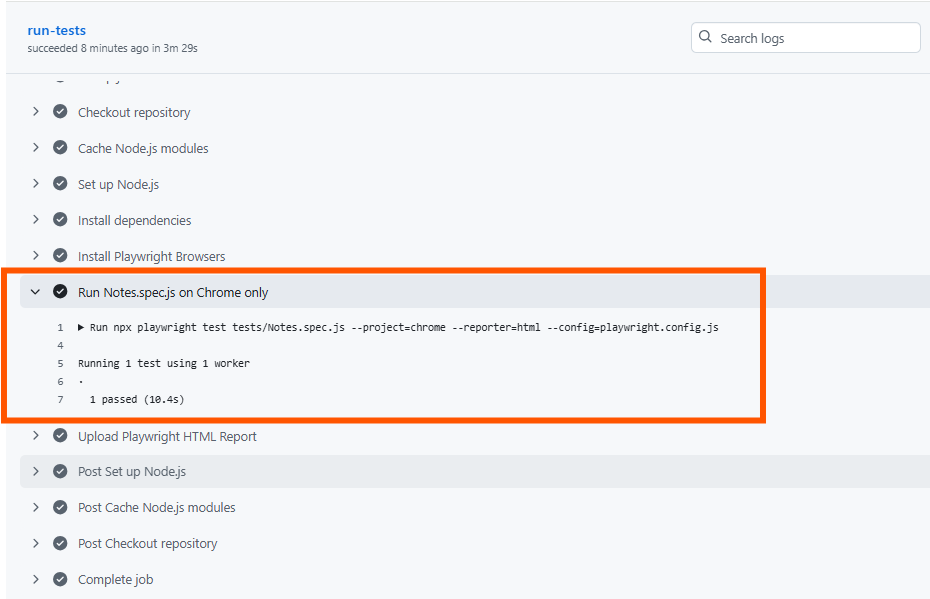
uses: actions/upload-artifact@v4

with:

