30 Câu Hỏi Phỏng Vấn Selenium — Phiên Bản Thực Chiến (Có Code)

> Tài liệu này dành cho vòng live coding, take-home assignment, hoặc khi em muốn ôn kỹ thuật trước phỏng vấn. Mỗi câu có code Java thực chiến với Selenium WebDriver 4 và JUnit 5 — stack phổ biến nhất trong các dự án enterprise — kèm các bẫy thường gặp và best practice từ kinh nghiệm thực tế.

Phần 1 — Câu hỏi căn bản đóng gói trong tình huống (10 câu)

Câu 1. Setup driver cho team đang chuyển từ Selenium 3 lên 4. Code cũ dùng WebDriverManager — em refactor thế nào?

Vì sao interviewer hỏi: Test em có biết Selenium 4 đã có Selenium Manager built-in, và có biết refactor an toàn không.

Code cũ (Selenium 3 style):

```
import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;

public class OldDriverFactory {
    public static WebDriver createDriver() {
        WebDriverManager.chromedriver().setup();  // Không cần nữa
        ChromeOptions options = new ChromeOptions();
        options.addArguments("--start-maximized");
        return new ChromeDriver(options);
    }
}
```

Code mới (Selenium 4.6+):

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import java.time.Duration;

public class DriverFactory {
    public static WebDriver createChromeDriver() {
        ChromeOptions options = new ChromeOptions();
        options.addArguments("--start-maximized");
        options.addArguments("--disable-notifications");
        options.addArguments("--disable-popup-blocking");

        // Selenium Manager TỰ ĐỘNG tải đúng version driver
        // KHÔNG cần WebDriverManager nữa
        WebDriver driver = new ChromeDriver(options);

        driver.manage().timeouts()
            .pageLoadTimeout(Duration.ofSeconds(30))
            .scriptTimeout(Duration.ofSeconds(10));

        return driver;
    }
}
```

Refactor checklist khi migrate:

```
// 1. Xóa WebDriverManager dependency khỏi pom.xml
// <dependency>
//     <groupId>io.github.bonigarcia</groupId>
//     <artifactId>webdrivermanager</artifactId>
// </dependency>

// 2. Xóa các dòng WebDriverManager.*.setup() — Selenium Manager tự lo

// 3. DesiredCapabilities → đổi sang Options class
// CŨ:
DesiredCapabilities caps = new DesiredCapabilities();
caps.setCapability(ChromeOptions.CAPABILITY, chromeOpts);

// MỚI:
ChromeOptions options = new ChromeOptions();
// merge trực tiếp, không qua DesiredCapabilities

// 4. driver.manage().timeouts() — đổi sang Duration
// CŨ:
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

// MỚI:
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
```

Bẫy migration:

- Quên xóa WebDriverManager khỏi pom.xml — vẫn dùng song song gây conflict version
- DevOps đang cache `~/.cache/selenium` cũ — clear cache trước khi build CI

- Test cũ phụ thuộc behavior của JSON Wire Protocol (Selenium 3) — Selenium 4 dùng W3C, một số response khác format

---

Câu 2. Em đang test dropdown có 50 option, verify option "Hà Nội" tồn tại. Viết code — và chỉ ra anti-pattern em sẽ tránh.

Vì sao interviewer hỏi: Test em biết khi nào dùng `findElement` vs `findElements`, có dùng exception đúng cách không.

❌ Anti-pattern (KHÔNG bao giờ làm):

```
// SAI: dùng exception để check tồn tại — đắt và không rõ ý
public boolean isOptionExists(String optionText) {
    try {
        driver.findElement(By.xpath("//option[text()='" + optionText + "']"));
        return true;
    } catch (NoSuchElementException e) {
        return false;
    }
}
```

✅ Cách đúng — dùng `findElements`:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import java.util.List;

public boolean isOptionExists(String optionText) {
    // findElements TRẢ VỀ LIST RỖNG khi không thấy, KHÔNG throw exception
    List<WebElement> options = driver.findElements(
        By.xpath("//select[@id='city']/option[text()='" + optionText + "']")
    );
    return !options.isEmpty();
}
```

✅ Cách tốt hơn — dùng `Select` class cho native dropdown:

```
import org.openqa.selenium.support.ui.Select;

public boolean isOptionExistsViaSelect(String optionText) {
    Select dropdown = new Select(driver.findElement(By.id("city")));
    return dropdown.getOptions().stream()
        .anyMatch(opt -> opt.getText().equals(optionText));
}

public void selectOptionIfExists(String optionText) {
    Select dropdown = new Select(driver.findElement(By.id("city")));

    boolean exists = dropdown.getOptions().stream()
        .anyMatch(opt -> opt.getText().equals(optionText));

    if (!exists) {
        throw new IllegalArgumentException(
            "Option '" + optionText + "' không tồn tại trong dropdown. " +
            "Available options: " + dropdown.getOptions().stream()
                .map(WebElement::getText).toList()
        );
    }

    dropdown.selectByVisibleText(optionText);
}
```

Bẫy với custom dropdown (div lồng nhau):

```
// Dropdown lười tải — option chỉ render KHI mở
public boolean isOptionInCustomDropdown(WebElement triggerBtn, String optionText) {
    // BƯỚC 1: Click mở dropdown
    triggerBtn.click();

    // BƯỚC 2: Đợi option list render xong
    wait.until(ExpectedConditions.visibilityOfElementLocated(
        By.cssSelector(".dropdown-menu.show")
    ));

    // BƯỚC 3: Find tất cả option SAU KHI dropdown đã mở
    List<WebElement> options = driver.findElements(
        By.cssSelector(".dropdown-menu .dropdown-item")
    );

    return options.stream().anyMatch(opt -> opt.getText().equals(optionText));
}
```

Pattern chung — Helper utility:

```
public class ElementUtils {

    public static boolean isPresent(WebDriver driver, By locator) {
        return !driver.findElements(locator).isEmpty();
    }

    public static int count(WebDriver driver, By locator) {
        return driver.findElements(locator).size();
    }

    public static boolean isVisible(WebDriver driver, By locator) {
        return driver.findElements(locator).stream()
            .anyMatch(WebElement::isDisplayed);
    }
}

// Sử dụng:
if (ElementUtils.isPresent(driver, By.id("error-msg"))) {
    // handle error
}
```

Câu 3. Dev push code mới làm vỡ 30 locator. Em viết script tự động phát hiện locator nào còn dùng được, locator nào chết — code thế nào?

Vì sao interviewer hỏi: Test em có biết viết tooling cho team, không chỉ viết test.

