



How to use AI with Playwright for UI & API Testing

Michał Ślęzak

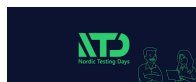
About me

Michał Ślęzak

Test Architect at Sii
Consultant, Trainer

mslezak@sii.pl
www.sii.pl/en

STAR EAST
A TECHWELL EVENT



testingplus.me





Sii – the leading technology consulting, digital transformation engineering, and business services vendor in Poland

Establishment

Since 2006

Shareholders: Sii France = 70%,
Gregoire Nitot (Founder & Sii Poland CEO) = 30%

7 300+

Power People and still growing

EUR 500M
PLN 2.1B

Yearly revenue

delivery centers in the biggest
cities in Poland

14





Testing

Our key strengths

1 500+ specialists

QA & Test Engineers

95% ISTQB

Certified engineers

350+ clients & impressive references

from retail, banking, healthcare, automotive and many other industries

End-to-end testing services

best testing practices and cutting-edge technologies

AI-powered solutions

from automation to reporting

Partnerships with market leaders

recognized by Gartner – partnering with industry leaders featured in their reports

Selected technologies



Partnerships



Agenda



Introduction
to Playwright



Ideas of using
AI & DEMOS &
Challenges



Q&A



What is Playwright?

PLAYWRIGHT



Often updated

API / UI tests

Created by Microsoft

Open source

Supports:

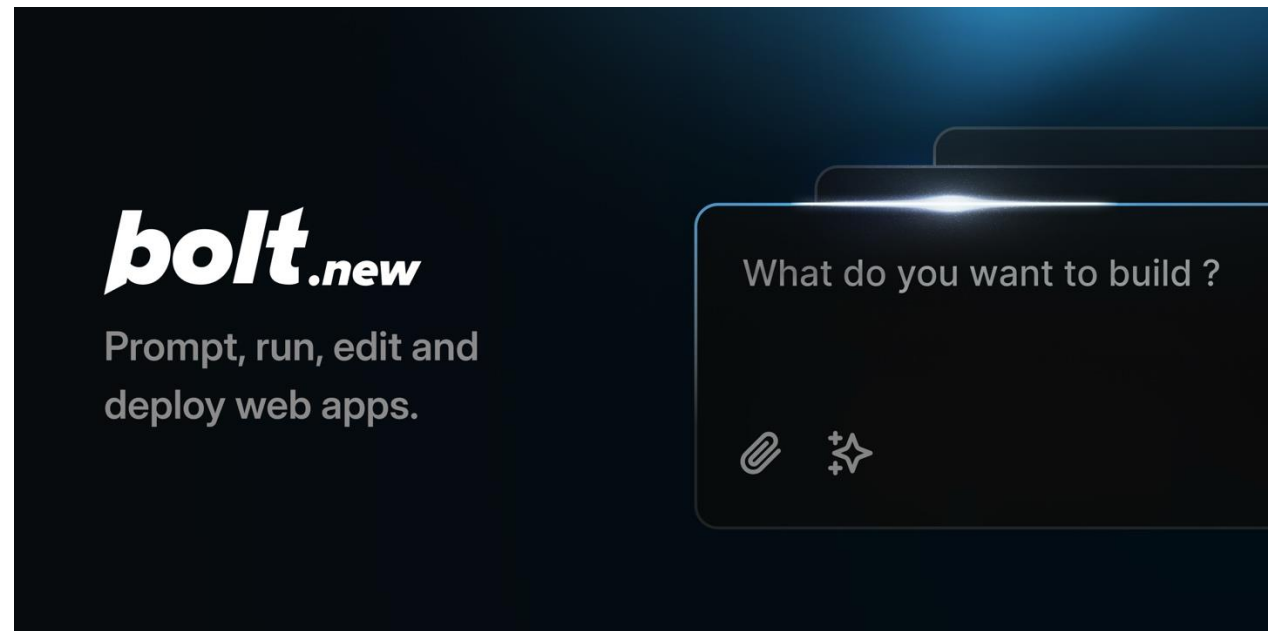
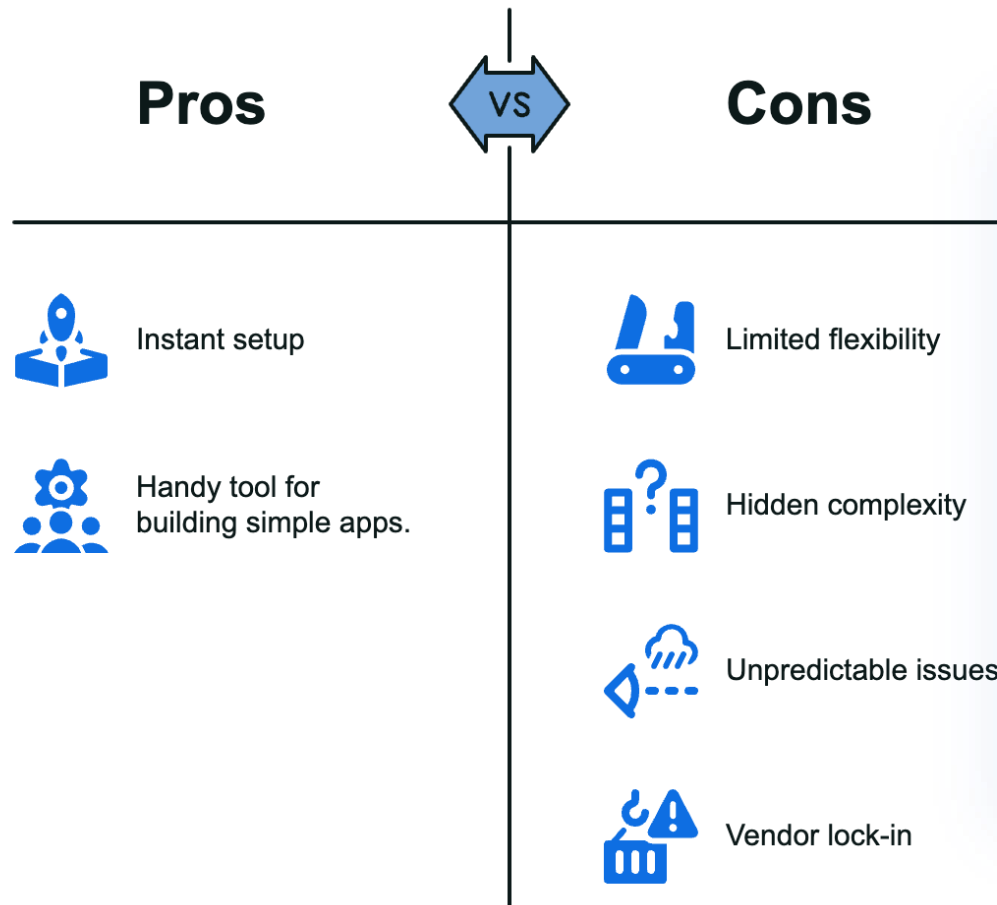
JS /TS

Python

C#

Java

Bolt.new – creating test apps in minutes



F Create an application to learn Playwright (Microsoft's testing tool).

The application should include:

A simple API

A basic database

UI controls such as a calendar, sliders, a canvas, dropdowns, and similar elements

The page will be used solely for my learning purposes.

Thinking

6s



Based on your requirements, I'll create a Playwright Testing Playground application that serves as an ideal environment for learning Playwright's capabilities. This first version will include various UI elements, a simple API, and a basic database to provide a comprehensive testing playground.

Core Features:

- Interactive UI elements dashboard with multiple testing targets (forms, calendar, dropdowns, sliders, canvas)
- Simple REST API endpoints to practice API testing scenarios

How can Bolt help you today?



Code Preview

5173 /



Playwright Playground

Dashboard



Dashboard



Form Elements



Calendar



Sliders



Canvas



API Testing



Authentication

Playwright Testing Playground

Welcome to your interactive testing environment. This application contains various UI elements, API endpoints, and testing scenarios to help you learn and practice Playwright testing.

Playwright Docs

API Status: Online

Test Components



Form Elements

Practice testing various form controls including inputs, checkboxes, and select menus.

Basic

Explore →



Calendar

Test date selection, range picking, and calendar navigation.

Intermediate

Explore →



Sliders

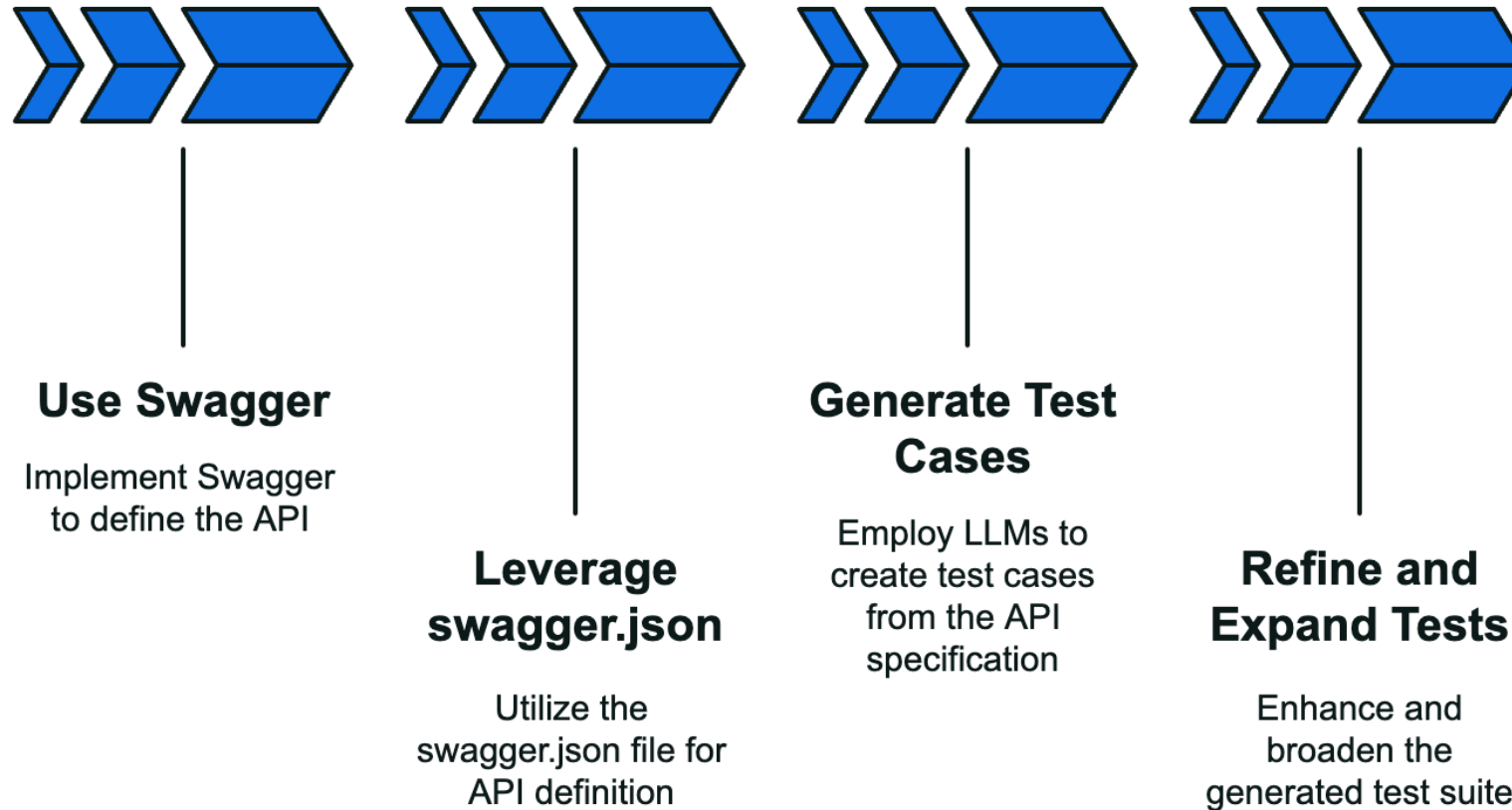
Practice testing range inputs and custom slider components.