name: playwright-report

path: playwright-report/

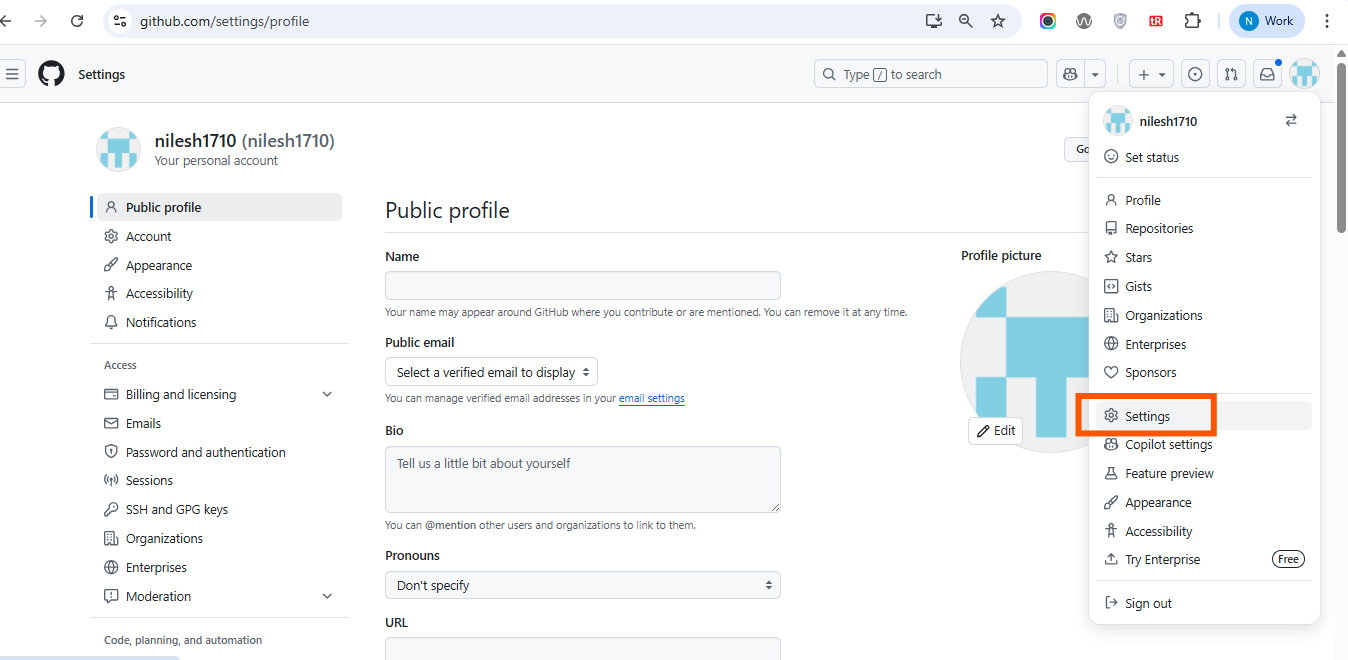


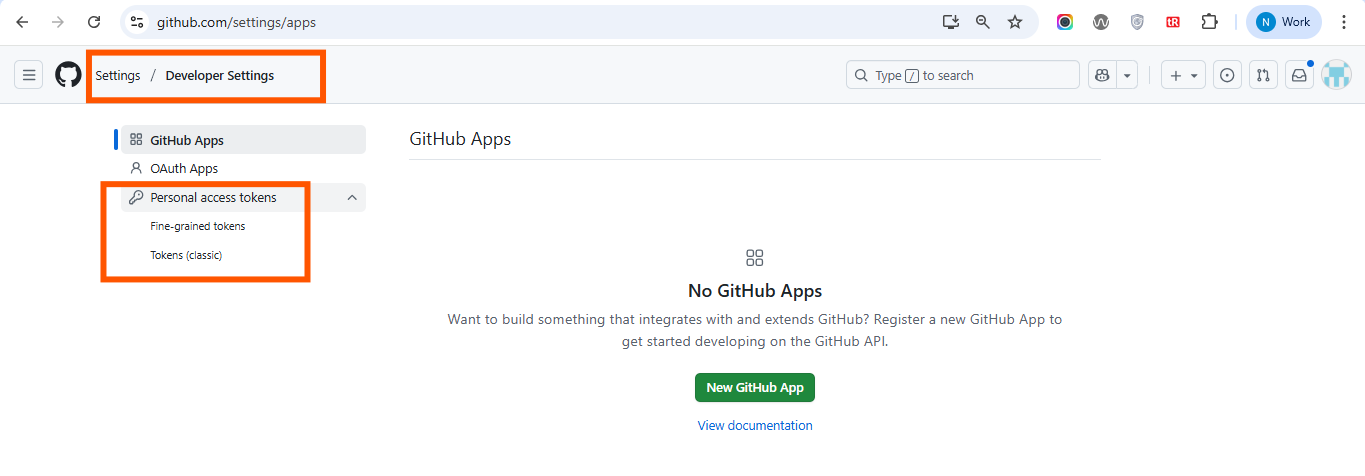


## How to Create a GitHub Personal Access Token (Classic)

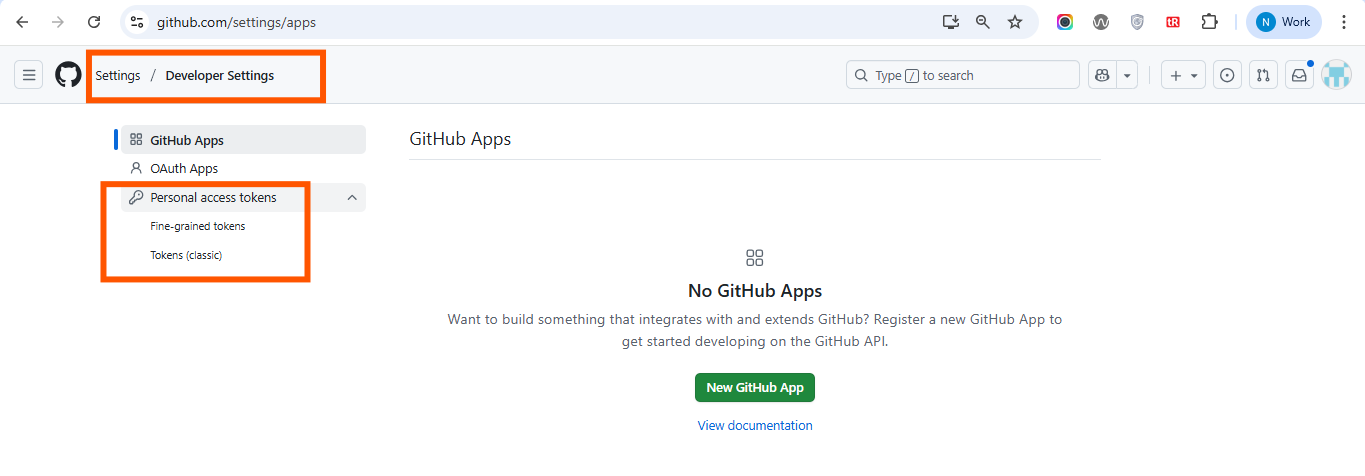
### Step 1: Go to Developer Settings

1. Open your browser and go to [**https://github.com/**](https://github.com/)
2. Click your **profile picture** (top-right corner)