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import java.util.*;
import java.util.stream.Collectors;

public class LocatorHealthChecker {

    public record LocatorHealth(String name, By locator, boolean found, int count, String error

    private final WebDriver driver;
    private final Map<String, By> locatorRegistry;

    public LocatorHealthChecker(WebDriver driver) {
        this.driver = driver;
        this.locatorRegistry = new LinkedHashMap<>();
    }

    public LocatorHealthChecker register(String name, By locator) {
        locatorRegistry.put(name, locator);
        return this;
    }

    public List<LocatorHealth> checkAll() {
        return locatorRegistry.entrySet().stream()
            .map(e -> checkOne(e.getKey(), e.getValue()))
            .collect(Collectors.toList());
    }

    private LocatorHealth checkOne(String name, By locator) {
        try {
            List<WebElement> elements = driver.findElements(locator);
            return new LocatorHealth(name, locator, !elements.isEmpty(), elements.size(), null)
        } catch (Exception e) {
            return new LocatorHealth(name, locator, false, 0, e.getMessage());
        }
    }

    public void printReport() {
        List<LocatorHealth> results = checkAll();

        long total = results.size();
        long broken = results.stream().filter(r -> !r.found()).count();

        System.out.println("=".repeat(70));
        System.out.printf("LOCATOR HEALTH REPORT — %d/%d broken (%.1f%%)%n",
            broken, total, broken * 100.0 / total);
        System.out.println("=".repeat(70));

        for (LocatorHealth r : results) {
            String status = r.found() ? "✅" : "❌";
            String detail = r.found()
                ? String.format("found %d element(s)", r.count())
                : "NOT FOUND";
            System.out.printf("%s  %-30s  %s%n", status, r.name(), detail);
            if (r.error() != null) {
                System.out.printf("    ↳ Error: %s%n", r.error());
```

```
            }
        }
    }
}
```

Sử dụng để health-check toàn bộ LoginPage:

```
@Test
void healthCheckLoginPageLocators() {
    driver.get("https://app.example.com/login");

    new LocatorHealthChecker(driver)
        .register("email_input", By.cssSelector("[data-testid='email']"))
        .register("password_input", By.cssSelector("[data-testid='password']"))
        .register("submit_button", By.cssSelector("[data-testid='submit']"))
        .register("forgot_password_link", By.linkText("Quên mật khẩu"))
        .register("signup_link", By.cssSelector("a[href='/signup']"))
        .register("error_message", By.cssSelector(".error-banner"))
        .printReport();
}

// Output:
// ======================================================================
// LOCATOR HEALTH REPORT — 2/6 broken (33.3%)
// ======================================================================
// ✅  email_input                  found 1 element(s)
// ✅  password_input               found 1 element(s)
// ✅  submit_button                found 1 element(s)
// ❌  forgot_password_link         NOT FOUND
// ✅  signup_link                  found 1 element(s)
// ❌  error_message                NOT FOUND
```

Pro tip — Gắn vào CI:

```
# .github/workflows/locator-health-check.yml
name: Daily Locator Health Check
on:
  schedule:
    - cron: '0 6 * * *'  # 6h sáng mỗi ngày
jobs:
  check:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Run health check
        run: mvn test -Dtest=LocatorHealthSuite
      - name: Notify if broken > 10%
        if: failure()
        run: |
          curl -X POST -H 'Content-type: application/json' \
            --data '{"text":"⚠️ Locator broken rate > 10% — dev đã đổi UI"}' \
            ${{ secrets.SLACK_WEBHOOK }}
```

→ Phát hiện locator vỡ trước khi suite chạy đêm fail hàng loạt. Báo cho dev kịp thời.

Câu 4. Em phát hiện code team trộn implicit wait và explicit wait. Em viết utility class wait chuẩn để cả team dùng — code thế nào?

Vì sao interviewer hỏi: Câu này test em có biết design API cho team dùng, không chỉ viết test.

```java
import org.openqa.selenium.*;
import org.openqa.selenium.support.ui.ExpectedCondition;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import java.time.Duration;
import java.util.List;
import java.util.function.Function;

/**
 * Wait utility chuẩn cho cả team.
 * QUY TẮC: KHÔNG dùng implicit wait ở bất kỳ đâu trong project.
 * Mọi wait phải đi qua class này.
 */
public class WaitUtils {

    private static final Duration DEFAULT_TIMEOUT = Duration.ofSeconds(10);
    private static final Duration POLLING_INTERVAL = Duration.ofMillis(500);

    private final WebDriver driver;
    private final WebDriverWait wait;

    public WaitUtils(WebDriver driver) {
        this(driver, DEFAULT_TIMEOUT);
    }

    public WaitUtils(WebDriver driver, Duration timeout) {
        this.driver = driver;
        this.wait = new WebDriverWait(driver, timeout);
        this.wait.pollingEvery(POLLING_INTERVAL);
        this.wait.ignoring(StaleElementReferenceException.class);
    }

    // ===== ELEMENT WAITS =====

    public WebElement forClickable(By locator) {
        return wait.until(ExpectedConditions.elementToBeClickable(locator));
    }

    public WebElement forVisible(By locator) {
        return wait.until(ExpectedConditions.visibilityOfElementLocated(locator));
    }

    public boolean forInvisible(By locator) {
        return wait.until(ExpectedConditions.invisibilityOfElementLocated(locator));
    }

    public List<WebElement> forAllVisible(By locator) {
        return wait.until(ExpectedConditions.visibilityOfAllElementsLocatedBy(locator));
    }

    public WebElement forPresent(By locator) {
        return wait.until(ExpectedConditions.presenceOfElementLocated(locator));
    }

    // ===== TEXT/ATTRIBUTE WAITS =====
```

```
    public boolean forTextToBe(By locator, String expectedText) {
        return wait.until(ExpectedConditions.textToBe(locator, expectedText));
    }

    public boolean forAttributeToBe(By locator, String attr, String value) {
        return wait.until(ExpectedConditions.attributeToBe(locator, attr, value));
    }

    // ===== URL/TITLE WAITS =====

    public boolean forUrlContains(String fragment) {
        return wait.until(ExpectedConditions.urlContains(fragment));
    }

    public boolean forTitleContains(String fragment) {
        return wait.until(ExpectedConditions.titleContains(fragment));
    }

    // ===== CUSTOM CONDITION =====

    public <T> T forCustom(Function<WebDriver, T> condition) {
        return wait.until(condition::apply);
    }

    // ===== AJAX/JS COMPLETION =====

    public void forPageReady() {
        wait.until(d -> ((JavascriptExecutor) d)
            .executeScript("return document.readyState").equals("complete"));
    }

    public void forJQueryIdle() {
        wait.until(d -> {
            try {
                Long active = (Long) ((JavascriptExecutor) d)
                    .executeScript("return jQuery.active");
                return active != null && active == 0;
            } catch (Exception e) {
                return true; // không có jQuery — coi như idle
            }
        });
    }
}
```

BaseTest cấu hình driver KHÔNG implicit wait:

```
public abstract class BaseTest {
    protected WebDriver driver;
    protected WaitUtils wait;

    @BeforeEach
    void setUp() {
        driver = DriverFactory.createChromeDriver();

        // QUAN TRỌNG: implicit wait phải = 0 (không dùng)
        driver.manage().timeouts().implicitlyWait(Duration.ZERO);

        wait = new WaitUtils(driver);
    }

    @AfterEach
    void tearDown() {
        if (driver != null) driver.quit();
    }
}
```

Page Object dùng WaitUtils:

```
public class LoginPage extends BasePage {
    private final By emailInput = By.id("email");
    private final By submitBtn = By.cssSelector("[data-testid='submit']");
    private final By dashboardHeader = By.id("dashboard-header");

    public LoginPage(WebDriver driver) {
        super(driver);
    }

    public DashboardPage loginAs(String email, String password) {
        wait.forVisible(emailInput).sendKeys(email);
        wait.forVisible(By.id("password")).sendKeys(password);
        wait.forClickable(submitBtn).click();

        // Chờ navigation hoàn tất
        wait.forUrlContains("/dashboard");
        wait.forVisible(dashboardHeader);

        return new DashboardPage(driver);
    }
}
```

Cách giải thích cho team:

