Playwright E2E Automation Challenge

Overview

This **Playwright** E2E test automates the complete user journey on the **DemoBlaze** e-commerce site, covering **login, product purchase, checkout,** and **logout**. Built with the Page Object Model, it keeps the test organized and easy to maintain. Flaky or slow UI elements are handled with explicit waits and alert handling, ensuring reliability. Allure reporting provides detailed step-by-step logs, screenshots, and videos for clear insights. This approach delivers a professional, stable, and interview-ready end-to-end automation solution.

Challenge Objective

Automate a complete user flow:

1. Open website
2. Login
3. Add products to cart
4. Complete checkout
5. Logout

Constraints:

- Single E2E test
- Page Object Model (POM)
- Handle flaky UI behavior
- Generate Allure reports

Tech Stack

- Playwright Test
- TypeScript
- Page Object Model (POM)
- Allure Reporting
- Node.js / npm

Project Structure

project-root/

├── pages/DemoBlazePage.ts

├── tests/buy-products.spec.ts

├── playwright.config.ts

├── allure-results/

└── allure-report/

Test Site

https://www.demoblaze.com (public demo e-commerce site)

Chosen to simulate **real-world instability** such as slow modals and JavaScript alerts.

Design Approach

- Page Object Model for clean separation
- Explicit waits instead of hard delays
- Alert handling before triggering actions
- Increased timeout for flaky UI

Test Scenario

Login → Add Products → Checkout → Logout

Implemented as a **single stable E2E test** with proper synchronization.

Allure Reporting

Allure is integrated to provide:

- Execution history
- Failure screenshots & videos
- Clear step-based reporting

Key Commands

npm install -D allure-playwright allure-commandline

npx playwright test

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

npx allure open allure-report

Best Practices Demonstrated

- Clean framework structure
- Stable synchronization strategy
- Professional reporting
- Interview-ready explanation

Interview Summary

> "I built a Playwright E2E framework using Page Object Model, handled flaky UI behavior with explicit waits and alert handling, and integrated Allure for professional test reporting."

Links & References

- **GitHub Repository:** https://github.com/saadali321/saad-ali-coding-challange.git
- **Test Video:** https://www.loom.com/share/3e5c470c13424056b1f2bb9caa4e11e6

Conclusion

This challenge showcases a **concise, production-style Playwright automation framework** focused on stability, maintainability, and professional reporting. It is suitable for coding challenges, interviews, and real-world QA automation projects.