3. Click **"Settings"** from the dropdown menu
4. Scroll down the left-hand sidebar
5. Click on **"Developer settings"** (last option)



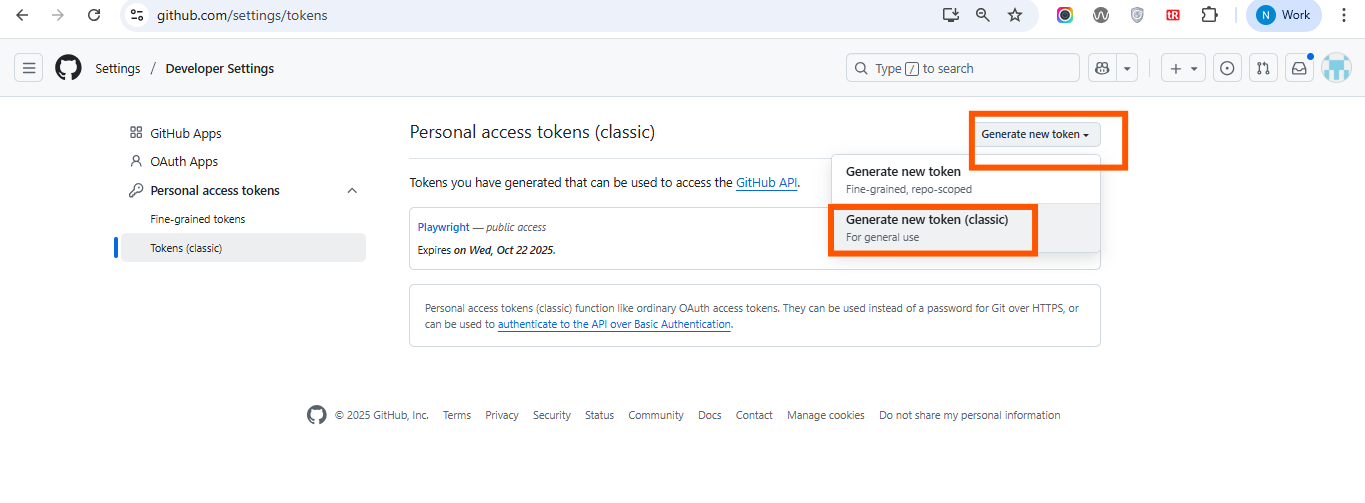


### Step 2: Open Personal Access Tokens

1. In the left sidebar under **Developer settings**, click:
   * **"Personal access tokens"**
2. Then click:
   * **"Tokens (classic)"**
   * 

