**PLAYWRIGHT - The Most trending Automation tool which is an tough competitor to Selenium & Cypress**This is one and only Comprehensive Playwright course available online   which covers everything from basics to Framework  including JavaScript basics

**\*\*\*\*\*\*\*\*\*\*10 reasons why  Playwright can be future of UI Automation\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

Playwright enables reliable end-to-end testing for modern web apps with its Auto-wait capability

what is reliable(nammadagina – trusty ) here?

you might know that there are some pin points that tests are flaky, that they intermittently fail. And when you run again, they might pass. So the flakiness is because of so many things. So synchronization issues, waits are network issues that a lot of things can happen on the backend.

So how playwright help you to solve that flakiness and make sure it passes without any issues?

For that, it has brought some auto wait capability. So that means you really need not give any manual artificial sleeps for holds 5 seconds.

( <https://playwright.dev/docs/actionability> )

wait until the given time, whatever you provide in your framework that it will make sure it had all these checks pass before clicking on that.

So if not, it will automate and wait until that element comes into a stable, visible and enable state.

how does playwright Claiming that it can help you to perform end to end testing without much flakiness ? – using auto wait capablity

· Works on major Browser which uses Chromium Engine, (Chrome & Edge) Firefox, Safari (Web kit) & Opera

· Works on any OS – Windows, MAC, Linux & Supports Native mobile emulation(So that means you can actually sync your browser to Android phone dimensions and you can test your apps in mobile view.) of Google Chrome for Android and Mobile IOS in Safari

· Works with any language – JavaScript, TypeScript, Java, Python, C#(.NET). – (This course is made on JavaScript)

· Playwright have excellent inbuilt features called Traces which can take automatic Screenshots, test video recording, Flaky test retry & Logging mechanism

· Playwright provides inspector tool which help us to monitor and debug every step of execution, see click points and verify page locators on fly

· Playwright has inbuilt API Testing libraries to fire the Network calls on fly within Web Application. (Test Edge Case scenarios with the mix of web & API testing)

(playwright supports API testing Also there is one login API to log in which you can do from API also(inspect->application->local storage->token stored after login )( playwright have ability to go to your network tab and it can manipulate any storage items. So at runtime you can create cookies and you open browser directly on home page so that the login will be bypassed because you are injecting cookie from your API response ) ( Now everyone taking the help of both API and Web to make the test case end to end very optimized )

· Playwright provides browser context feature which help to save and transfer the browser state to any other new browser. ( login & after you login you can tell playwright that go and collect all the cookies, storage items, everything from this browser so playwright can actually collect this cookies, everything And now what happens is now you open fresh browser and you can tell whatever you collected from this browser, inject it into the fresh browser if it injected in that browser.So again, when you hit url, you were directly land on home page. So you will not go to login screen because all the cookies you got from this browser. & you grab that into one file and that file you will inject into the new browser. So when you hit this url, you will be directly logged in, but you won't see that login screen. So that means storage state, that's sessions data, one browser you can transfer to another browser. ( ex : 10 TC’s – 1st TC you login and remaining 9TC’s it starts from the home page bcze if any the new browser it starts from the home page bcze we injected the cookies, session storage data )

· Playwright provides codegen tool which Generate test code by recording your actions. Save them into any language. ( So that means you manually go and create an order, log in, place an order, and then automatically playwright can listen to your actions and it will generate code for you )

With all the above top features it has, Playwright is now tough competitor for Selenium & Cypress automation tools.

**Breakdown of Playwright topics in this course**

We are using JavaScript language binding to build Playwright Automation tests. (Js is mainly picked language for Playwright)  
For Beginners, there is 2 hours Javascript video in the last section of this course which covers all JS basics from Scratch.

Playwright Web/UI Automation (covers all core concepts) – 50%

Mix of Web & API Tests including Network mocking. – 25%

Playwright Advanced Framework features. – 25%

JavaScript Basics for Beginners (optional)

4. Install Node.js & Visual Studio for setting up Playwright Environment

what is Node.js ?

Node.js is an open-source,cross-platform, back-end javaScript runtime environment that runs on the V8 engine and executes javascript code outside a webbrowser.

Install Node.js

Install Visual studio code ( Best for Javascript/Type script )

Playwright official website – <https://playwright.dev/>

Install playwright dependencies - npm init playwright

In google ( node install -> open official website ( [Node.js — Download Node.js® (nodejs.org)](https://nodejs.org/en/download/) ) -> LTS(window installer(.msi) 64bit -select -> once it download -> double click ( This package will install these two(node.js, npm ) softwares into this paths ) -> continue->continue->agree -> just read it ( make sure this node.js path is in your path variable) (  as node.js is sitting in this path And if you don't have this path present in your path variable, then you cannot use node globally )

So once node.js is successfully installed, it will store in program files in your C directory.

(copy the path - C:\Program Files\nodejs )

go to the system Environmental Properties. ( system properties->environment varaiables->system variables->select new -> variable name : NODE\_HOME Variable value : paste the path ) -> okay

we need to have an editor to write our code.

VS Code is the best editor for all your JavaScript and TypeScript needs.

Google(visual studio code -> open the official website -> download ( stable )