Canvas

Learn how to test canvas-based drawing applications.

Automated API Testing with Swagger and LLMs



```
{
  "openapi": "3.0.0",
  "info": {
    "title": "Trello REST API",
    "version": "0.0.1",
    "servers": [
      {
        "url": "https://api.trello.com/1",
        "components": {
          "securitySchemes": {
            "bearerAuth": {
              "type": "apiKey",
              "in": "header",
              "name": "Authorization"
            }
          }
        }
      }
    ]
  }
}
```

M Using the Trello OpenAPI specification, create an automated test in Playwright with TypeScript that performs the following steps: logs into the application, creates a new board, and then deletes it. Structure the code according to SOLID principles for maintainability and scalability.

Edit

I'll create a Playwright test with TypeScript that follows SOLID principles to interact with the Trello API. Based on the OpenAPI specification, I'll implement the board creation and deletion operations.

Trello Automated Test with SOLID Principles
Code

```
// 1. First, let's define interfaces for our core abstractions
// ApiClient interface - abstraction for API interactions
interface ApiClient {
  request<T>(endpoint: string, method: string, data?: Record<string, any>): Promise<T>;
}
```

I've created a comprehensive Playwright test suite with TypeScript that follows SOLID principles for testing Trello board creation and deletion. Here's a breakdown of the implementation:

Key Features:

1. SOLID Principles Implementation:

- **Single Responsibility:** Each class has one specific responsibility
 - `TrelloAuthService`: Handles authentication
 - `PlaywrightApiClient`: Manages API requests
 - `TrelloBoardServiceImpl`: Handles board operations
 - `TrelloAutomatedTest`: Orchestrates the test flow
- **Open/Closed:** Classes are open for extension but closed for modification
- **Liskov Substitution:** Implementations can be substituted without breaking the system
- **Interface Segregation:** Focused interfaces for different concerns
- **Dependency Inversion:** High-level modules depend on abstractions

Reply to Claude...



Claude 3.7 Sonnet v



// 1. First, let's define interfaces for our core abstractions

// ApiClient interface - abstraction for API interactions

```
interface ApiClient {
  request<T>(endpoint: string, method: string, data?: Record<string, any>): Promise<T>;
}
```

// TrelloBoardService interface - handles board operations

```
interface TrelloBoardService {
  createBoard(name: string, desc?: string): Promise<Board>;
  deleteBoard(boardId: string): Promise<void>;
}
```

// AuthService interface - handles authentication

```
interface AuthService {
  authenticate(): Promise<void>;
  getAuthHeaders(): Record<string, string>;
  getAuthQueryParams(): Record<string, string>;
}
```

// Data models

```
interface Board {
  id: string;
  name: string;
  desc?: string;
  closed?: boolean;
  shortUrl?: string;
}
```

// 2. Implement concrete classes

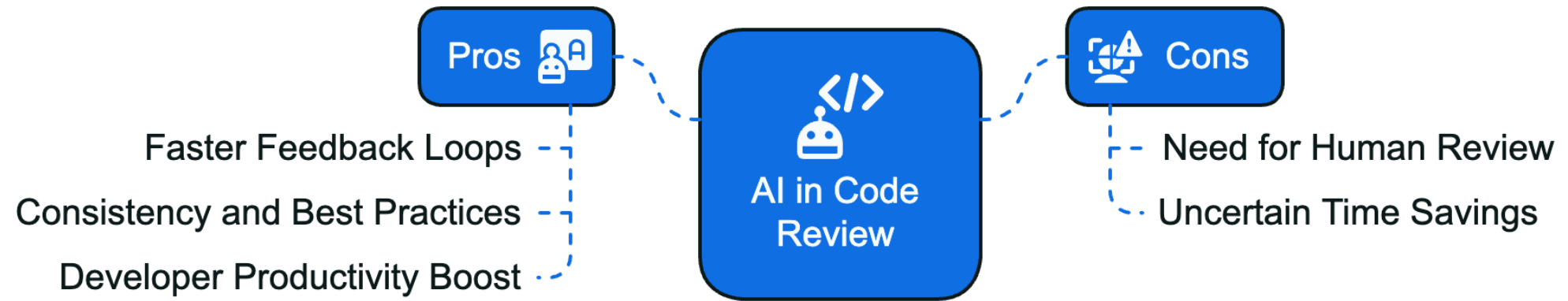
// TrelloAuthService - handles Trello API authentication using API key and token

```
class TrelloAuthService implements AuthService {
  private readonly apiKey: string;
  private readonly apiToken: string;

  constructor(apiKey: string, apiToken: string) {
    this.apiKey = apiKey;
    this.apiToken = apiToken;
  }

  async authenticate(): Promise<void> {
    // Validate that credentials are provided
    if (!this.apiKey || !this.apiToken) {
```

```
    }
  }
}
```



Add a test for login.spec.ts #1

Edit

<> Code

Open

mslezaksii wants to merge 1 commit into `main` from `add_a_test_for_login_form`

Conversation 10

Commits 1

Checks 0

Files changed 4

+204 -0



mslezaksii commented 5 days ago • edited by qodo-merge-pro bot

PR Type

Tests, Enhancement

Description

- Add Playwright end-to-end test for login functionality
- Introduce Playwright configuration for multi-browser testing
- Add Playwright and Node types as development dependencies
- Initialize project with basic package metadata

Changes walkthrough

	Relevant files
Dependencies	<div><div>▶ package.json</div><div>Initialize project and add Playwright dependencies</div><div>+14/-0</div></div>
Configuration changes	<div><div>▶ playwright.config.ts</div><div>Add Playwright test configuration for multiple browsers</div><div>+79/-0</div></div>
Tests	<div><div>▶ login.spec.ts</div><div>Add end-to-end login test using Playwright</div><div>+14/-0</div></div>

▶ Need help?

Reviewers

Copilot

Still in progress? [Convert to draft](#)

Assignees

No one—[assign yourself](#)

Labels

Review effort 2/5

Qodo Merge filters

Show All

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close these issues.

None yet

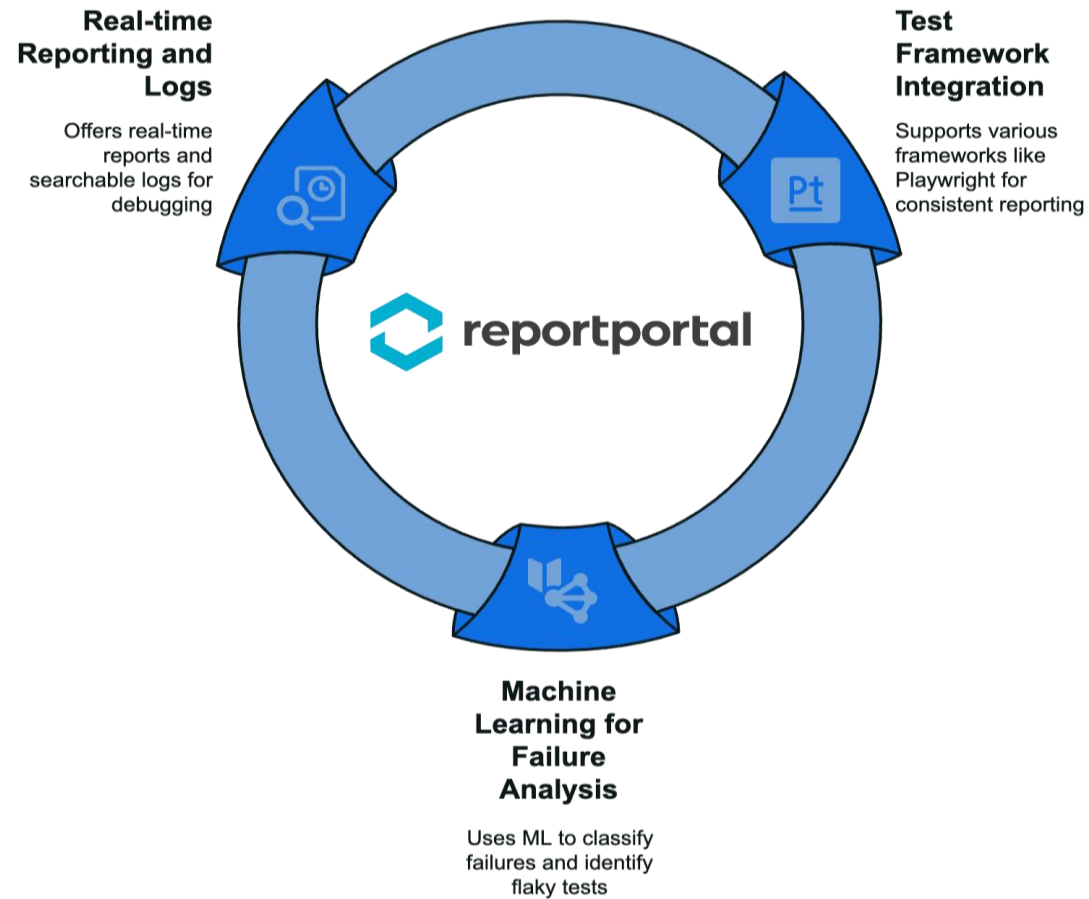
Notifications

Customize

Unsubscribe

You're receiving notifications because you're watching this repository.