### Step 3: Generate New Token

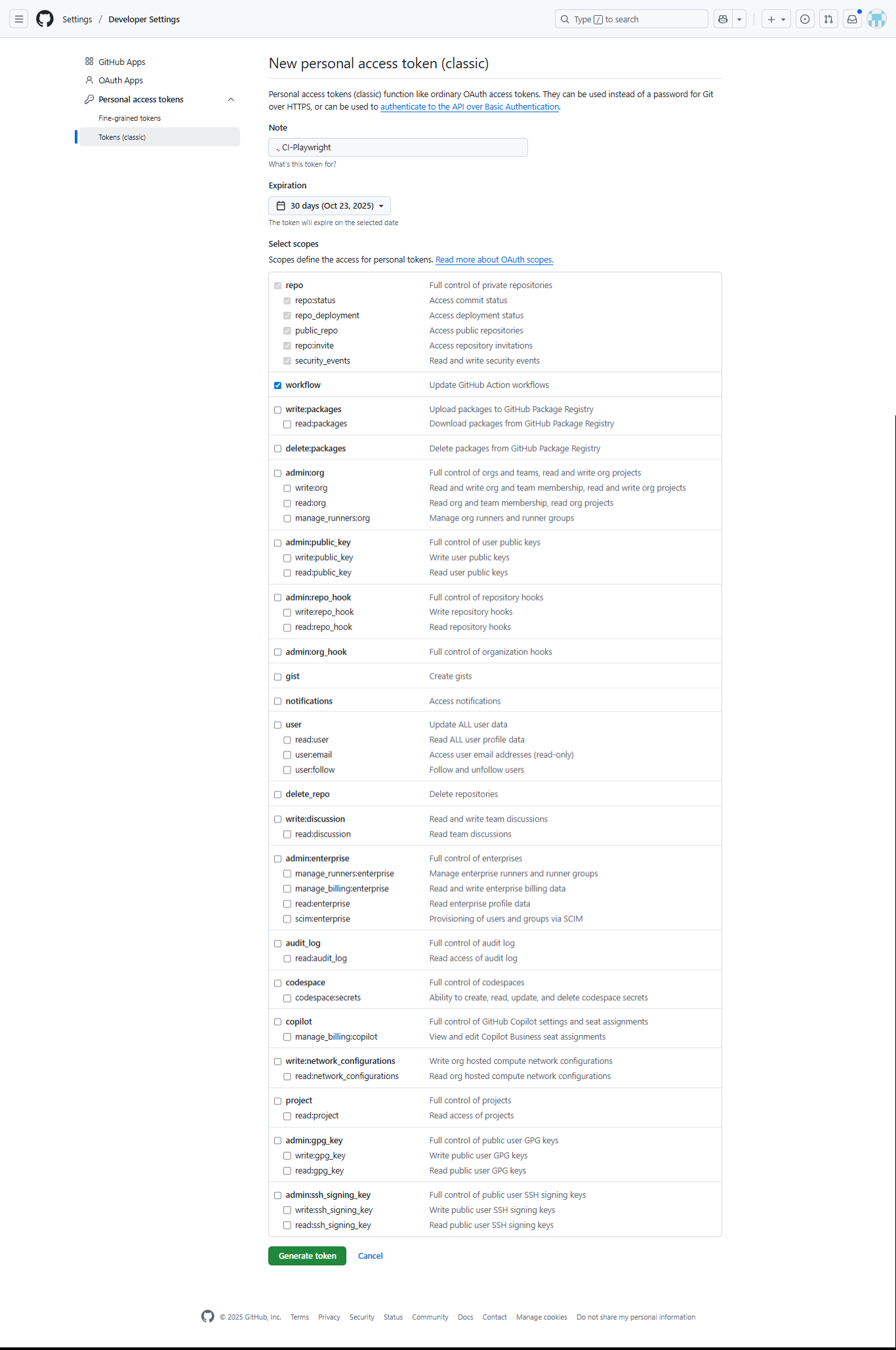
1. Click the **"Generate new token"** button
2. Choose:
   * **"Generate new token (classic)"**

