Automation Testing with Playwright and TypeScript

Duration: 4 hours

Prerequisites: Knowledge of any oops programming language.

Day 1: Introduction to TypeScript and Playwright

Topics Covered:

✅ **Basic TypeScript Concepts (for Test Automation)**

- Introduction to TypeScript: what, why, how
- Type annotations: string, number, boolean, any, unknown
- Interfaces and types
- Functions and arrow syntax
- async/await basics
- Modules and imports

✅ **Getting Started with Playwright**

- What is Playwright? Key features and comparisons
- Installing Node.js and Playwright
- Setting up a Playwright project with TypeScript
- Understanding folder structure and config files
- Writing your first test using Playwright

Labs:

- Create a basic TypeScript file with functions and interfaces
- Initialize a Playwright + TypeScript project
- Write and run a simple test for a sample login page

Day 2: Interactions, Assertions & TypeScript in Tests

Topics Covered:

✅ **Playwright Selectors & Actions**

- Types of selectors: CSS, text, role, XPath
- UI interactions: click, fill, type, check, selectOption
- Keyboard and mouse simulation

✅ **Assertions & Auto-waiting**

- Using the expect API

- Understanding auto-waiting and retries
- Common assertion patterns

✅ **TypeScript in Test Files**

- Creating helper functions with types
- Using interfaces for test data
- Structuring test files with type-safe imports

Labs:

- Automate a form submission with validations
- Create custom TypeScript utility functions for reusability
- Add multiple assertion types to verify UI states

Day 3: Advanced Playwright + Real-world Interactions

Topics Covered:

✅ **Dealing with Complex UI Elements**

- Handling modals, iframes, file uploads
- Multiple tabs/windows
- Advanced waiting strategies

✅ **Authentication & API Handling**

- Login flow automation (UI + API-based)
- Mocking network responses
- Making API calls with Playwright's request context

✅ **Using TypeScript for Test Data & Utilities**

- Creating data models with interfaces
- Using enums and constants for clean test design
- Organizing reusable TS modules

Labs:

- Automate a login flow using UI and session storage
- Handle file upload + iframe in a single test
- Use TypeScript interfaces to define API response mocks

Day 4: Project Structure, Debugging, and Mini Project

Topics Covered:

✅ **Test Organization & Debugging**

- Organizing tests using folders and test groups
- Writing reusable helper methods with TS
- Debugging techniques: pw-debug, --debug, VS Code tools
- Screenshots, videos, and tracing

✅ **Real-World Test Scenarios**

- Writing end-to-end scenarios: login → dashboard → update → logout
- Best practices for maintainable and readable test code
- Simple HTML report generation

Labs:

- Build a mini framework with reusable selectors and utilities
- Implement a real-world test suite for a sample app
- Practice debugging failed tests and analyzing artifacts