ReportPortal & Playwright



PLAYWRIGHTREPO

factories

TS BoardFactory.ts

U

TS CardFactory.ts

U

fixtures

TS trelloFixture.ts

U

interfaces

TS Board.ts

U

TS Card.ts

U

TS List.ts

U

lib

TS trelloAPI.ts

U

TS TrelloBoardAssertio...

U

TS TrelloSettings.ts

U

node_modules

page

TS TrelloBoardPage.ts

U

TS TrelloHomePage.ts

U

TS TrelloLoginPage.ts

U

playwright-report

test-results

tests / trelloApi

TS trelloBoardTests.spe...

U

TS trelloBoardTests.spec.ts U

TS BoardFactory.ts U

{} package.json M

TS playwright.config.ts M

TS playwright.config_copy.ts U

{} trelloSettings.json U

1

tests > trelloApi > TS trelloBoardTests.spec.ts > ...

3

test.describe('Trello API and UI Tests', () => {

6

}, async ({

20

await cardFactory.createCards(cardTitles, board.id);

21

await trelloAPI.createList(newColumnName, board.id);

22

23

await trelloLoginPage.go(settings.baseUrl);

24

await trelloLoginPage.login(settings.username, settings.password);

25

await trelloBoardPage.navigateToBoard(board.url);

26

27

await trelloBoardPage.moveCardsToColumn(cardTitles, newColumnName);

28

await trelloBoardAssertions.verifyCardsInColumn(cardTitles, newColumnName);

29

});

30

31

test('Check is it possible to login correctly', {

32

}, async ({

33

trelloLoginPage, settings

34

}) => {

35

await trelloLoginPage.go(settings.baseUrl);

36

await trelloLoginPage.login(settings.username, settings.password);

37

await expect(async () => {

38

await trelloLoginPage.getAllBoardsVisibleStatus();

39

}).toPass();

40

});

41

});

Run | Debug | Inspect | Execute Playwright Test

Problems

Output

Debug Console

Test Results

Ports

Playwright

TERMINAL

playwright

micah_sle@MacBook-Pro-Micha-2 playwrightRepo % npx playwright test "trelloBoardTests.spec.ts" -g

TEST RESULTS

* Test output is only available for new test runs.

Playwright Test for VSCode

⊗ Check is it doable to login correctly

> 15 older results

NOTEPADS

OUTLINE

TIMELINE

add_code_tests*

8 ↓ 0 ↑

0 0 0

0

⌕

Ln 41, Col 4

Spaces: 2

UTF-8

LF

TypeScript

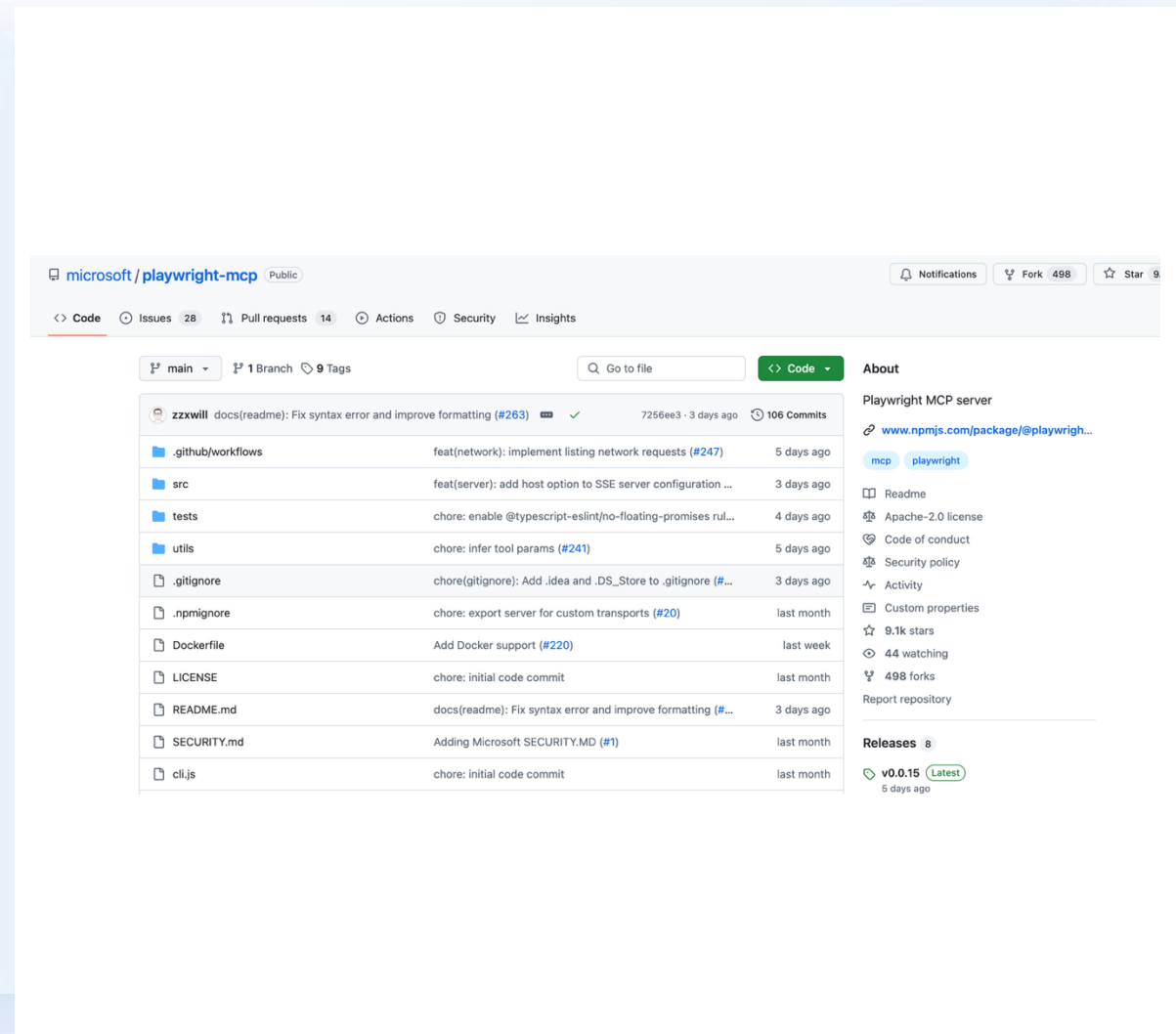
Prettier

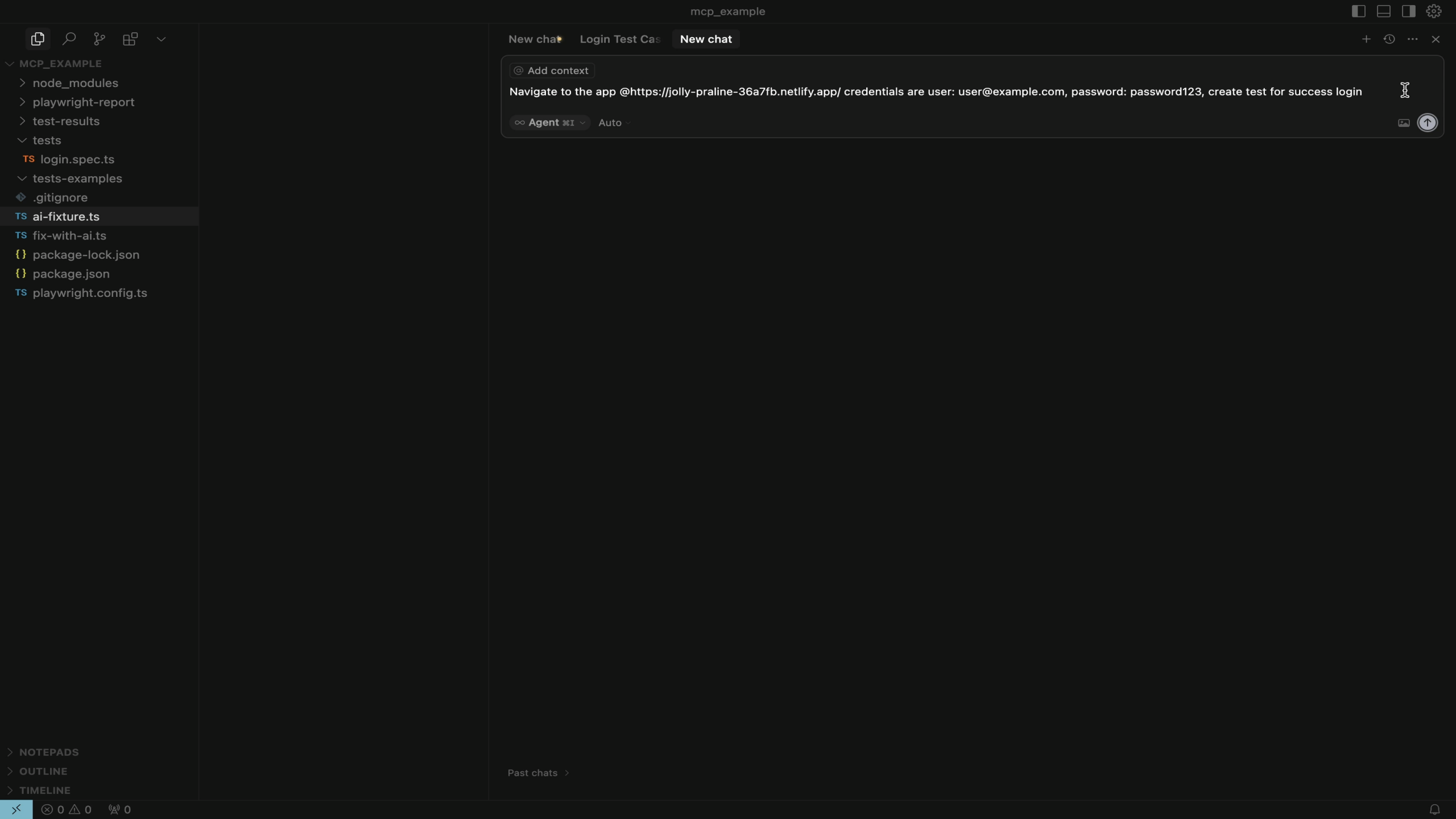
Playwright MCP

How it works: LLMs can send structured instructions via the MCP protocol to remotely control Playwright, enabling dynamic, AI-driven interactions with web apps (e.g. filling forms, clicking buttons, validating UI).

