

Beginner-Level Challenges

1. Open a webpage and verify title.
 2. Find an element by ID, name, class, XPath.
 3. Click a button and validate the result.
 4. Enter text in a form field and submit.
 5. Select an option from a dropdown.
 6. Check if a checkbox is selected or not.
 7. Handle radio buttons.
 8. Get text of a web element.
 9. Capture a screenshot of the page.
 10. Verify URL after redirection.
-

Intermediate Challenges

11. Handle JavaScript alerts, confirms, and prompts.
12. Switch between browser tabs or windows.
13. Upload a file using Selenium.
14. Download a file and verify it exists.
15. Handle iframes and nested frames.
16. Scroll to a specific element on a page.
17. Validate if an image is loaded properly.

18. Wait for an element to be visible (Explicit Wait).
 19. Use FluentWait for dynamic content.
 20. Interact with dynamic web tables (read/write).
 21. Check broken links on a page.
 22. Mouse hover over a menu and click sub-item.
 23. Right-click (context-click) on an element.
 24. Drag and drop elements.
 25. Verify tooltips using `getAttribute("title")`.
-



Advanced Challenges

26. Create a custom wait condition.
 27. Capture console logs from the browser.
 28. Run tests in headless mode.
 29. Automate login with session persistence (cookies).
 30. Handle CAPTCHA or 2FA (manual or bypass demo).
 31. Parallel test execution using Selenium Grid or TestNG.
 32. Integrate Selenium with CI/CD pipeline (e.g., Jenkins).
 33. Automate interaction with a chatbot or dynamic content.
 34. Use Selenium with cloud platforms (e.g., BrowserStack, SauceLabs).
 35. Create a page object model (POM) based framework.
-



Bonus Challenges for Framework Integration

36. Run Selenium tests with JUnit/TestNG (Java) or Pytest (Python).
37. Use data-driven testing (from Excel, JSON, or CSV).
38. Implement logging and reporting (e.g., Allure, ExtentReports).
39. Retry failed tests automatically.
40. Generate and email HTML reports after execution.