```
// ❌ TRƯỚC KHI fix — trộn implicit + explicit
public class BadLoginPage {
    public BadLoginPage(WebDriver driver) {
        // Implicit wait set ở đâu đó trong BaseTest
        driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

        // Explicit wait LẠI ở đây — timeout cộng dồn KHÔNG kiểm soát được
        new WebDriverWait(driver, Duration.ofSeconds(10))
            .until(ExpectedConditions.elementToBeClickable(By.id("submit")));
    }
}

// ✅ SAU KHI fix
public class GoodLoginPage {
    public GoodLoginPage(WebDriver driver) {
        // Implicit wait = 0 ở BaseTest
        // Explicit wait qua WaitUtils — 100% kiểm soát được timing
        wait.forClickable(By.id("submit"));
    }
}

// Số liệu để thuyết phục team:
// Trước: suite flaky 15%, trung bình 1 test fail tốn 24s timeout
// Sau:   suite flaky 2%, trung bình 1 test fail tốn 10s timeout
// Tiết kiệm: ~14s × 100 test × 5 lần fail/ngày = 116 phút/ngày
```

Câu 5. Em được giao refactor test cũ đầy `Thread.sleep`. Viết quy trình refactor an toàn — bao gồm code "before/after" cho một sleep cụ thể.

Vì sao interviewer hỏi: Test em có biết refactor incremental, không phá test đang pass.

Quy trình 4 bước:

```
// ========================================================================
// BƯỚC 1: Đọc context của sleep — comment intent vào code
// ========================================================================
public class LegacyCheckoutTest {

    public void completeCheckout() {
        driver.findElement(By.id("add-to-cart")).click();
        Thread.sleep(2000);  // ???
        driver.findElement(By.id("checkout-btn")).click();
        Thread.sleep(3000);  // ???
        driver.findElement(By.id("confirm")).click();
        Thread.sleep(5000);  // ???
        Assert.assertTrue(driver.findElement(By.id("success")).isDisplayed());
    }
}

// Sau khi đọc và comment intent:
public class LegacyCheckoutTestAnnotated {

    public void completeCheckout() {
        driver.findElement(By.id("add-to-cart")).click();
        Thread.sleep(2000);  // INTENT: chờ cart counter cập nhật

        driver.findElement(By.id("checkout-btn")).click();
        Thread.sleep(3000);  // INTENT: chờ navigate sang trang checkout

        driver.findElement(By.id("confirm")).click();
        Thread.sleep(5000);  // INTENT: chờ payment API response

        Assert.assertTrue(driver.findElement(By.id("success")).isDisplayed());
    }
}
```

Code refactor cụ thể cho một sleep:

```
// ========================================================================
// BƯỚC 2: Thay TỪNG sleep MỘT, không thay tất cả cùng lúc
// ========================================================================

// ❌ TRƯỚC: sleep cứng 2 giây
driver.findElement(By.id("add-to-cart")).click();
Thread.sleep(2000);  // INTENT: chờ cart counter cập nhật

// ✅ SAU: chờ điều kiện cụ thể (text của counter)
driver.findElement(By.id("add-to-cart")).click();
new WebDriverWait(driver, Duration.ofSeconds(5))
    .until(ExpectedConditions.textToBePresentInElementLocated(
        By.id("cart-counter"), "1"
    ));

// Hoặc nếu counter chỉ đơn giản là tăng:
String oldCount = driver.findElement(By.id("cart-counter")).getText();
driver.findElement(By.id("add-to-cart")).click();
new WebDriverWait(driver, Duration.ofSeconds(5))
    .until(d -> !d.findElement(By.id("cart-counter")).getText().equals(oldCount));
```

Test runner để verify refactor an toàn:

```
public class RefactorSafetyRunner {

    /**
     * Chạy test N lần liên tiếp, in tỉ lệ pass và thời gian trung bình.
     * Dùng trước và sau refactor để so sánh.
     */
    public static void runStabilityTest(Runnable testMethod, int iterations) {
        int passed = 0, failed = 0;
        long totalTime = 0;
        List<String> failures = new ArrayList<>();

        for (int i = 1; i <= iterations; i++) {
            long start = System.currentTimeMillis();
            try {
                testMethod.run();
                passed++;
                System.out.printf("Run %d: ✅ PASS (%dms)%n",
                    i, System.currentTimeMillis() - start);
            } catch (Throwable t) {
                failed++;
                failures.add("Run " + i + ": " + t.getMessage());
                System.out.printf("Run %d: ❌ FAIL — %s%n", i, t.getMessage());
            }
            totalTime += (System.currentTimeMillis() - start);
        }

        System.out.println("\n" + "=".repeat(60));
        System.out.printf("Stability: %d/%d passed (%.1f%%)%n",
            passed, iterations, passed * 100.0 / iterations);
        System.out.printf("Avg time:  %.1fs%n", totalTime / 1000.0 / iterations);
        System.out.println("=".repeat(60));
    }
}

// Sử dụng:
@Test
void measureCheckoutStability() {
    RefactorSafetyRunner.runStabilityTest(
        () -> new CheckoutFlow(driver).completeCheckout(),
        20  // chạy 20 lần
    );
}
```

Sleep được phép giữ — phải có comment giải thích:

```
// === SLEEP HỢP LỆ ===
// Lý do: CSS animation slide-in 300ms, không có hook để detect kết thúc
// Đã thử: animationend event không fire trong Firefox
// TODO: nếu Firefox fix bug, refactor thành waitForAnimation()
Thread.sleep(350);

// === SLEEP HỢP LỆ ===
// Lý do: reproduce race condition cố ý cho test concurrent
// Không được xóa — đây là điểm test critical
Thread.sleep(1000);
```

Câu 6. Test pass local nhưng fail trên CI với `StaleElementReferenceException`. Em viết helper retry và quy ước cho team — code thế nào?

Vì sao interviewer hỏi: Câu này check em có biết viết defensive code mà không che bug.

```
import org.openqa.selenium.*;
import java.time.Duration;
import java.util.function.Supplier;

public class StaleResistantActions {

    private static final int MAX_RETRY = 3;
    private static final Duration RETRY_INTERVAL = Duration.ofMillis(300);

    /**
     * Retry action khi gặp StaleElementReferenceException.
     * KHÔNG retry với các exception khác — để chúng bubble lên.
     */
    public static <T> T retryOnStale(Supplier<T> action) {
        StaleElementReferenceException lastException = null;

        for (int attempt = 1; attempt <= MAX_RETRY; attempt++) {
            try {
                return action.get();
            } catch (StaleElementReferenceException e) {
                lastException = e;
                System.out.printf("⚠️  Stale element, retry %d/%d%n", attempt, MAX_RETRY);
                sleep(RETRY_INTERVAL);
            }
        }

        throw new RuntimeException(
            "Stale element after " + MAX_RETRY + " retries", lastException);
    }

    public static void retryOnStale(Runnable action) {
        retryOnStale(() -> { action.run(); return null; });
    }

    private static void sleep(Duration duration) {
        try {
            Thread.sleep(duration.toMillis());
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
}
```

Pattern dùng đúng — không lưu reference lâu:

```
public class ProductListPage extends BasePage {

    private final By productCards = By.cssSelector(".product-card");
    private final By addToCartBtn = By.cssSelector(".add-to-cart");