What is MCP: MCP (Model Context Protocol) is a communication protocol that enables Playwright to execute tests remotely and in parallel across multiple browsers and platforms using a unified API.

Modern Cross-Platform Testing: Playwright MCP enables end-to-end testing across all modern browsers (Chromium, Firefox, WebKit) on multiple platforms (Windows, Linux, macOS) with a single API.





Playwright MCP – Generated Code

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

```
const ROOT_URL = 'https://jolly-praline-36a7fb.netlify.app/';
```

```
const VALID_EMAIL = 'user@example.com';
```

```
const VALID_PASSWORD = 'password123';
```

Run Test | Watch Test | Run | Debug | Inspect | ▶ Execute Playwright Test

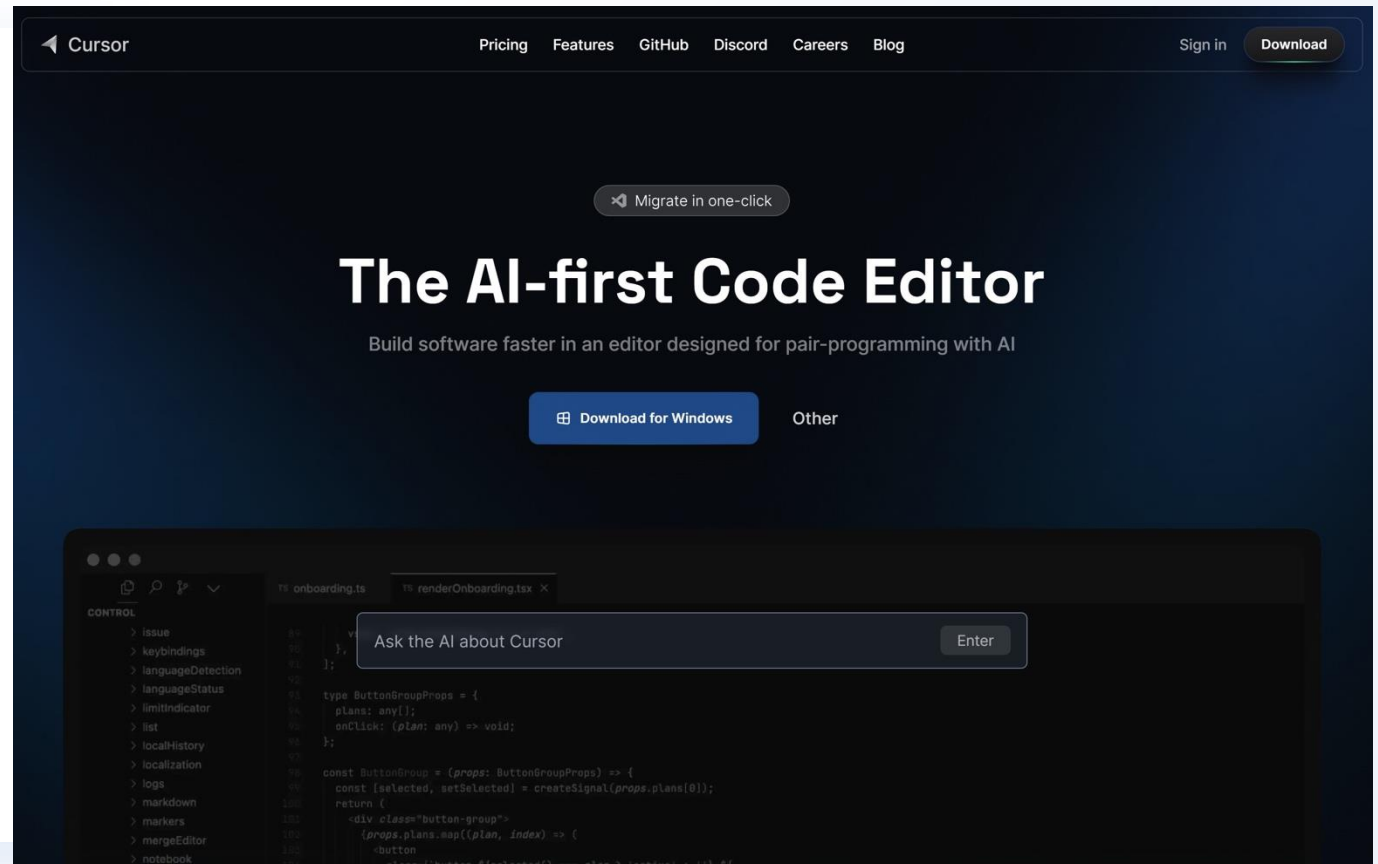
```
test('successful login redirects to dashboard and shows welcome message', async ({ page }) => {  
  await page.goto(ROOT_URL);  
  await page.getByLabel('Email Address').fill(VALID_EMAIL);  
  await page.getByLabel('Password').fill(VALID_PASSWORD);  
  await page.getByRole('button', { name: 'Sign in' }).click();  
  await expect(page).toHaveURL(/.*\/dashboard/);  
  await expect(page.getByRole('heading', { name: 'Welcome, Demo User!' })).toBeVisible();  
});
```

Playwright & Cursor IDE

Flexible AI Control: Developers can tweak AI settings (model, temperature, context) to fine-tune how the assistant supports coding tasks.

AI-First Editing: Cursor embeds AI deeply into the coding workflow, helping with code generation, refactoring, and debugging directly in the editor.

Lightweight and Familiar: Built on top of VS Code, so it feels instantly familiar while offering a faster and more focused experience.



PLAYWRIGHTTESTSEXA...
node_modules
playwright-report
test-results
tests
pages
Ts LoginPage.ts U
Ts login.spec.ts
tests-examples
.gitignore
ai-fixture.ts U
fix-with-ai.ts U
package-lock.json
package.json
playwright.config.ts

tests > pages > TS LoginPage.ts > LoginPage > login
1 import { Page, Locator } from '@playwright/test';
2
3 export class LoginPage {
4 readonly page: Page;
5 readonly emailInput: Locator;
6 readonly passwordInput: Locator;
7 readonly signInButton: Locator;
8 readonly welcomeHeading: Locator;
9
10 constructor(page: Page) {
11 this.page = page;
12 this.emailInput = page.getByRole('textbox', { name: 'Email Address' });
13 this.passwordInput = page.getByRole('textbox', { name: 'Password' });
14 this.signInButton = page.getByRole('button', { name: 'Sign in' });
15 this.welcomeHeading = page.getByRole('heading', { name: 'Welcome, Demo User!' });
16 }
17
18 async goto() {
19 await this.page.goto('https://jolly-praline-36a7fb.netlify.app');
20 }
21
22 async login(email: string, password: string) {
23 await this.emailInput.click();
24 await this.emailInput.fill(email);
25 await this.emailInput.press('Tab');
26 await this.passwordInput.fill(password);
27 await this.signInButton.click();
28 }
29
30 async isWelcomeVisible() {
31 return this.welcomeHeading.isVisible();
32 }
33 }

New chat

@ TS LoginPage.ts

Refactor the code to align with SOLID principles. Add an ESLint configuration. Move the URL to a separate JSON settings file and load it from there. Initialize selector values directly in the field declarations instead of in the constructor.

Agent AI Auto

↑

Past chats >

Problems Output Debug Console Terminal Ports Playwright

Filter (e.g. text, **/*.ts, !**/node_modules/**)

No problems have been detected in the workspace.

NOTEPADS
OUTLINE
TIMELINE

Creating a Page Object by LLM

Generate Page Objects from Defined HTML Code

1. Provide some HTML input to the LLM.
2. Add the following prompt:
"Based on this HTML, create a Page Object for Playwright (TypeScript version), following SOLID principles."

Welcome Back

Sign in to access your account

Email Address

Password

☐ Remember me [Forgot password?](#)

Sign in

Don't have an account? [Sign up now](#)

Demo credentials: user@example.com / password123

**HTML:**

```
<div class="flex min-h-screen flex-col items-center justify-center bg-
gradient-to-br from-gray-100 to-gray-200 px-4 py-12 sm:px-6 lg:px-8"> <div
class="w-full max-w-md animate-fade-in"> <div class="card overflow-hidden">
<div class="bg-primary-600 px-6 py-8 text-center"> <h1 class="text-2xl font-
semibold text-white">Welcome Back</h1> <p class="mt-2 text-primary-100">Sign
in to access your account</p> </div> <div class="bg-white px-6 py-8"> <form
class="space-y-6"> <div class="space-y-4"> <div class="form-group"> <label
for="email" class="form-label block text-sm font-medium text-gray-700">Email
Address</label> <div class="relative"> <input id="email" class="input border-
gray-300 focus:ring-primary-500" type="email" placeholder="you@example.com"
autocomplete="email" required> </div> </div> <div class="form-group"> <label
for="password" class="form-label block text-sm font-medium text-gray-
700">Password</label> <div class="relative"> <input id="password" class="input
border-gray-300 focus:ring-primary-500" type="password" placeholder="....."
autocomplete="current-password" required> </div> </div> </div> <div
class="flex items-center justify-between"> <div class="flex items-center">
<input id="remember-me" name="remember-me" type="checkbox" class="h-4 w-4
rounded border-gray-300 text-primary-600 focus:ring-primary-500"> <label
for="remember-me" class="ml-2 block text-sm text-gray-700">Remember me</label>
</div> <a href="#" class="text-sm font-medium text-primary-600 hover:text-
primary-500">Forgot password?</a> </div> <button class="btn bg-primary-600
text-white hover:bg-primary-700 h-12 rounded-md px-6 text-base w-full"
type="submit"> Sign in </button> </form> <div class="mt-6 text-center text-sm
text-gray-500"> <p>Don't have an account? <a href="#" class="font-medium text-
primary-600 hover:text-primary-500">Sign up now</a></p> </div> </div> <div
class="mt-6 text-center text-sm text-gray-500"> <p>Demo credentials:
user@example.com / password123</p> </div> </div> </div> </div>
```

Based on the provided HTML snippet, here's a Page Object Model (POM) for the login page using Playwright:

Reply to Claude...



