Playwright

Day 1: Basic Web Development Knowledge

- **HTML/CSS**:
  - Introduction to HTML: Elements, attributes, and structure.
  - Introduction to CSS: Styling elements, selectors, properties.
  - Building a basic webpage.
- **JavaScript**:
  - Variables, data types, and operators.
  - Functions, loops, and conditionals.
  - Basic DOM manipulation.
- **DOM (Document Object Model)**:
  - Understanding the DOM tree structure.
  - Selecting and manipulating DOM elements.
  - Event handling.
- **Introduction to Playwright**:
  - Overview of Playwright and its features.
  - Comparison with other automation tools (Selenium, Puppeteer).

Day 2-3: Playwright Fundamentals

- **Installation and Setup**:
  - Installing Playwright on different OS (Windows, macOS, Linux).
  - Setting up a project (Node.js environment, npm packages).
  - Configuring Playwright (launch options, browser configurations).
- **Creating and Running Tests**:
  - Writing a basic test script.
  - Running tests using the Playwright test runner.
  - Understanding test results and logs.

Day 4-5: Core Playwright Concepts

- **Selectors**:
  - Using CSS selectors.
  - Using XPath.
  - Playwright-specific selectors (e.g., text=, role=).
- **Interacting with Elements**:
  - Clicking buttons and links.
  - Typing into input fields.
  - Selecting options from dropdowns.
  - Drag and drop operations.
- **Assertions**:
  - Using Playwright's assertion library.
  - Common assertions (e.g., toBeVisible, toHaveText).
  - Custom assertions.
- **Navigation**:

- Handling page navigation and redirects.
- Waiting for elements to appear.
- Handling page load and AJAX requests.

Day 6: Frames and Iframes & Handling Dialogs

- **Frames and Iframes**:
  - Selecting and interacting with frames.
  - Handling nested iframes.
- **Handling Dialogs**:
  - Managing alerts, prompts, and confirmations.
  - Automating dialog interactions.

Day 7-8: Advanced Playwright Features I

- **Network Interception**:
  - Mocking network requests and responses.
  - Monitoring network activity.
- **Emulating Devices**:
  - Simulating different devices (e.g., mobile, tablet).
  - Configuring screen sizes and input methods.
- **Visual Comparisons**:
  - Taking screenshots.
  - Comparing screenshots for visual regressions.

Day 9: Advanced Playwright Features II

- **Multi-browser Testing**:
  - Running tests across Chromium, Firefox, and WebKit.
  - Configuring browser-specific options.
- **Parallel Test Execution**:
  - Running tests in parallel.
  - Configuring parallelism in Playwright.

Day 10: Playwright Integration with Various Test Frameworks

- **Playwright Test (Native Test Runner)**:
  - Writing and running tests with Playwright Test.
- **Jest**:
  - Integrating Playwright with Jest.
  - Writing Jest test cases using Playwright.
- **Mocha**:
  - Setting up Playwright with Mocha.
  - Writing Mocha test cases.
- **Cucumber.js (BDD)**:
  - Setting up Playwright with Cucumber.js.
  - Writing BDD test cases using Playwright.
- **Allure Report Integration**:
  - Generating test reports with Allure.

- Customizing Allure reports.

Day 11: Handling Object Recognition Problems I

- **Robust Selectors, Chained Selectors, Playwright-Specific Selectors**:
  - Creating robust and reliable selectors.
  - Using chained selectors for complex elements.
- **Locators API**:
  - Using the Locators API for efficient element selection.
- **Handling Dynamic Elements**:
  - Strategies for dealing with dynamic content.
- **Text and Attribute Assertions**:
  - Validating text content and attributes.
- **Retry Mechanism for Flaky Elements**:
  - Implementing retry logic for unstable elements.

Day 12: Handling Object Recognition Problems II

- **Handling Frames and Iframes**:
  - Techniques for working with frames and iframes.
- **Shadow DOM Handling**:
  - Accessing and interacting with Shadow DOM elements.
- **Using expect for Enhanced Assertions**:
  - Leveraging the expect library for more powerful assertions.

Day 13: Error & Exception Handling

- **Try-Catch Blocks**:
  - Implementing try-catch for error handling.
- **Playwright Test Hooks**:
  - Using hooks for setup and teardown.
- **Event Handling**:
  - Managing events during test execution.
- **Custom Error Handling Functions**:
  - Creating custom error handling logic.
- **Retry Logic**:
  - Implementing retries for failed tests.
- **Assertions and Expect Assertions**:
  - Advanced usage of assertions and expect statements.

Day 14: Test Organization and Management

- **Test Suites and Test Cases**:
  - Structuring tests into suites and cases.
- **Data-driven Testing**:
  - Parameterizing tests with multiple data sets.
- **Test Fixtures**:
  - Setting up and tearing down test preconditions.
- **Custom Test Reports**:

- Generating and customizing test reports.

Day 15: Integration with CI/CD, Debugging, and Best Practices

- **Continuous Integration**:
  - Integrating Playwright tests with CI tools (Jenkins, GitHub Actions, GitLab CI).
- **Continuous Deployment**:
  - Running tests in the deployment pipeline.
  - Ensuring test quality in CI/CD workflows.
- **Debugging and Troubleshooting**:
  - Using Playwright's debugging tools (e.g., debug).
  - Analyzing test executions step-by-step.
- **Best Practices**:
  - Writing clean, maintainable, and reusable code.
  - Ensuring tests are reliable and not flaky.
  - Optimizing test performance for faster execution.
- **Hands-on Practice & Community Resources**:
  - Web-based application automation using Playwright.
  - Leveraging official Playwright documentation and API references.
  - Engaging with the Playwright community on platforms like GitHub, Stack Overflow, and Slack.