    // ❌ SAI — lưu list element, sau khi click 1 cái, list bị re-render
    public void addAllToCartBad() {
        List<WebElement> cards = driver.findElements(productCards);
        for (WebElement card : cards) {
            card.findElement(addToCartBtn).click();  // StaleException!
        }
    }

    // ✅ ĐÚNG — index by position, re-find mỗi iteration
    public void addAllToCartGood() {
        int count = driver.findElements(productCards).size();

        for (int i = 0; i < count; i++) {
            // Re-find list mỗi vòng lặp
            WebElement card = driver.findElements(productCards).get(i);
            card.findElement(addToCartBtn).click();

            // Chờ DOM ổn định trước iteration tiếp
            wait.forJQueryIdle();
        }
    }

    // ✅ ĐÚNG — dùng helper retryOnStale cho trường hợp phức tạp
    public void addProductToCart(int productIndex) {
        StaleResistantActions.retryOnStale(() -> {
            List<WebElement> cards = driver.findElements(productCards);
            cards.get(productIndex).findElement(addToCartBtn).click();
        });
    }
}
```

Code review checklist — quy ước team:

```
/*
 * QUY ƯỚC TEAM về xử lý WebElement (đặt trong CONTRIBUTING.md):
 *
 * 1. KHÔNG lưu WebElement vào field của class
 *    ❌ private WebElement submitBtn;
 *    ✅ private final By submitBtn = By.id("submit");
 *
 * 2. KHÔNG dùng biến WebElement dài quá 10 dòng code
 *    Nếu cần dùng nhiều lần — find lại
 *
 * 3. KHÔNG tương tác với WebElement sau khi navigate/refresh
 *    Mọi reference cũ đều stale sau page change
 *
 * 4. DÙNG StaleResistantActions.retryOnStale() cho dynamic content
 *    Khi không tránh được phải lưu reference tạm thời
 *
 * 5. KHÔNG retry nhiều hơn 3 lần
 *    Nhiều hơn = che bug, không phải xử lý stale
 */
```

Câu 7. Test form đăng ký 15 trường — design code POM nâng cao với data builder. Viết đủ 3 lớp: page, builder, test.

Vì sao interviewer hỏi: Câu này test em có biết design system, không chỉ viết một class.

LỚP 1 — Page class:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;

public class RegistrationPage extends BasePage {

    // ===== Locators (final, đặt đầu class) =====
    private final By firstNameInput = By.cssSelector("[data-testid='first-name']");
    private final By lastNameInput = By.cssSelector("[data-testid='last-name']");
    private final By emailInput = By.cssSelector("[data-testid='email']");
    private final By passwordInput = By.cssSelector("[data-testid='password']");
    private final By confirmPasswordInput = By.cssSelector("[data-testid='confirm-password']");
    private final By phoneInput = By.cssSelector("[data-testid='phone']");
    private final By dobInput = By.cssSelector("[data-testid='dob']");
    private final By genderRadio = By.name("gender");
    private final By countrySelect = By.cssSelector("[data-testid='country']");
    private final By citySelect = By.cssSelector("[data-testid='city']");
    private final By addressInput = By.cssSelector("[data-testid='address']");
    private final By zipInput = By.cssSelector("[data-testid='zip']");
    private final By newsletterCheckbox = By.cssSelector("[data-testid='newsletter']");
    private final By termsCheckbox = By.cssSelector("[data-testid='terms']");
    private final By submitBtn = By.cssSelector("[data-testid='submit']");

    private final By errorAlert = By.cssSelector(".error-alert");

    public RegistrationPage(WebDriver driver) {
        super(driver);
    }

    public RegistrationPage open() {
        driver.get(config.getBaseUrl() + "/register");
        wait.forVisible(firstNameInput);
        return this;
    }

    // KHÔNG có 15 method setter riêng lẻ
    // CHỈ có một method fillForm nhận object data
    public RegistrationPage fillForm(RegistrationData data) {
        type(firstNameInput, data.firstName());
        type(lastNameInput, data.lastName());
        type(emailInput, data.email());
        type(passwordInput, data.password());
        type(confirmPasswordInput, data.password());  // dùng lại password
        type(phoneInput, data.phone());
        type(dobInput, data.dateOfBirth());
        selectRadio(genderRadio, data.gender());
        selectDropdown(countrySelect, data.country());
        selectDropdown(citySelect, data.city());
        type(addressInput, data.address());
        type(zipInput, data.zipCode());

        if (data.subscribeNewsletter()) check(newsletterCheckbox);
        if (data.acceptTerms()) check(termsCheckbox);

        return this;
    }
```

```
    public DashboardPage submitExpectingSuccess() {
        click(submitBtn);
        wait.forUrlContains("/welcome");
        return new DashboardPage(driver);
    }

    public RegistrationPage submitExpectingError() {
        click(submitBtn);
        wait.forVisible(errorAlert);
        return this;
    }

    public String getErrorMessage() {
        return wait.forVisible(errorAlert).getText();
    }
}
```

LỚP 2 — Data record + Builder:

```
// Record bất biến — Java 16+
public record RegistrationData(
    String firstName,
    String lastName,
    String email,
    String password,
    String phone,
    String dateOfBirth,
    String gender,
    String country,
    String city,
    String address,
    String zipCode,
    boolean subscribeNewsletter,
    boolean acceptTerms
) {}

public class RegistrationDataBuilder {
    // Defaults — đa số test dùng giá trị này
    private String firstName = "Test";
    private String lastName = "User";
    private String email = "test" + System.currentTimeMillis() + "@example.com";
    private String password = "Test@1234";
    private String phone = "0901234567";
    private String dateOfBirth = "1990-01-15";
    private String gender = "male";
    private String country = "Vietnam";
    private String city = "Ho Chi Minh";
    private String address = "123 Test Street";
    private String zipCode = "70000";
    private boolean subscribeNewsletter = false;
    private boolean acceptTerms = true;

    // ===== Builder methods cho từng field =====
    public RegistrationDataBuilder withEmail(String email) {
        this.email = email;
        return this;
    }

    public RegistrationDataBuilder withPassword(String password) {
        this.password = password;
        return this;
    }

    // ===== Preset methods cho các kịch bản phổ biến =====
    public RegistrationDataBuilder validData() {
        // defaults đã hợp lệ — không thay đổi gì
        return this;
    }

    public RegistrationDataBuilder withMissingEmail() {
        this.email = "";
        return this;
    }

    public RegistrationDataBuilder withInvalidEmail() {
```

```
        this.email = "not-an-email";
        return this;
    }

    public RegistrationDataBuilder withMismatchedPassword() {
        this.password = "weak";
        return this;
    }

    public RegistrationDataBuilder withoutAcceptingTerms() {
        this.acceptTerms = false;
        return this;
    }

    public RegistrationData build() {
        return new RegistrationData(
            firstName, lastName, email, password, phone, dateOfBirth,
            gender, country, city, address, zipCode,
            subscribeNewsletter, acceptTerms
        );
    }
}
```