Playwright Copy Prompt

- It's available in Playwright report
- In report we have an option to copy logs like a prompt
- Useful way to put details if we don't use Cursor or Copilot

☀ Good afternoon, michal

Instructions

- Following Playwright test failed.
- Explain why, be concise, respect Playwright best practices.
- Provide a snippet of code with the fix, if possible.

Test info

- Name: Login >> Failed test- should login successfully with valid credentials
- Location:
/Users/michal_sle/Documents/artykućy_blog/playwright_cursor_example/mcp_example/tests/login.spec.ts:32:7

Error details

...

Error: Timed out 5000ms waiting for expect(locator).toHaveURL(expected)
 Locator: locator(':root')
 Expected pattern: /*\s*/dashboard/
 Received string: "https://jolly-praline-36a7fb.netlify.app/login"
 Call log:



Claude 3.7 Sonnet

4



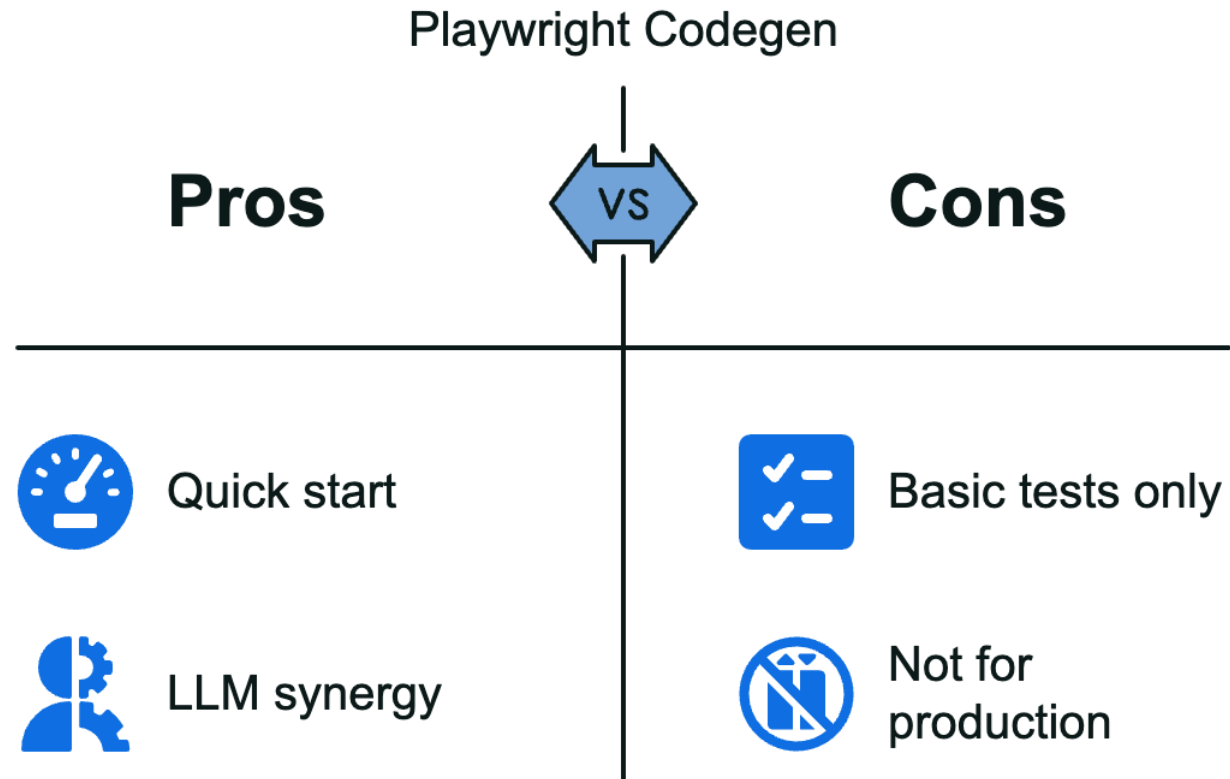
Errors

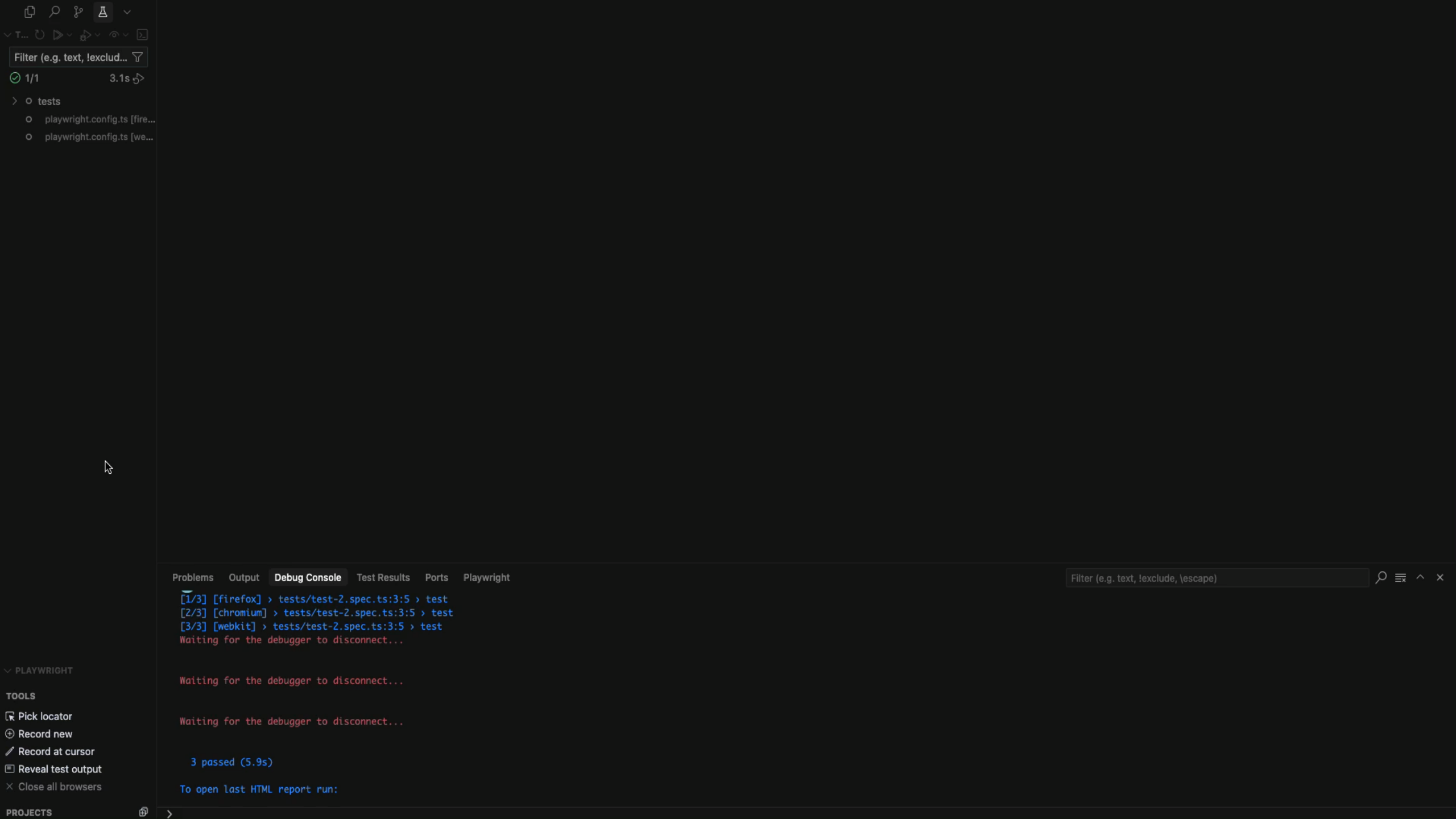
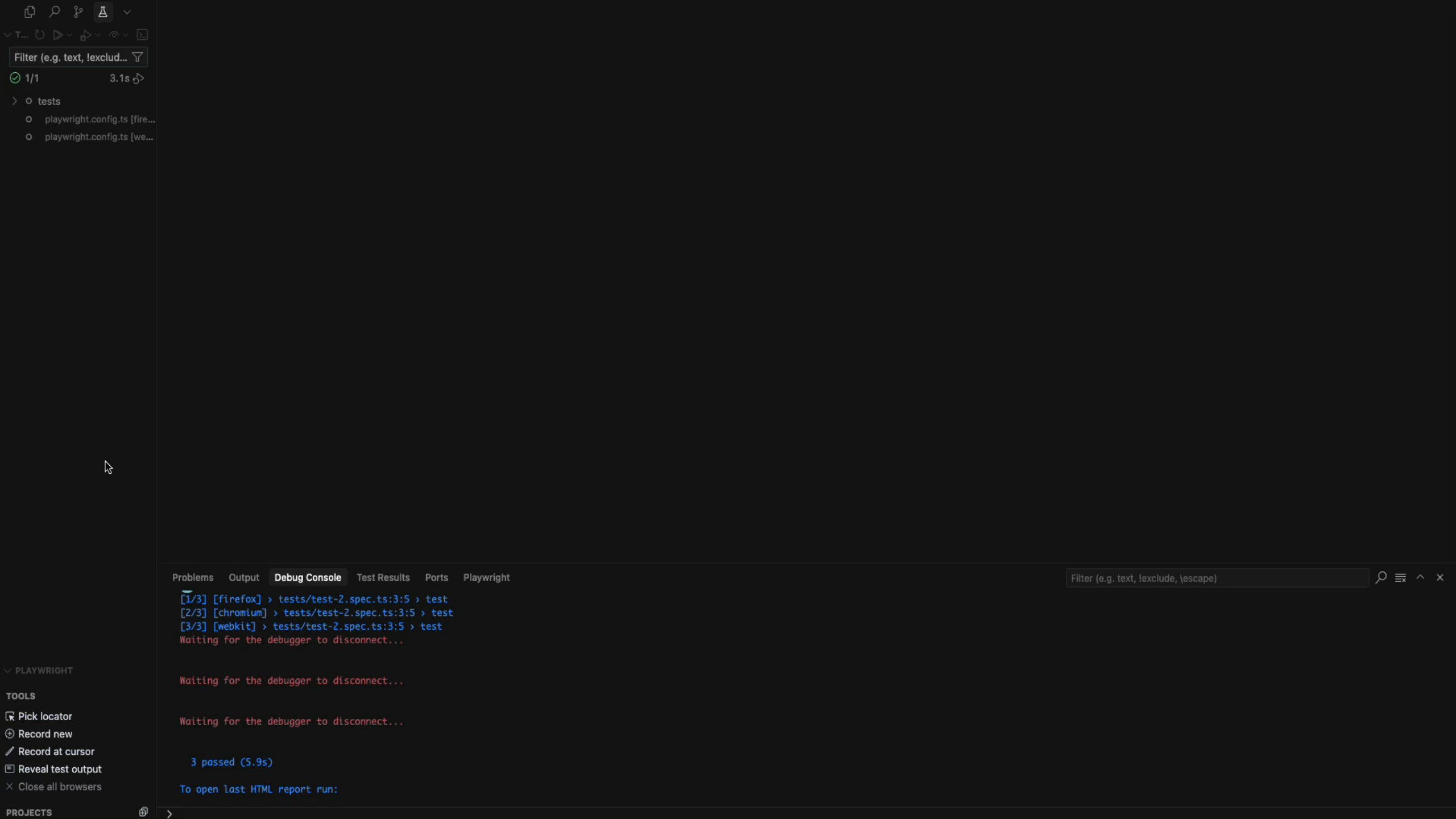
Error: Timed out 5000ms waiting for expect(locator).toHaveURL(expected)