****

### Step 4: Set Token Details

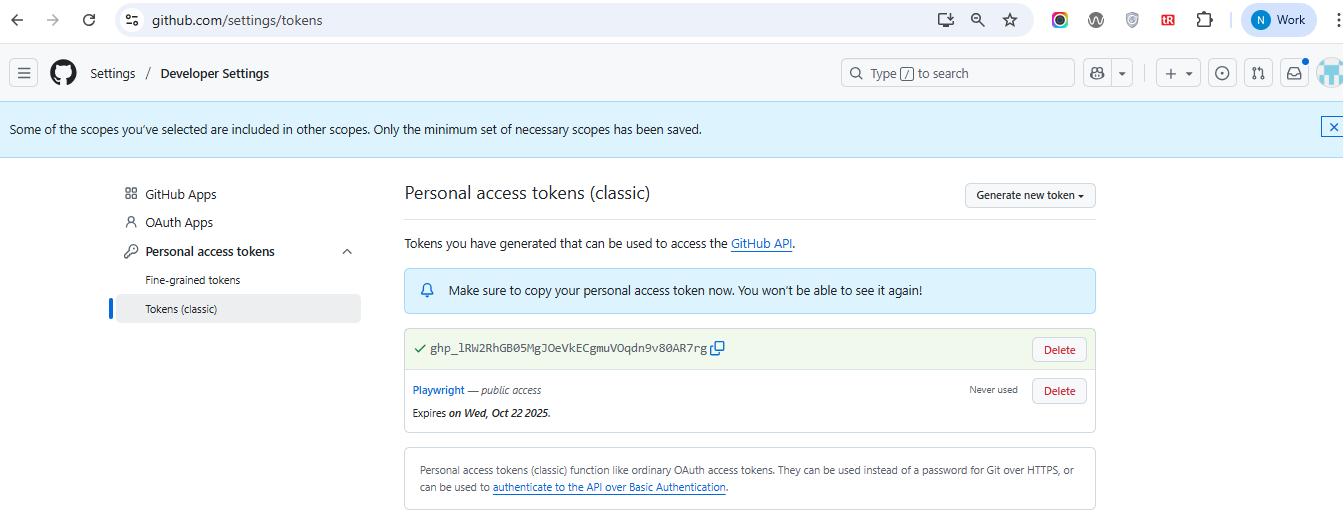
1. **Note**: Give your token a name (e.g., CI-Playwright)
2. **Expiration**: Select how long the token should be valid (e.g., 30 or 90 days)
3. **Scopes** – Tick the following checkboxes:
   * repo (Full control of private repositories)
   * workflow (To access GitHub Actions)
   * read:org (Optional – if working with organizations)

ℹ️ For basic CI/CD and pushing code, repo + workflow are usually enough.



### Step 5: Generate and Copy the Token

1. Click the **green "Generate token"** button at the bottom
2. **Copy the token immediately!**
   * ⚠️ You won’t be able to view it again later.
   * Save it in a safe place (e.g., a password manager or environment variable)