LỚP 3 — Test class (sạch, đọc như tiếng Anh):

```
public class RegistrationTests extends BaseTest {

    @Test
    @DisplayName("User đăng ký với data hợp lệ — chuyển sang welcome page")
    void shouldRegisterSuccessfully() {
        RegistrationData data = new RegistrationDataBuilder()
            .validData()
            .build();

        DashboardPage dashboard = new RegistrationPage(driver)
            .open()
            .fillForm(data)
            .submitExpectingSuccess();

        assertThat(dashboard.getWelcomeMessage())
            .contains("Xin chào " + data.firstName());
    }

    @Test
    @DisplayName("Email không hợp lệ — hiển thị lỗi")
    void shouldShowErrorForInvalidEmail() {
        RegistrationData data = new RegistrationDataBuilder()
            .withInvalidEmail()
            .build();

        String error = new RegistrationPage(driver)
            .open()
            .fillForm(data)
            .submitExpectingError()
            .getErrorMessage();

        assertThat(error).contains("Email không đúng định dạng");
    }

    @Test
    @DisplayName("Không tick điều khoản — submit bị chặn")
    void shouldBlockSubmitWithoutAcceptingTerms() {
        RegistrationData data = new RegistrationDataBuilder()
            .withoutAcceptingTerms()
            .build();

        String error = new RegistrationPage(driver)
            .open()
            .fillForm(data)
            .submitExpectingError()
            .getErrorMessage();

        assertThat(error).contains("Bạn phải đồng ý điều khoản");
    }
}
```

Lợi ích design này:

```
- UI đổi → chỉ sửa RegistrationPage (1 file)
- Thêm field mới → thêm 1 dòng vào record + 1 dòng vào fillForm
- Kịch bản test mới → thêm preset method vào builder (1 dòng)
- Test case → đọc như tiếng Anh tự nhiên, không lo chi tiết kỹ thuật
```

Câu 8. Test SPA React — element em đang giữ bị unmount sau khi click. Code xử lý chuẩn và pattern cho team.

Vì sao interviewer hỏi: Câu này check em có hiểu React lifecycle và testing pattern hiện đại.

```
public class ReactPageHelpers {

    private final WebDriver driver;
    private final WaitUtils wait;

    public ReactPageHelpers(WebDriver driver) {
        this.driver = driver;
        this.wait = new WaitUtils(driver);
    }

    /**
     * Pattern: Chờ STATE MỚI ổn định, không chờ element cũ biến mất.
     */
    public <T> T clickAndWaitForNewState(By clickTarget, By newStateIndicator,
                                         Function<WebDriver, T> resultExtractor) {
        wait.forClickable(clickTarget).click();

        // Chờ element của state MỚI xuất hiện
        wait.forVisible(newStateIndicator);

        // Optional: chờ React idle (không còn re-render đang chạy)
        waitForReactIdle();

        return resultExtractor.apply(driver);
    }

    /**
     * Chờ React không còn pending updates.
     */
    public void waitForReactIdle() {
        wait.forCustom(d -> {
            JavascriptExecutor js = (JavascriptExecutor) d;
            try {
                // React 18 — kiểm tra concurrent rendering
                Object isIdle = js.executeScript(
                    "if (window.__REACT_DEVTOOLS_GLOBAL_HOOK__) {" +
                    "  var hook = window.__REACT_DEVTOOLS_GLOBAL_HOOK__;" +
                    "  return Object.keys(hook.renderers || {}).length > 0;" +
                    "}" +
                    "return true;"
                );
                return Boolean.TRUE.equals(isIdle);
            } catch (Exception e) {
                return true; // Không có React DevTools — coi như idle
            }
        });
    }

    /**
     * Click một button mà sau khi click, button text đổi (loading state).
     * Pattern: button cùng data-testid nhưng text khác.
     */
    public void clickAndWaitForReady(By buttonLocator) {
        WebElement button = wait.forClickable(buttonLocator);
        String originalText = button.getText();
```

```
        button.click();

        // Chờ button trở lại text gốc (loading → ready)
        wait.forCustom(d -> {
            try {
                String currentText = d.findElement(buttonLocator).getText();
                return currentText.equals(originalText);
            } catch (StaleElementReferenceException e) {
                return false;
            }
        });
    }
}
```

Sử dụng pattern này:

```
public class TodoListPage extends BasePage {

    private final By addBtn = By.cssSelector("[data-testid='add-todo']");
    private final By todoInput = By.cssSelector("[data-testid='todo-input']");
    private final By todoItems = By.cssSelector("[data-testid='todo-item']");
    private final By todoCount = By.cssSelector("[data-testid='todo-count']");

    private final ReactPageHelpers reactHelpers;

    public TodoListPage(WebDriver driver) {
        super(driver);
        this.reactHelpers = new ReactPageHelpers(driver);
    }

    /**
     * Add todo — React re-render list sau khi add.
     * Pattern: chờ count tăng thay vì giữ reference list cũ.
     */
    public TodoListPage addTodo(String text) {
        int oldCount = getTodoCount();

        type(todoInput, text);

        reactHelpers.clickAndWaitForNewState(
            addBtn,
            By.cssSelector("[data-testid='todo-item']:nth-child(" + (oldCount + 1) + ")"),
            d -> null
        );

        return this;
    }

    /**
     * Delete todo theo index — React unmount item và shift các item khác.
     */
    public TodoListPage deleteTodoAt(int index) {
        int oldCount = getTodoCount();
        String todoTextBefore = getTodoTextAt(index);

        // Click delete button của item index
        wait.forClickable(
            By.cssSelector("[data-testid='todo-item']:nth-child(" + (index + 1) + ") .delete-bt
        ).click();

        // Chờ count giảm
        wait.forCustom(d -> getTodoCount() == oldCount - 1);

        return this;
    }

    public int getTodoCount() {
        return driver.findElements(todoItems).size();
    }

    public String getTodoTextAt(int index) {
        return driver.findElements(todoItems).get(index).getText();
```

```
    }
}
```

Anti-pattern trong React testing:

```
// ❌ ĐỪNG sleep sau khi click
button.click();
Thread.sleep(500);  // React render time variable

// ❌ ĐỪNG giữ reference qua re-render
WebElement item = driver.findElement(By.cssSelector(".todo-item"));
button.click();  // có thể trigger re-render
item.getText();  // StaleElementReferenceException

// ❌ ĐỪNG dùng XPath dài dựa vào DOM structure
By.xpath("//div[3]/div[2]/ul/li[5]/span")  // React đổi DOM thường xuyên

// ✅ ĐÚNG: dùng data-testid ổn định qua mọi state
By.cssSelector("[data-testid='todo-item-5']")
```

Câu 9. Em là interviewer phỏng vấn junior cho team. Viết code 1 bài test ngắn để đánh giá ứng viên — kèm rubric chấm.

Vì sao interviewer hỏi: Câu này dành cho senior — test khả năng design assessment và đánh giá người khác.