Copy prompt

Locator: locator(':root')
 Expected pattern: /*\s*/dashboard/
 Received string: "https://jolly-praline-36a7fb.netlify.app/login"
 Call log:
 - expect.toHaveURL with timeout 5000ms
 - waiting for locator(':root')
 9 x locator resolved to <html lang="en">...</html>
 - unexpected value "https://jolly-praline-36a7fb.netlify.app/login"

```
36 |     await page.getByRole('button', { name: 'Sign in' }).click();
37 |     // Expect to be redirected to dashboard and see welcome message
> 38 |     await expect(page).toHaveURL(/.*\s*/dashboard/);
    |                               ^
39 |     await expect(page.getByRole('heading', { name: 'Dashboard' })).toBeVisible();
40 |     await expect(page.getByRole('heading', { name: 'Welcome, Demo User!' })).toBeVisible();
41 |   });
    at /Users/michal_sle/Documents/artykućy_blog/playwright_cursor_example/mcp_example/tests/login.spec.ts:38:
```





- 
- PLAYWRIGHTTESTEXAMPLECR
- > node_modules

> playwright-report

> test-results

tests

TS login.spec.ts

TS test-1.spec.ts

tests-examples

.gitignore

TS ai-fixture.ts

TS fix-with-ai.ts

{ } package-lock.json

{ } package.json

TS playwright.config.ts

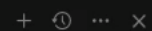
TS test-1.spec.ts U X



tests > TS test-1.spec.ts > ...

```
1 import { test, expect } from '@playwright/test';
2
3 > Execute Playwright Test | Run Test | Watch Test | Run | Debug | Inspect
4 test('test', async ({ page }) => {
5   await page.goto('https://jolly-praline-36a7fb.netlify.app/login');
6   await page.getByRole('textbox', { name: 'Email Address' }).click();
7   await page.getByRole('textbox', { name: 'Email Address' }).fill('user@example.com');
8   await page.getByRole('textbox', { name: 'Email Address' }).press('Tab');
9   await page.getByRole('textbox', { name: 'Password' }).fill('password123');
10  await page.getByRole('button', { name: 'Sign in' }).click();
11  await expect(page.getByRole('heading', { name: 'Welcome, Demo User!' })).toBeVisible();
12  await expect(page.getByRole('heading', { name: 'Welcome, Demo User!' })).toBeVisible();
13 });
```

New chat



@ TS test-1.spec.ts

Act as a Senior Test Automation Engineer with 10+ years of experience. Write a maintainable and scalable test using the Page Object Model and adhere to SOLID principles. Ensure the code demonstrates best practices in test structure, separation of concerns, and reusability.