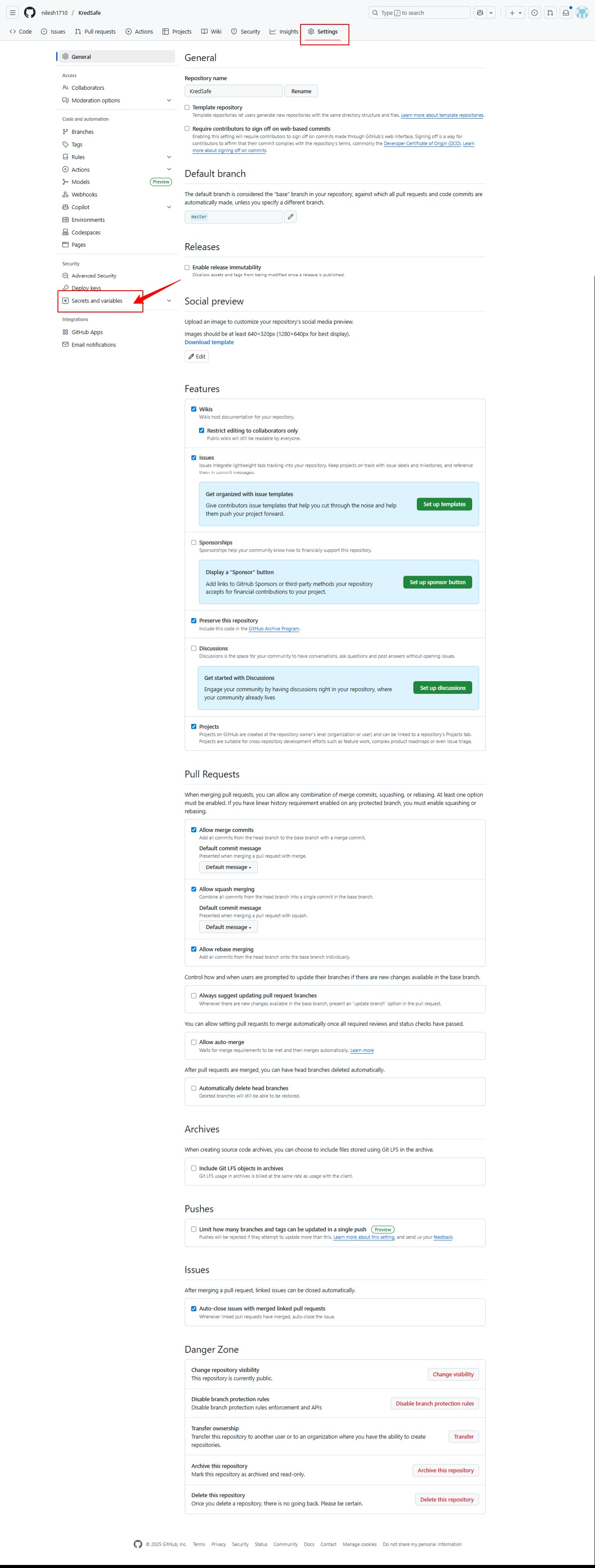
### Step 6: Use the Token

#### Option 1: In the Terminal (for Git push)

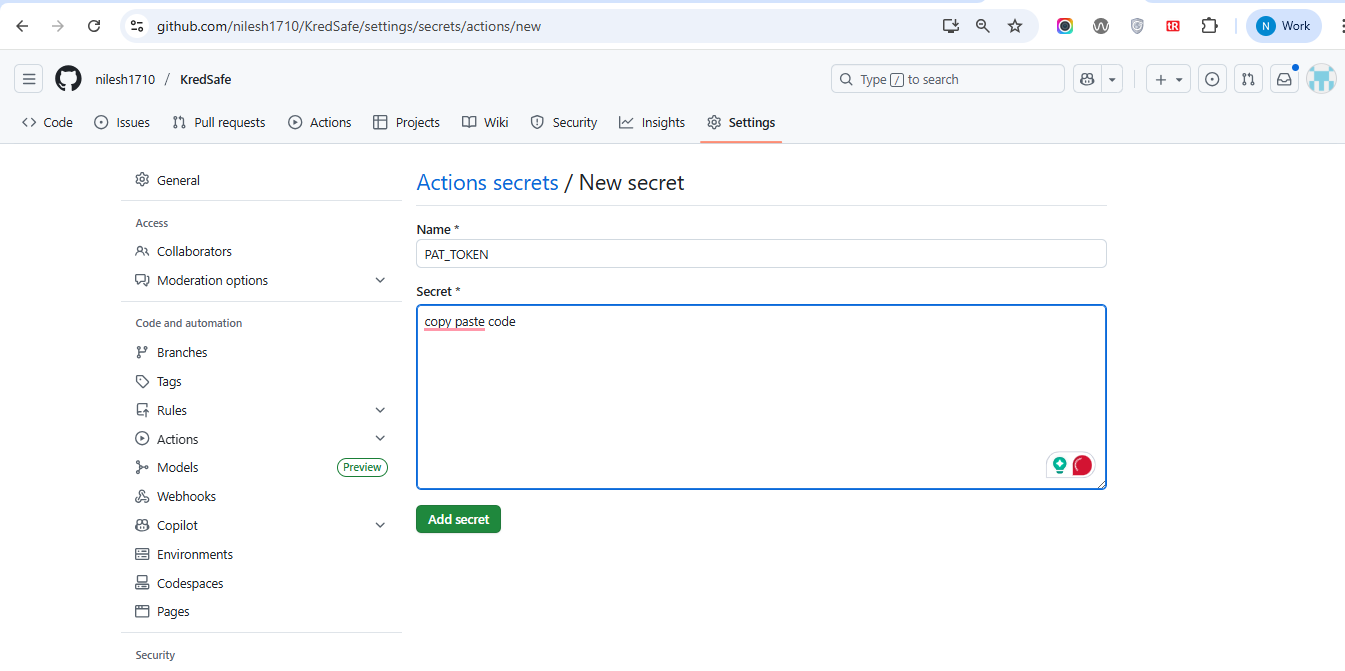
* When pushing to GitHub via HTTPS:
  + Use your GitHub **username**
  + Paste the **token** as the password

#### Option 2: In GitHub Actions

1. Go to your repo → **Settings** → **Secrets and variables** → **Actions**



1. Click **"New repository secret"**
2. Add:
   * Name: GH\_PAT
   * Value: (paste your token)



### Git Push Commands

1. **Go to your local repo folder:**

cd path/to/your/local/repo

1. **Check what changed:**

git status

1. **Stage all changed/new files:**

git add .