Đề bài đưa cho ứng viên:

```
/**
 * BÀI TEST — 30 PHÚT
 *
 * Cho trang https://demo-app.com/login với:
 *   - Input email: id="email"
 *   - Input password: id="password"
 *   - Button submit: id="submit-btn"
 *   - Error message: class="error-message" (xuất hiện sau khi submit fail)
 *   - Welcome page URL chứa "/dashboard" sau khi login thành công
 *
 * Yêu cầu:
 * 1. Viết 1 test case kiểm tra login fail với password sai
 * 2. Verify error message hiển thị đúng "Invalid credentials"
 * 3. Code phải có thể chạy lại nhiều lần ổn định
 *
 * Bạn có thể dùng IDE, search Google, KHÔNG được dùng ChatGPT.
 */
public class LoginTest {
    // TODO: viết code ở đây
}
```

Rubric chấm 5 mức (interviewer dùng):

```
public class InterviewRubric {

    /**
     * MỨC 1 — KHÔNG ĐẠT (1/5)
     * Code không chạy được, syntax error, không biết cấu trúc cơ bản.
     */
    public class Level1_NotWorking {
        // findElement viết sai, không có WebDriver setup
        // → Loại ngay
    }

    /**
     * MỨC 2 — BASIC (2/5)
     * Test chạy được nhưng có Thread.sleep hoặc implicit wait kiểu mù.
     */
    public class Level2_Basic {
        @Test
        public void testLoginFail() throws InterruptedException {
            driver.get("https://demo-app.com/login");
            Thread.sleep(2000);  // ❌ Sleep mù

            driver.findElement(By.id("email")).sendKeys("test@test.com");
            driver.findElement(By.id("password")).sendKeys("wrongpass");
            driver.findElement(By.id("submit-btn")).click();

            Thread.sleep(2000);  // ❌ Sleep mù
            String error = driver.findElement(By.className("error-message")).getText();
            Assert.assertEquals(error, "Invalid credentials");
        }
        // → Có thể tuyển cho intern, nhưng cần training
    }

    /**
     * MỨC 3 — ĐẠT (3/5)
     * Có explicit wait, có teardown, không có anti-pattern lớn.
     */
    public class Level3_Acceptable {
        WebDriver driver;
        WebDriverWait wait;

        @BeforeEach
        void setUp() {
            driver = new ChromeDriver();
            wait = new WebDriverWait(driver, Duration.ofSeconds(10));
        }

        @AfterEach
        void tearDown() {
            if (driver != null) driver.quit();
        }

        @Test
        void testLoginFail() {
            driver.get("https://demo-app.com/login");

            driver.findElement(By.id("email")).sendKeys("test@test.com");
```

```
            driver.findElement(By.id("password")).sendKeys("wrongpass");
            driver.findElement(By.id("submit-btn")).click();

            WebElement error = wait.until(
                ExpectedConditions.visibilityOfElementLocated(By.className("error-message"))
            );
            Assert.assertEquals(error.getText(), "Invalid credentials");
        }
        // → Tuyển được cho junior position
    }

    /**
     * MỨC 4 — TỐT (4/5)
     * Có POM, có constants, có comment.
     */
    public class Level4_Good {
        // Có LoginPage class riêng, có URL constant
        // Có @BeforeEach setup driver, @AfterEach cleanup
        // Code đọc dễ hiểu, biến đặt tên tốt
        // → Tuyển được cho mid-level
    }

    /**
     * MỨC 5 — XUẤT SẮC (5/5)
     * Có POM + thoughts về test stability, mentions retry pattern,
     * hỏi clarifying questions về requirements trước khi code.
     */
    public class Level5_Excellent {
        // Bắt đầu bằng câu hỏi: "Login fail có nhiều case không (sai email,
        // sai password, account locked)? Em viết case nào cụ thể?"
        //
        // Đề xuất: "Em sẽ dùng @ParameterizedTest cho nhiều case fail khác nhau"
        //
        // Code có:
        // - POM pattern
        // - Constants/config riêng
        // - Explicit wait với condition cụ thể (elementToBeClickable cho button)
        // - Verify cả URL không đổi (vẫn ở /login) — không chỉ verify error text
        // - Comment giải thích lựa chọn locator
        //
        // → Tuyển được cho senior position
    }
}
```

Câu hỏi follow-up theo level:

```
public class FollowUpQuestions {

    // Hỏi MỌI ứng viên (kể cả level 2-3):
    String q1 = "Test này có thể chạy song song với 9 instance khác không? Vì sao?";
    String q2 = "Nếu password sai, em verify gì để chắc chắn login KHÔNG thành công?";

    // Hỏi level 3-4:
    String q3 = "Locator By.className('error-message') có rủi ro gì không?";
    String q4 = "Nếu sau 1 tháng dev đổi class name, em handle thế nào?";

    // Hỏi level 4-5:
    String q5 = "Suite có 200 test login như này, em refactor ra sao?";
    String q6 = "Khi nào em chuyển login qua API thay vì UI?";
}
```

Câu 10. Bàn giao suite test cho team mới. Viết README + một file documentation quan trọng nhất theo em.

Vì sao interviewer hỏi: Câu này check em có nghĩ về sustainability dài hạn.

File 1 — README.md (technical onboarding):

````
# Selenium E2E Test Suite

## Quick Start

```bash
# Yêu cầu: Java 17+, Maven 3.8+, Chrome browser
git clone <repo>
cd selenium-e2e-suite
cp .env.example .env  # Điền credentials test environment

# Chạy smoke tests (5 phút)
mvn test -Pgroups=smoke

# Chạy regression (45 phút)
mvn test -Pgroups=regression
```

## Project Structure

```
src/
├── main/java/com/company/
│   ├── config/         # ConfigLoader đọc .env và config.properties
│   ├── driver/         # DriverFactory + DriverManager (ThreadLocal)
│   ├── pages/          # Page Object classes
│   ├── utils/          # WaitUtils, ElementUtils, ApiClient
│   └── data/           # Test data builders
└── test/java/com/company/tests/
    ├── smoke/          # 5-10 critical tests, chạy mỗi PR
    ├── regression/     # 200+ tests, chạy nightly
    └── e2e/            # Full user journey tests
```

## Environment Variables

| Variable | Required | Default | Description |
|---|---|---|---|
| BASE_URL | Yes | - | App URL (e.g. https://staging.app.com) |
| TEST_USER_EMAIL | Yes | - | Account dùng cho test |
| TEST_USER_PASSWORD | Yes | - | Mật khẩu account |
| BROWSER | No | chrome | chrome, firefox, edge |
| HEADLESS | No | false | true/false |
| PARALLEL_THREADS | No | 4 | Số test chạy song song |
````