∞ Agent #1 Auto



> NOTEPADS

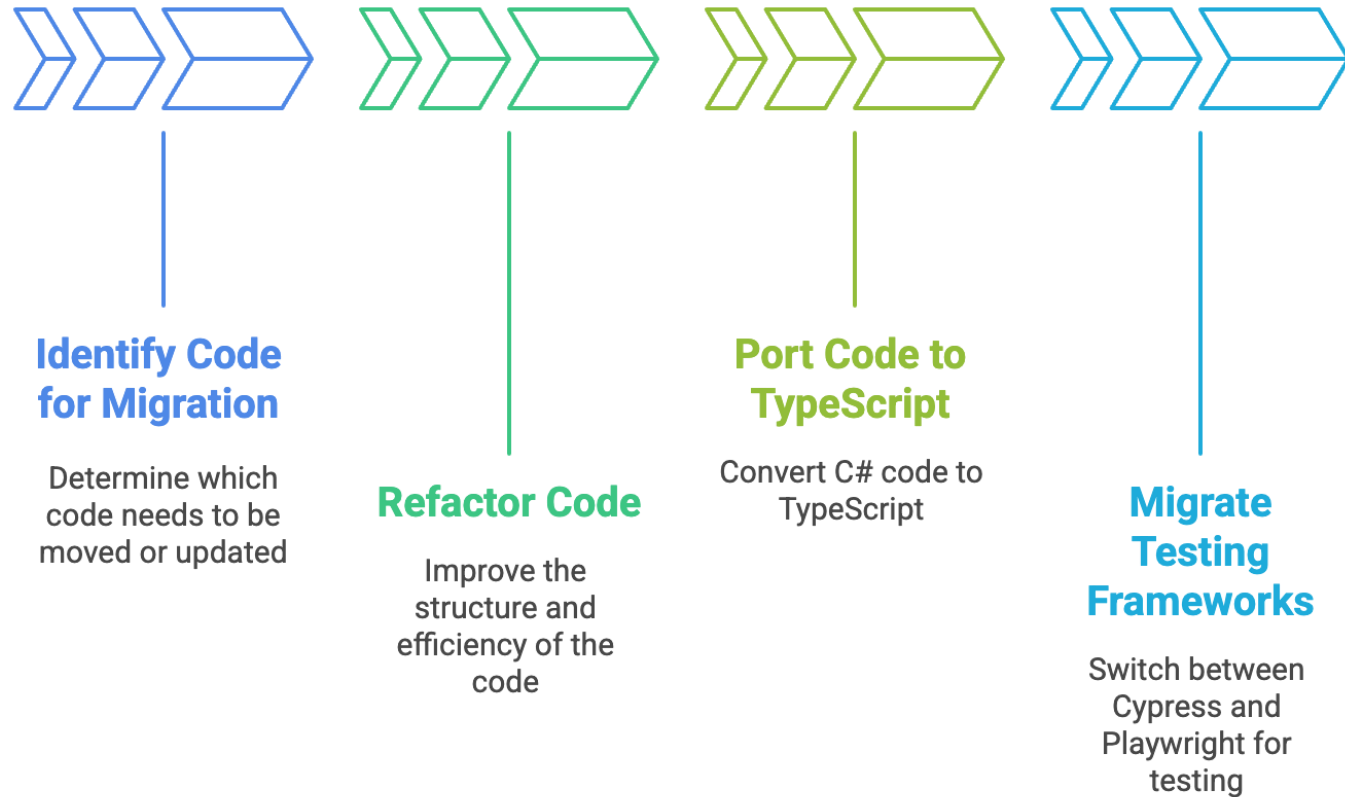
> OUTLINE

> TIMELINE

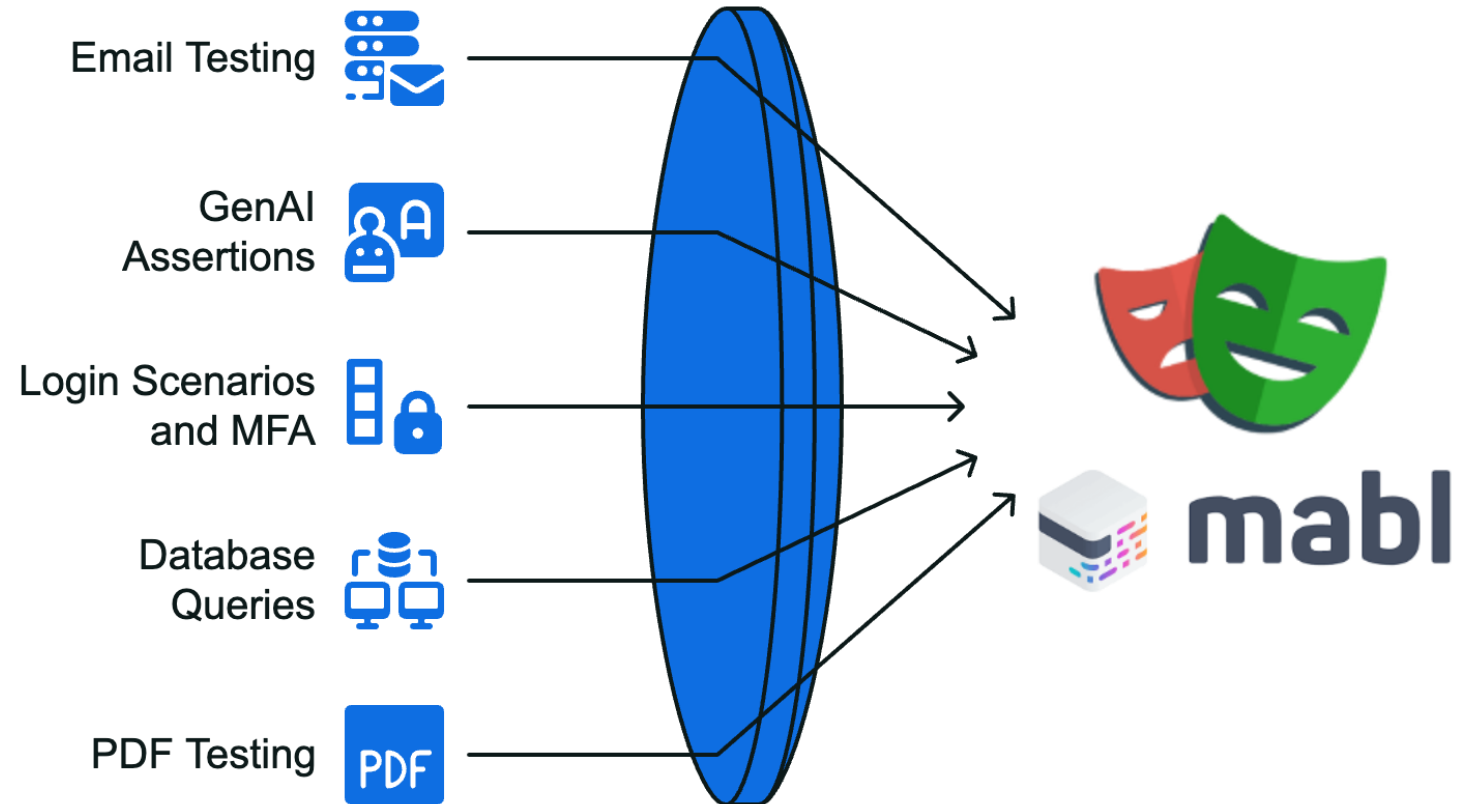
Past chats >

Challenge – Rewriting tests from C# to TS

Code Migration and Refactoring Process

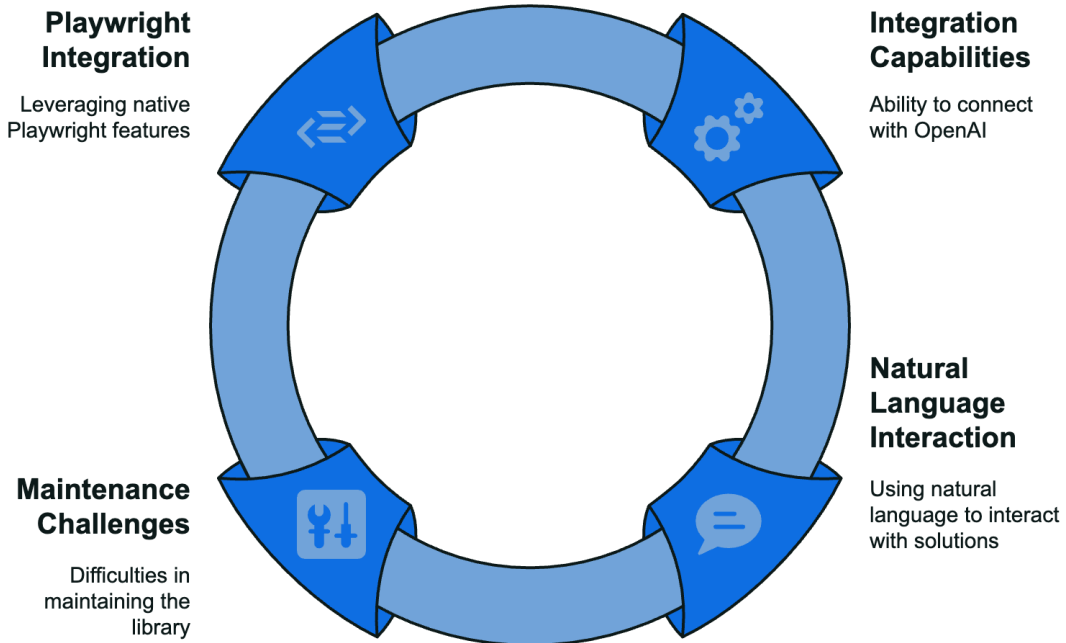


Playwright & Mabl integration



Auto Playwright

Run Playwright tests using AI.



```
import { test, expect } from "@playwright/test";
import { auto } from "auto-playwright";

test("auto Playwright example", async ({ page }) => {
  await page.goto("/");

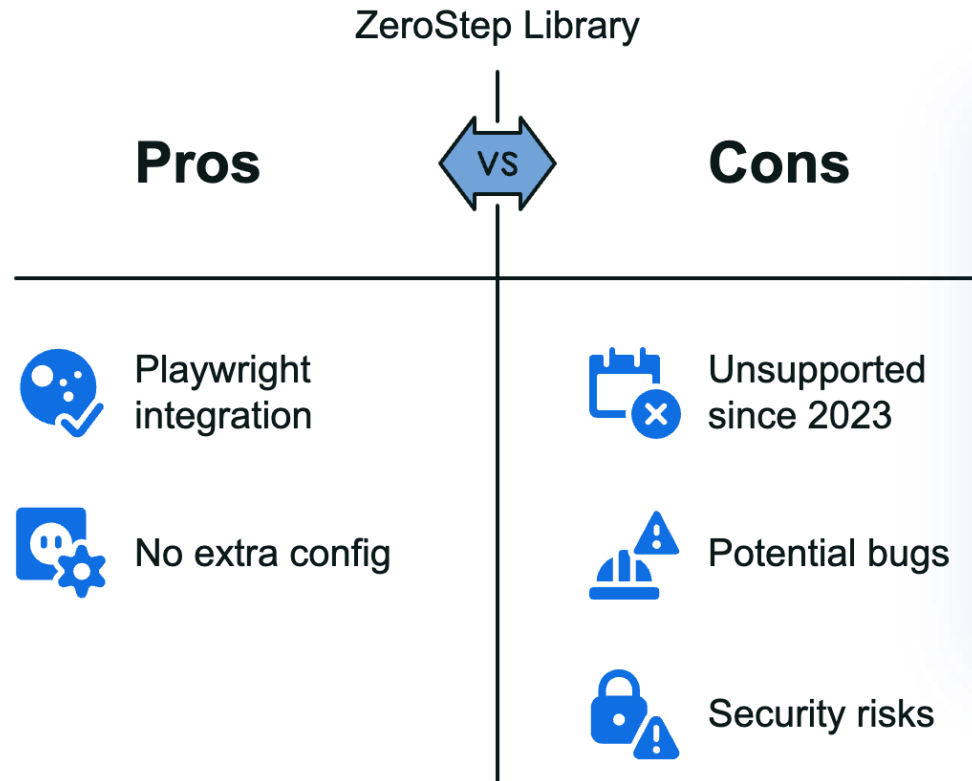
  // `auto` can query data
  // In this case, the result is plain-text contents of the header
  const headerText = await auto("get the header text", { page, test });

  // `auto` can perform actions
  // In this case, auto will find and fill in the search text input
  await auto(`Type "${headerText}" in the search box`, { page, test });

  // `auto` can assert the state of the website
  // In this case, the result is a boolean outcome
  const searchInputHasHeaderText = await auto(`Is the contents of the search box equal to "${h

  expect(searchInputHasHeaderText).toBe(true);
});
```


Playwright ZeroStep – deprecated (?)



Supercharge your Playwright tests with AI

ZeroStep's `ai()` function unlocks the power of GPT3.5 and GPT4 to make Playwright tests simpler and more resilient to change.