1. **Commit the changes with a message:**

git commit -m "Your commit message"

1. **Push to the remote repository (usually master or main branch):**

git push origin master

### My git hub

cd C:\Users\Admin\AppData\Local\ms-playwright\tests-examples\KredSafe

git add .

git commit -m "Add/update Playwright test scripts"

git push origin master

git add .github/workflows/playwright.yml

git commit -m "Add GitHub Actions workflow for Playwright tests"

git push origin master

**Advantages of Playwright**

1. **Cross-browser Testing**  
   Supports Chromium (Chrome, Edge), Firefox, and WebKit (Safari), enabling tests across all major browsers.
2. **Multi-language Support**  
   Official libraries for JavaScript/TypeScript, Python, Java, and C#.
3. **Auto-waiting Mechanism**  
   Automatically waits for elements to be ready before actions, reducing flaky tests.
4. **Powerful Selectors**  
   Supports CSS, XPath, text selectors, and custom selectors like role-based accessibility queries.
5. **Headless and Headful Modes**  
   Tests can run visually or in the background.
6. **Parallel Execution & Test Isolation**  
   Built-in test runner supports parallel test execution and isolates tests by default.
7. **Mobile Emulation & Geolocation**  
   Can simulate mobile devices and control geolocation, permissions, and network conditions.
8. **Rich Debugging Tools**  
   Offers tracing, screenshots, videos, and inspector tools to debug flaky tests.
9. **Network Interception & Mocking**  
   Intercept and modify network requests/responses easily.
10. **CI/CD Friendly**  
    Easy integration with GitHub Actions, Jenkins, CircleCI, and more.

**Disadvantages of Playwright**

1. **Newer Tool**  
   Compared to Selenium, it’s relatively new, so community support and integrations might be smaller.
2. **Larger Initial Setup**  
   Requires downloading browser binaries (~200-300MB) which might increase build time in CI.
3. **Learning Curve**  
   Advanced features require understanding asynchronous code and Playwright API specifics.
4. **Limited Legacy Browser Support**  
   Doesn’t support older browsers like Internet Explorer.
5. **API Changes**  
   As it’s actively developed, APIs can change, requiring occasional updates to your tests.

**Recommended For**

* **Cross-browser testing** on modern browsers (Chromium, Firefox, WebKit/Safari).
* Teams that use **JavaScript/TypeScript**, **Python**, **Java**, or **C#** and want official support.
* Projects needing **reliable, fast, and parallelized end-to-end (E2E) tests**.
* Developers who want **built-in auto-waiting** to reduce flaky tests.
* Testing complex web apps with **dynamic content, SPA frameworks** (React, Angular, Vue).
* Automating **mobile browser testing** (via device emulation).
* Teams using **CI/CD pipelines** and wanting good integration out of the box.

**Not Recommended For**

* Projects that require **testing legacy or very old browsers** like Internet Explorer.
* Teams that primarily use **non-supported languages** outside of Playwright’s official set.
* Simple UI tests where a lightweight tool or record-and-playback tool (like Selenium IDE) suffices.
* Cases where your team needs a **massive ecosystem of plugins** and integrations that Selenium currently dominates.
* Testing **non-web platforms** (e.g., native mobile apps) — Playwright focuses on web browsers.
* Very large enterprise environments already heavily invested in other mature frameworks (Selenium, TestCafe) unless you’re planning migration.

## Conclusion

Playwright is a modern, powerful, and versatile end-to-end testing framework designed for reliable cross-browser web automation. It excels in handling complex, dynamic web applications with robust auto-waiting, parallel test execution, and native support for Chromium, Firefox, and WebKit browsers. Its rich API, multi-language support, and CI/CD friendliness make it an excellent choice for teams aiming for fast, stable, and maintainable test suites.

However, it’s less suitable if you need to support legacy browsers like Internet Explorer or require a broader plugin ecosystem that more mature tools offer. For most modern web projects, especially those using JavaScript/TypeScript or Python, Playwright provides a cutting-edge solution for comprehensive and efficient automated testing.