File 2 — DECISIONS.md (TÀI LIỆU QUAN TRỌNG NHẤT):

```
# Architecture Decision Records

> File này ghi lại VÌ SAO team đưa ra các quyết định kỹ thuật.
> Khi đọc code không hiểu vì sao làm cách này, hãy đọc file này TRƯỚC khi refactor.

---

## ADR-001: Dùng Selenium 4 + JUnit 5, không phải Playwright/Cypress

**Date:** 2024-03-15
**Status:** Accepted

### Context
Team có 4 dev/QA, stack Java backend, app web Angular. Cần chọn framework test E2E.

### Decision
Selenium 4 + JUnit 5 + Maven.

### Rationale
1. Toàn team đã quen Java — không cần học stack JavaScript mới
2. CI/CD đã có Maven pipeline — không cần đầu tư thêm
3. App có khách dùng Safari — Cypress không hỗ trợ Safari
4. Stability ngành 15+ năm — risk thấp khi team rotate

### Consequences
- Setup chậm hơn Cypress (~2 ngày so với 1 giờ)
- Cú pháp verbose hơn Cypress/Playwright
- Đổi lại: chạy được mọi browser, tích hợp Java ecosystem tốt

### When to reconsider
- Nếu team chuyển sang full JavaScript stack
- Nếu Selenium release breaking changes lớn
- Nếu Playwright thêm Java support tốt hơn

---

## ADR-002: KHÔNG dùng implicit wait, BẮT BUỘC explicit wait

**Date:** 2024-04-02
**Status:** Accepted

### Context
Suite ban đầu flaky 15%, debug ra do trộn implicit + explicit wait.

### Decision
- `driver.manage().timeouts().implicitlyWait(Duration.ZERO)` ở BaseTest
- Mọi wait phải dùng `WaitUtils` class
- Code review reject PR có `driver.findElement` không qua wait

### Rationale
Trộn implicit + explicit gây timeout cộng dồn không kiểm soát. Đã đo:
- Trước: avg test fail timeout 24s, flaky 15%
- Sau: avg test fail timeout 10s, flaky 2%

### Consequences
- Code dài hơn (mỗi find phải qua wait)
```

```
- Đổi lại: test deterministic, debug dễ

---

## ADR-003: Test data sinh qua API, không hard-code

**Date:** 2024-05-10
**Status:** Accepted

### Context
Test cần user data. Có 3 lựa chọn: hard-code, file JSON, sinh qua API.

### Decision
Sinh qua API với `UserDataBuilder.create()`.

### Rationale
- Test độc lập (mỗi test có user riêng, chạy song song được)
- Data luôn fresh (không bị stale như hard-code)
- API là contract ổn định hơn database schema

### Consequences
- Cần API endpoint cho test data (dev đã expose `/api/test/users`)
- Cleanup data sau test (đã có `@AfterEach` cleanup)

---

## ADR-004: Login qua API + cookie injection, KHÔNG qua UI

**Date:** 2024-06-20
**Status:** Accepted

### Context
85% test case cần login. Login qua UI mất ~6 giây/test.

### Decision
`LoginHelper.loginViaApi()` — lấy token qua POST /api/auth/login, inject vào cookie/localStorag

### Rationale
- Tiết kiệm 6s × 200 test = 20 phút/lần chạy
- Login UI vẫn được test riêng trong 5 test dedicated

### Trade-off
- Không test được full login UI mỗi lần
- Đổi lại: suite chạy 18 phút thay vì 38 phút

---

## ADR-005: Browser tools không dùng JS click

**Date:** 2024-08-05
**Status:** Accepted

### Context
Có PR đề xuất default click qua JavaScript executor để "ổn định hơn".

### Decision
JavaScript click chỉ dùng khi `ElementClickInterceptedException` thật sự xảy ra. Mọi PR có `js.
```

```
### Rationale
JS click bypass tất cả check của browser (visible, enabled, intercepted). Test pass nhưng user

### Real incident
PR-2845 dùng JS click cho nút "Mua hàng" → test pass → user thật không mua được vì có overlay a

---

## Notes for future maintainers

- **Khi muốn refactor một quyết định nào ở trên**, đọc lại "Context" và "Real incident" trước
- **Khi thêm quyết định mới**, follow format ADR này
- **Đừng xóa ADR cũ** — kể cả khi đã reverse decision, giữ lại với status "Superseded by ADR-XX
```

Buổi knowledge transfer (không phải file, nhưng critical):

```
## Knowledge Transfer Session Plan (3 buổi × 1 giờ)

### Buổi 1: Walkthrough code
- Demo chạy smoke test local
- Walk qua 1 test case từ đầu đến cuối
- Mở 1 Page Object class — giải thích pattern
- Q&A

### Buổi 2: Walkthrough infrastructure
- CI/CD pipeline (.github/workflows)
- Test data lifecycle (tạo, cleanup)
- Cách debug khi test fail trên CI

### Buổi 3: Pain points và TODOs
- 5 test đang flaky — đã biết nguyên nhân, chưa fix
- 3 page chưa có data-testid — đang ping dev
- Roadmap: chuyển sang Allure 3.x quý sau

→ Record video lại các buổi này. Người mới join 6 tháng sau xem lại được.
```

Phần 2 — Framework và best practices (10 câu)

Câu 11. Cấu trúc project Maven chuẩn cho Selenium

```
selenium-framework/
├── pom.xml
├── src/
│   ├── main/java/
│   │   └── com/company/automation/
│   │       ├── config/         # config loader, properties
│   │       ├── driver/         # DriverFactory, DriverManager
│   │       ├── pages/          # Page Objects
│   │       │   ├── BasePage.java
│   │       │   ├── LoginPage.java
│   │       │   └── DashboardPage.java
│   │       ├── components/     # reusable UI parts (header, modal)
│   │       └── utils/          # helpers, waits, file ops
│   └── test/
│       ├── java/
│       │   └── com/company/automation/tests/
│       │       ├── BaseTest.java
│       │       ├── LoginTests.java
│       │       └── CheckoutTests.java
│       └── resources/
│           ├── config.properties
│           ├── testdata/        # JSON, CSV test data
│           └── log4j2.xml
└── target/
    ├── allure-results/
    └── screenshots/
```

pom.xml chính (Selenium 4 + JUnit 5):

```
<dependencies>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>4.18.1</version>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter</artifactId>
        <version>5.10.2</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>io.qameta.allure</groupId>
        <artifactId>allure-junit5</artifactId>
        <version>2.25.0</version>
    </dependency>
    <dependency>
        <groupId>org.assertj</groupId>
        <artifactId>assertj-core</artifactId>
        <version>3.25.3</version>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>3.2.5</version>
            <configuration>
                <parallel>methods</parallel>
                <threadCount>4</threadCount>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Câu 12. Parallel Execution với JUnit 5

junit-platform.properties:

```
junit.jupiter.execution.parallel.enabled=true
junit.jupiter.execution.parallel.mode.default=concurrent
junit.jupiter.execution.parallel.mode.classes.default=concurrent
junit.jupiter.execution.parallel.config.strategy=fixed
junit.jupiter.execution.parallel.config.fixed.parallelism=4
```

ThreadLocal cho WebDriver:

```
public class DriverManager {
    private static final ThreadLocal<WebDriver> driverThread = new ThreadLocal<>();

    public static WebDriver getDriver() {
        if (driverThread.get() == null) {
            driverThread.set(createDriver());
        }
        return driverThread.get();
    }

    public static void quitDriver() {
        WebDriver driver = driverThread.get();
        if (driver != null) {
            driver.quit();
            driverThread.remove();
        }
    }

    private static WebDriver createDriver() {
        ChromeOptions options = new ChromeOptions();
        options.addArguments("--headless=new");
        return new ChromeDriver(options);
    }
}

public class BaseTest {
    @BeforeEach
    void setup() {
        DriverManager.getDriver();
    }

    @AfterEach
    void teardown() {
        DriverManager.quitDriver();
    }
}
```

Quy tắc test phải tuân thủ parallel:

```
// ❌ Không dùng static state chia sẻ
public class BadTest {
    static String userId;  // sẽ bị race condition

    @Test void test1() { userId = "123"; }
    @Test void test2() { assertEquals("123", userId); }  // không đảm bảo
}

// ✅ Mỗi test có data riêng
public class GoodTest {
    @Test void test1() {
        String userId = generateUniqueId();
        // ...
    }
}
```

Câu 13. Selenium Grid 4 với Docker Compose

docker-compose.yml:

