Automation Testing

---

1. What is Automation Testing?

Definition:
Automation Testing is a software testing technique where test cases are executed automatically using specialized tools and scripts instead of manual human intervention.

Purpose:

- Increase test coverage
- Improve accuracy and repeatability
- Reduce human error
- Save time and cost in regression testing

When to use Automation Testing:

- Regression testing
- Repetitive test cases
- Performance/load testing
- Cross-browser testing
- Data-driven scenarios

Example scenario:

Imagine you have an e-commerce website. Every time you update the payment module, you need to:

1. Log in
2. Add product to cart
3. Proceed to checkout
4. Make payment

   Doing this **manually every release** is time-consuming. Automation allows running these steps automatically.

---

2. Different Tools for Automation Testing

Category	Tool Name	Purpose
Web Automation	Selenium, Cypress, Playwright	Test browser-based applications
Mobile Automation	Appium, Espresso, XCUITest	Test mobile apps (Android, iOS)
API Testing	Postman (Newman), RestAssured	Test REST/SOAP APIs
Performance	JMeter, LoadRunner	Load, stress, performance testing
BDD Tools	Cucumber, SpecFlow	Behavior Driven Development testing
Unit Testing	JUnit, TestNG, NUnit, PyTest	Test individual code units

---

3. Types of Applications and Tools Categorization

Application Type	Automation Tools
Web Applications	Selenium, Cypress, Playwright
Mobile Applications	Appium, Espresso, XCUITest
API Services	Postman, RestAssured, Karate
Desktop Applications	WinAppDriver, Winium, AutoIt
Performance Testing	JMeter, Gatling, LoadRunner
Security Testing	OWASP ZAP, Burp Suite

---

4. What is Selenium? / Why Selenium?

Definition:
Selenium is an open-source framework for automating web browsers. It supports multiple browsers (Chrome, Firefox, Edge) and programming languages (Java, Python, C#, JavaScript).

Why Selenium?

- Free and open source
- Supports multiple browsers
- Works with multiple programming languages
- Large community support
- Integrates with CI/CD pipelines

---

5. Difference Between Selenium and Other Automation Tools

Feature	**Selenium**	**Cypress**	**Playwright**
License	Open-source	Open-source	Open-source
Supported Apps	Web browsers	Web browsers	Web browsers & APIs
Languages	Java, Python, C#, JS, etc.	JavaScript/TypeScript only	JavaScript/TypeScript, C#, Python, Java
Cross-browser	Yes	Yes	Yes
Speed	Moderate	Fast	Fast
Mobile Testing	Via Appium	No	No

---

6. Introduction to Automation Frameworks

Definition:
An automation framework is a set of guidelines, coding standards, and best practices to create and manage test scripts efficiently.

Benefits:

- Reusability of code
- Easy maintenance
- Better reporting
- Scalability

---

7. Setting up the Environment

Example (Java + Selenium + TestNG):

1. Install **Java JDK**
2. Install **IDE** (Eclipse/IntelliJ)
3. Add **Selenium WebDriver JARs**
4. Install **TestNG plugin** in IDE
5. Create a **Maven Project** and add dependencies in `pom.xml`

---

8. Unit Testing (JUnit, TestNG)

- **JUnit**: Popular Java unit testing framework
- **TestNG**: More advanced than JUnit, supports parallel execution, data-driven tests, and better reporting

Example – TestNG:

```
import org.testng.annotations.Test;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class LoginTest {
    @Test
    public void testLogin() {
        WebDriver driver = new ChromeDriver();
        driver.get("https://example.com/login");
        // login steps
        driver.quit();
    }
}
```

---

9. Selenium with TestNG – Data Driven Test Framework

Data-driven testing: Test data is stored in external files (Excel, CSV, DB) and passed into the test scripts.

Example:

```
@DataProvider(name="loginData")
public Object[][] getData() {
    return new Object[][] {
        {"user1", "pass1"},
        {"user2", "pass2"}
    };
}

@Test(dataProvider="loginData")
public void loginTest(String username, String password) {
    // Selenium code to perform login
}
```

---

10. Page Object Model (POM)

- Each page of the application is represented as a Java class
- Improves code readability and maintenance

Example:

```
public class LoginPage {
    WebDriver driver;
    By usernameField = By.id("username");
    By passwordField = By.id("password");
    By loginButton = By.id("login");

    public LoginPage(WebDriver driver) {
        this.driver = driver;
    }
    public void login(String user, String pass) {
        driver.findElement(usernameField).sendKeys(user);
        driver.findElement(passwordField).sendKeys(pass);
        driver.findElement(loginButton).click();
    }
}
```

---

11. Keyword Driven Framework

- Uses keywords (login, click, type) stored in Excel/CSV to drive execution
- Testers without coding skills can write test cases

Example:

Keyword	Locator	Value
open	url	https://site.com
type	id=username	user1
click	id=login	

---

12. Hybrid Framework

- Combination of **POM + Data Driven + Keyword Driven**
- Most real projects use Hybrid frameworks

Example Live Scenario:

- **POM** for page classes
- **Excel** for test data
- **Keywords** to define steps
- **TestNG** for execution control