Get started

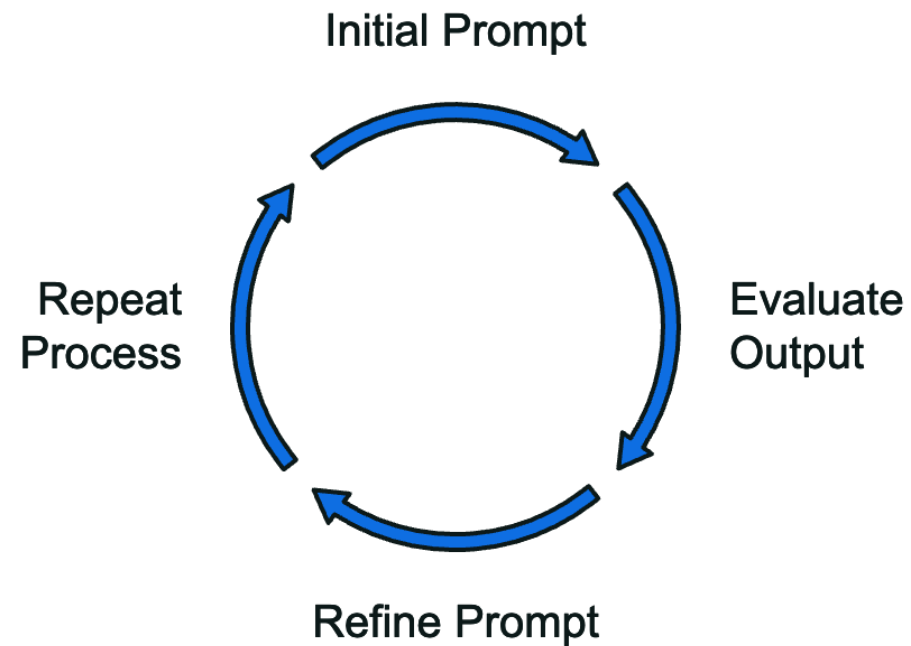
```
> npm i @zerostep/playwright -D
```

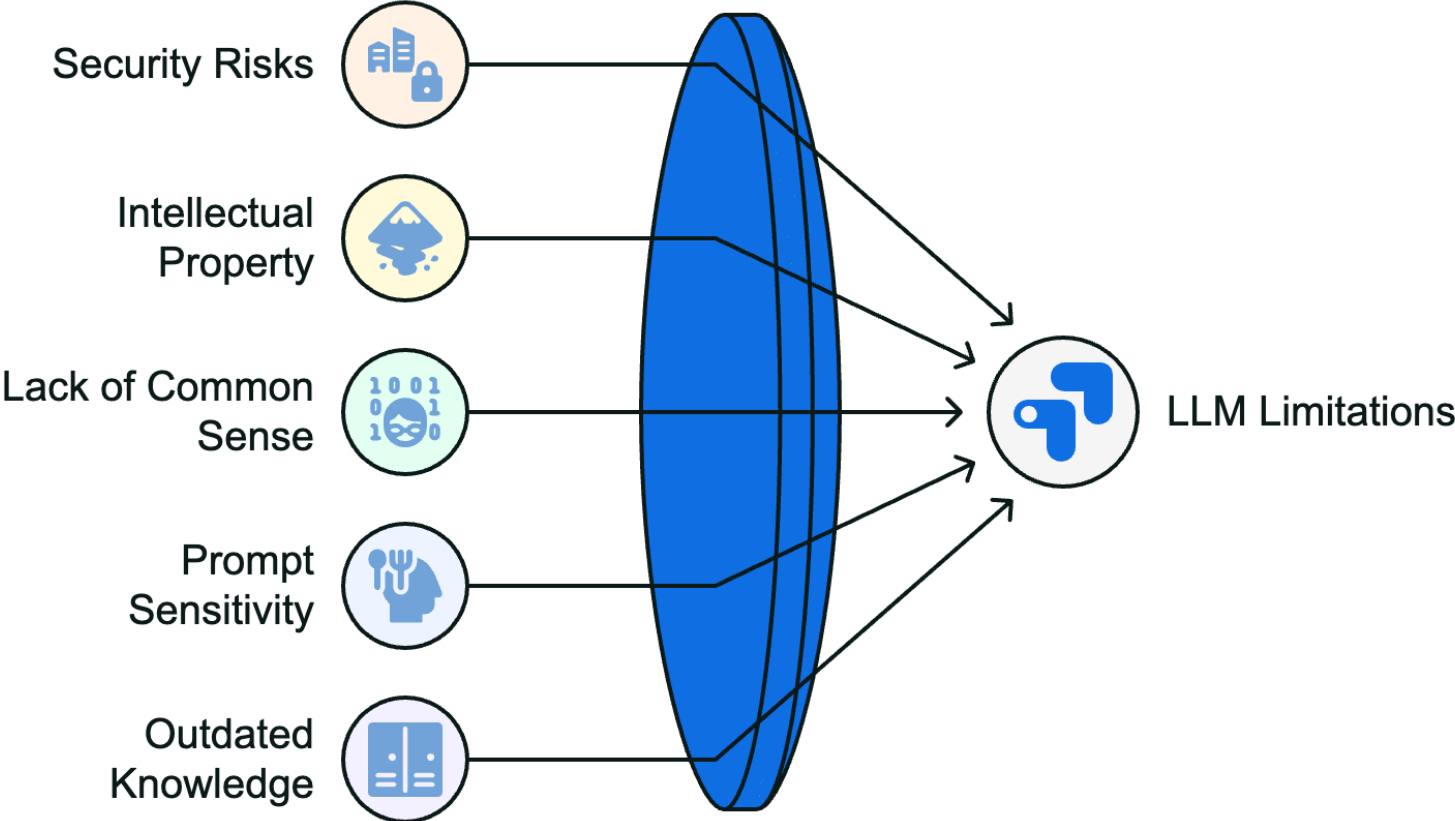



CHALLENGES

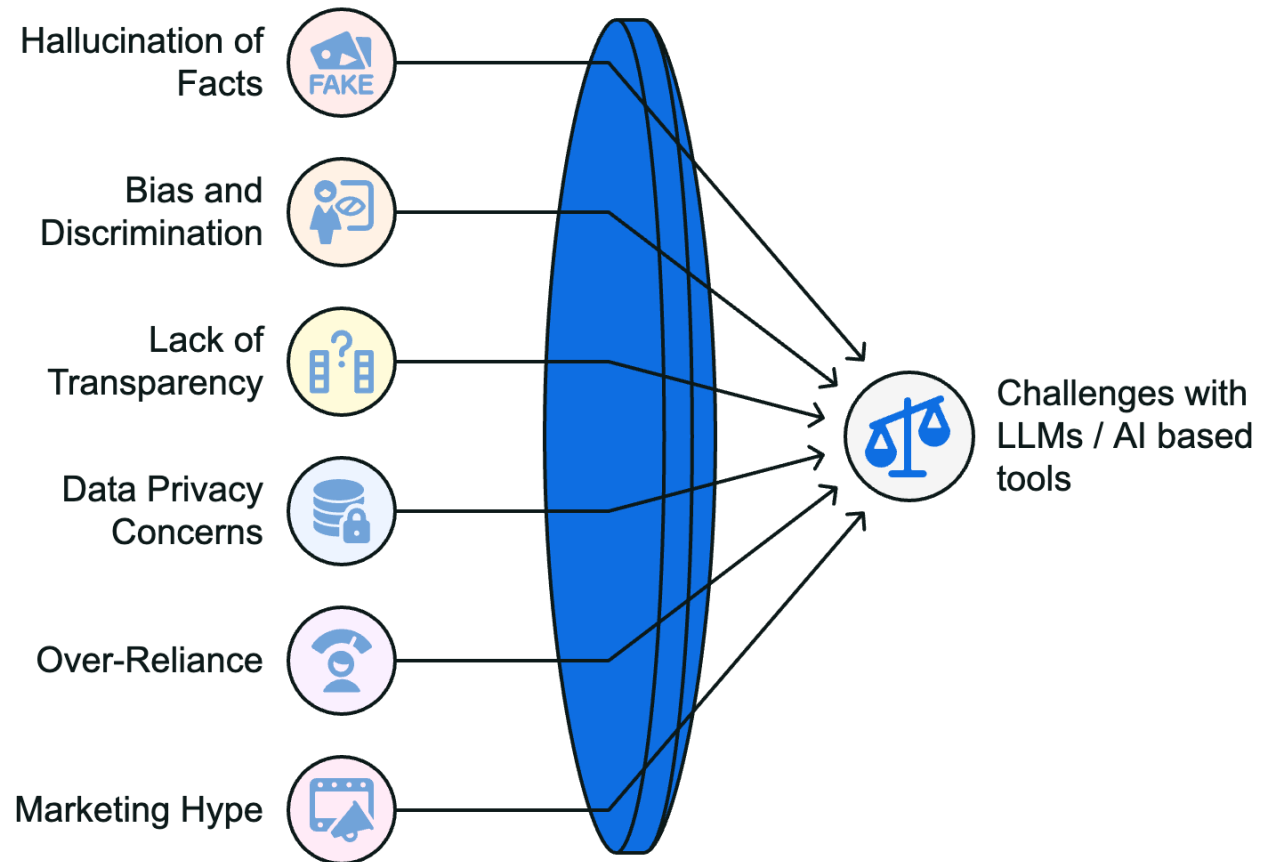
Marcin
Associate Delivery Manager

Iterative Prompting Cycle






Challenges in LLM Deployment



Articles about Playwright framework & visual regression





WHO WE AREWHAT WE OFFERINDUSTRIESCAREERTRAININGSNEWS & EVENTS

Contact us

Sii > BlogerSii > Playwright in practice: 5 steps to an effective UI & Web test automation framework. Part I


Testing7 February 2025

Playwright in practice: 5 steps to an effective UI & Web test automation framework. Part I


Michał Ślęzak

FrameworkTestingPlaywrightToolsGuidebookTypescript

Share









Next-Level Visual Regression Testing with Playwright: Track UI Changes Across Branches & Versions with Visual Regression Tracker (Open Source) (Docker Ready!)


READ MORE

Visual Regression

Visual Regression Tracker – it's a useful an open-source tool which extend a default functionalities in Playwright.

2025-03-02



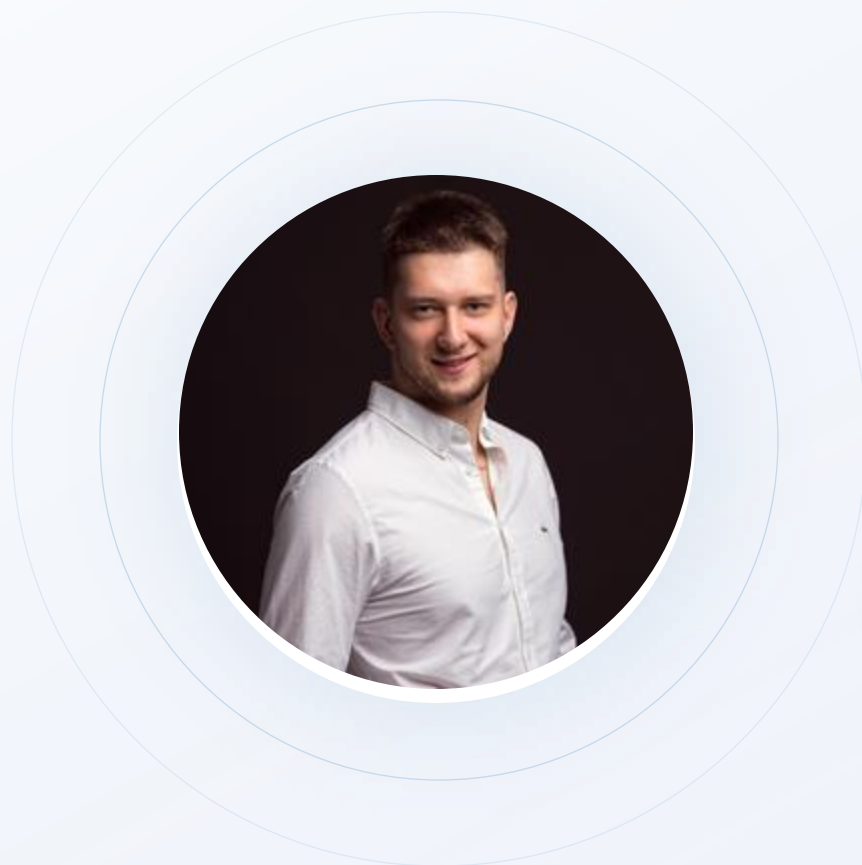


Visual Regression Testing: A Beginner's Guide with Playwright (Part 1)

Visual Regression Testing: Beginner's Guide with Playwright (Part 1)

2025-02-16

Contact me



Michał Ślęzak

Test Architect at Sii, consultant

Blogger: testingplus.me

mslezak@sii.pl

Q&A