1: Introduction

In this session, we are evaluating two popular tools for frontend automation testing: Playwright and Puppeteer.

Both are built on the Chrome DevTools Protocol and designed for modern web testing.

We will compare them across key developer concerns like language support, selector handling, login automation, and CI/CD integration.

The objective is to assess which tool best fits our app's architecture, and align on one standard framework.

2: Developer Experience

Playwright supports multiple languages: JavaScript, TypeScript, Python, .NET, and Java.

This multi-language support fits our tech stack, including backend in C++, Redis-based services, and UI in JS/TS.

Integrated development experience with VS Code and built-in test runner makes setup smooth.

Puppeteer, in contrast, supports only JavaScript/TypeScript, limiting flexibility and requiring external test runner setup.

3: Finding Selectors

Playwright supports powerful selectors: `getByText`, `getByRole`, `getByPlaceholder`, CSS, XPath, etc.

Built-in retry logic reduces flakiness when elements load dynamically.

Codegen tool helps generate selectors interactively and improves productivity.

Example usage:

js

await page.goto('http://localhost:3000/performance-analytics/returnsmetrics');

await page.getByPlaceholder('Select date').fill('30-04-2025');

Puppeteer only offers basic selectors and lacks retry capabilities, requiring more custom logic.

4: Testing Pattern

Playwright supports BDD test frameworks like CucumberJS for JS/TS and SpecFlow for .NET.

Built-in fixtures, test hooks (`beforeAll`, `afterEach`, etc.) and test isolation help maintain clean test structure.

Supports reusable setup and teardown patterns for modular test design.

Puppeteer has no built-in lifecycle management or support for structured test frameworks. Setup is manual.

5: Handling Login

Playwright supports persistent login via `storageState.json` — credentials are saved and reused across tests.

Works with enterprise auth flows including ADFS and MSAL.

In CI environments, non-interactive login can use NTID credentials for seamless test runs.

Puppeteer lacks native support for persistent auth. Developers must handle login logic manually per test.

6: Pluggability

Playwright works across different environments: Python with Pytest, .NET with SpecFlow, Java with JUnit.

Enables collaboration across full-stack teams with diverse language backgrounds.

Offers custom fixtures, reporters, and plugin support to extend capabilities.

Puppeteer is limited to JavaScript/TypeScript and lacks official multi-language support.

7: Test Runners and CI

Playwright includes a native test runner with:

Parallel execution

Retry logic

Test sharding and filtering

Integrates well with GitHub Actions, Azure DevOps, GitLab, Jenkins, and supports Docker environments.

Tests can be run across Chromium, Firefox, and WebKit.

Puppeteer lacks built-in parallelism and test runner — additional tools and configuration are required.

8: Reporting

Playwright includes rich reporting by default:

HTML reports with trace viewer

Screenshots and videos on failure

Custom reporter integration (Allure, TestRail)

Dot-style output reduces noise while keeping logs clear.

Puppeteer has no built-in reporting — developers need to integrate external tools for even basic reporting.

9: Conclusion and Recommendation

Playwright is a complete, modern testing solution with rich functionality and strong support for enterprise testing needs.

It reduces setup time, lowers test flakiness, and supports multiple teams with different stacks.

With its CI/CD integrations, reusable auth handling, and detailed reporting, it significantly improves test stability and efficiency.

Recommendation: Standardize on Playwright as our frontend automation tool to ensure long-term scalability and maintainability.