

Automation Testing / SDET

Design good tests, then automate them across the UI, the API and the database

Software Testing · intermediate · Live online, instructor-led

This syllabus is a draft awaiting instructor review and may change.

A twelve-week live course in test design and automation with Playwright and TypeScript: locators, page objects, fixtures, API and SQL checks, CI with GitHub Actions, flaky-test fixes, visual, accessibility and k6 performance basics.

Length and schedule

72 hours of live teaching, on either schedule:

- Weekday Intensive (Mon–Fri): about 15 weeks before holidays
- Weekend Evening (Sat & Sun evenings): about 18 weeks before holidays

Fees per student, whole course

- Group: Rs. 11,999
- Duo: Rs. 19,999
- One-to-One: Rs. 35,999

Launch fee — GST treatment to be confirmed.

What you will be able to do

- Derive test cases systematically with equivalence partitioning, boundary values, decision tables and state transitions
- Write clear, reproducible bug reports with evidence
- Automate reliable browser tests with Playwright locators, web-first assertions and auto-waiting
- Structure a framework with page objects, custom fixtures and managed test data
- Test REST APIs and verify stored data with SQL in the same suite
- Run tests in parallel in GitHub Actions and diagnose failures with reports and traces
- Investigate and fix flaky tests, and add visual, accessibility and basic load checks

Before you start

- Basic programming in any language: variables, conditions, loops and functions
- Familiarity with using web applications and browser developer tools is helpful
- Basic SQL (SELECT with WHERE and joins) is helpful; a short refresher is included
- A laptop that can run Node.js, VS Code and modern browsers (Windows, macOS or Linux; 8 GB RAM recommended)
- A free GitHub account for the CI module

Tools

Playwright Test, TypeScript, Node.js and npm, VS Code with the Playwright extension, Playwright trace viewer and HTML reporter, PostgreSQL and the node-postgres (pg) client, axe-core (@axe-core/playwright), k6, Git and GitHub, GitHub Actions

Syllabus

Module 1: Testing fundamentals and test design

Why we test, how to decide what to test, and how to report what you find.

1. Testing in the software lifecycle (2 h)
2. Equivalence partitioning and boundary values (2 h)
3. Decision tables, state transitions and risk-based testing (2 h)
4. Bug reporting workshop and doubt-clearing (2 h)

Module 2: TypeScript essentials for testers

Enough TypeScript and Node.js to write clean, typed test code with confidence.

5. Node.js, npm and TypeScript setup (2 h)
6. Types, functions and objects (2 h)
7. Asynchronous code with async/await (2 h)
8. Classes, modules and doubt-clearing (2 h)

Module 3: Playwright fundamentals

Write your first reliable browser tests with locators, assertions and auto-waiting.

9. Your first Playwright test (2 h)
10. Locators that survive change (2 h)
11. Assertions and auto-waiting (2 h)
12. Automation lab and doubt-clearing (2 h)

Module 4: Page objects and fixtures

Keep a growing suite maintainable with page objects, fixtures and shared sign-in.

13. The page object model (2 h)
14. Built-in and custom fixtures (2 h)
15. Authentication and storage state (2 h)
16. Refactoring lab and review (2 h)

Module 5: API testing and SQL data validation

Test below the UI: REST APIs with Playwright's request context and stored data with SQL.

17. API testing with the request context (2 h)
18. Negative, authorisation and contract checks (2 h)
19. Validating stored data with SQL (2 h)
20. UI, API and database lab and doubt-clearing (2 h)

Module 6: Framework architecture and test data

Grow the suite into a framework a team can share, with reliable test data.

21. Framework architecture (2 h)
22. Reusable fixtures and helpers (2 h)
23. Test data management (2 h)
24. Reading Selenium WebDriver code and doubt-clearing (2 h)

Module 7: Parallel execution, reporting and CI

Run the suite quickly and automatically, and make every failure easy to diagnose.

25. Parallel execution and sharding (2 h)

- 26. Reports and the trace viewer (2 h)
- 27. Continuous integration with GitHub Actions (2 h)
- 28. CI lab and doubt-clearing (2 h)

Module 8: Reliability, visual, accessibility and performance checks

Fix flaky tests at their root, and extend coverage beyond functional behaviour.

- 29. Investigating flaky tests (2 h)
- 30. Visual and accessibility checks (2 h)
- 31. Introduction to performance testing with k6 (2 h)
- 32. Module review and doubt-clearing (2 h)

Module 9: Capstone: test automation framework

Plan, build and present a complete framework for a realistic application.

- 33. Capstone brief and test strategy (2 h)
- 34. Capstone build clinic (2 h)
- 35. Capstone peer review (2 h)
- 36. Final presentations and final assessment (2 h)

Final project

Build a test automation framework for a course-booking application. You receive the application, its API documentation and read access to its PostgreSQL database. You deliver:- A short test strategy: the risks you identified, what you automate at the UI, API and database levels and why, and what you deliberately leave manual.- Test cases derived with at least three design techniques, traced to the requirements they cover.- A Playwright and TypeScript framework in a Git repository with page objects, custom fixtures, environment configuration and test data that is created and cleaned up by the tests.- UI journeys for sign-up, search, booking and cancellation; API tests including negative and authorisation cases; SQL checks that the stored data matches what the UI and API reported.- An accessibility scan and at least one visual comparison on key pages, plus a small k6 script with thresholds for the booking API.- A GitHub Actions workflow that runs the suite in parallel on every push and publishes the HTML report and traces.- Two bug reports for real defects planted in the application, and a short note on one flaky test you diagnosed and fixed. Reviewed live in the final week: you walk through the framework, run it in CI and answer questions about your design choices.

Certificate

- At least 80% attendance
- All assignments submitted
- Projects passed
- Final assessment passed (60%)

A ZORK Academy course completion certificate; paying for the course does not by itself earn it.

Next step

Book a free consultation: <https://zorkacademy.online/consultation?course=automation-testing>

We help you prepare for work; we do not guarantee jobs, placements or salaries.