```
version: "3"
services:
  selenium-hub:
    image: selenium/hub:4.18.1
    container_name: selenium-hub
    ports:
      - "4442:4442"
      - "4443:4443"
      - "4444:4444"

  chrome:
    image: selenium/node-chrome:4.18.1
    shm_size: 2gb
    depends_on:
      - selenium-hub
    environment:
      - SE_EVENT_BUS_HOST=selenium-hub
      - SE_EVENT_BUS_PUBLISH_PORT=4442
      - SE_EVENT_BUS_SUBSCRIBE_PORT=4443
      - SE_NODE_MAX_SESSIONS=4
    deploy:
      replicas: 3

  firefox:
    image: selenium/node-firefox:4.18.1
    shm_size: 2gb
    depends_on:
      - selenium-hub
    environment:
      - SE_EVENT_BUS_HOST=selenium-hub
      - SE_EVENT_BUS_PUBLISH_PORT=4442
      - SE_EVENT_BUS_SUBSCRIBE_PORT=4443
```

Khởi tạo RemoteWebDriver:

```
import org.openqa.selenium.remote.RemoteWebDriver;
import java.net.URL;

public class GridDriverFactory {

    public static WebDriver createRemoteDriver(String browser) throws Exception {
        URL hubUrl = new URL("http://localhost:4444/wd/hub");

        switch (browser.toLowerCase()) {
            case "chrome":
                ChromeOptions chromeOpts = new ChromeOptions();
                chromeOpts.addArguments("--headless=new");
                return new RemoteWebDriver(hubUrl, chromeOpts);

            case "firefox":
                FirefoxOptions firefoxOpts = new FirefoxOptions();
                firefoxOpts.addArguments("-headless");
                return new RemoteWebDriver(hubUrl, firefoxOpts);

            default:
                throw new IllegalArgumentException("Unknown browser: " + browser);
        }
    }
}
```

Khởi động grid:

```
docker-compose up -d
# Truy cập http://localhost:4444 để xem grid console
```

Câu 14. Test Data Management — JSON + builder pattern

testdata/users.json:

```
{
  "validUser": {
    "email": "test@example.com",
    "password": "Test@123",
    "fullName": "Nguyễn Văn A"
  },
  "invalidUser": {
    "email": "invalid@example.com",
    "password": "wrongpass"
  }
}
```

Loader:

```
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.JsonNode;

public class TestDataLoader {
    private static final ObjectMapper mapper = new ObjectMapper();
    private static JsonNode root;

    static {
        try {
            InputStream is = TestDataLoader.class.getResourceAsStream("/testdata/users.json");
            root = mapper.readTree(is);
        } catch (Exception e) {
            throw new RuntimeException("Failed to load test data", e);
        }
    }

    public static User getUser(String key) {
        return mapper.convertValue(root.get(key), User.class);
    }
}
```

Builder pattern cho data sinh động:

```
public class UserBuilder {
    private String email = "user" + UUID.randomUUID() + "@test.com";
    private String password = "Test@123";
    private boolean verified = false;
    private List<String> roles = new ArrayList<>();

    public UserBuilder withEmail(String email) {
        this.email = email;
        return this;
    }

    public UserBuilder verified() {
        this.verified = true;
        return this;
    }

    public UserBuilder withRole(String role) {
        this.roles.add(role);
        return this;
    }

    public User build() {
        return new User(email, password, verified, roles);
    }

    // Tạo qua API thay vì UI
    public User create() {
        User user = build();
        return apiClient.createUser(user);  // gọi API thật
    }
}

// Sử dụng:
@Test
void testAdminCanAccessSettings() {
    User admin = new UserBuilder()
        .verified()
        .withRole("admin")
        .create();  // tạo qua API, nhanh hơn UI

    new LoginPage(driver)
        .loginAs(admin.getEmail(), admin.getPassword())
        .openSettings();
}
```

Câu 15. Login một lần — Cookie injection

LoginHelper.java:

```
import org.openqa.selenium.Cookie;
import com.fasterxml.jackson.databind.ObjectMapper;
import java.io.File;
import java.util.Set;

public class LoginHelper {
    private static final File COOKIE_FILE = new File("target/cookies.json");
    private static final ObjectMapper mapper = new ObjectMapper();

    // Gọi 1 lần ở @BeforeAll
    public static void saveSessionCookies(WebDriver driver, String email, String password) {
        new LoginPage(driver).open().loginAs(email, password);

        Set<Cookie> cookies = driver.manage().getCookies();
        try {
            mapper.writeValue(COOKIE_FILE, cookies);
        } catch (Exception e) {
            throw new RuntimeException("Cannot save cookies", e);
        }
    }

    // Gọi ở @BeforeEach của mỗi test
    public static void restoreSessionCookies(WebDriver driver, String baseUrl) {
        // Phải get domain trước khi add cookie
        driver.get(baseUrl);

        try {
            Cookie[] cookies = mapper.readValue(COOKIE_FILE, Cookie[].class);
            for (Cookie c : cookies) {
                driver.manage().addCookie(c);
            }
        } catch (Exception e) {
            throw new RuntimeException("Cannot restore cookies", e);
        }

        driver.navigate().refresh();
    }
}
```

Cách nhanh hơn — login qua API:

```
public class ApiLoginHelper {

    public static String getAuthToken(String email, String password) {
        // Gọi API login endpoint
        Response response = RestAssured.given()
            .contentType("application/json")
            .body(Map.of("email", email, "password", password))
            .post("https://api.example.com/auth/login");

        return response.jsonPath().getString("token");
    }

    public static void injectAuthToken(WebDriver driver, String token) {
        driver.get("https://app.example.com");  // load domain trước

        // Set vào localStorage
        ((JavascriptExecutor) driver).executeScript(
            "window.localStorage.setItem('authToken', arguments[0]);", token
        );

        // Hoặc set vào cookie
        driver.manage().addCookie(new Cookie("auth_token", token, ".example.com", "/", null));

        driver.navigate().refresh();
    }
}

@BeforeEach
void loginViaApi() {
    String token = ApiLoginHelper.getAuthToken("test@example.com", "Test@123");
    ApiLoginHelper.injectAuthToken(driver, token);
    // Test bắt đầu ở trạng thái đã đăng nhập, không cần UI login
}
```

Câu 16. Retry Mechanism cho Flaky Tests

JUnit 5 với @RetryingTest (từ junit-pioneer):

```
import org.junitpioneer.jupiter.RetryingTest;

@RetryingTest(maxAttempts = 2, suspendForMs = 1000)
void testLoginShouldSucceed() {
    // Test sẽ retry tối đa 1 lần nếu fail
}
```

Custom Extension cho retry với log:

```
import org.junit.jupiter.api.extension.*;

public class RetryExtension implements TestExecutionExceptionHandler {
    private static final int MAX_RETRIES = 1;

    @Override
    public void handleTestExecutionException(
        ExtensionContext context, Throwable throwable) throws Throwable {

        Integer count = context.getStore(ExtensionContext.Namespace.GLOBAL)
            .getOrDefault("retryCount", Integer.class, 0);

        if (count >= MAX_RETRIES) {
            throw throwable;  // fail thật sự
        }

        context.getStore(ExtensionContext.Namespace.GLOBAL)
            .put("retryCount", count + 1);

        System.out.println("⚠️ Test failed, retrying... (attempt " + (count + 1) + ")");
        // Re-run test method
        Method method = context.getRequiredTestMethod();
        method.invoke(context.getRequiredTestInstance());
    }
}
```

Quy tắc retry:

```
// ✅ Retry tối đa 1 lần
// ✅ Log mỗi lần retry để theo dõi flaky
// ❌ Đừng retry 3-5 lần — đó là che bug, không phải fix
// ❌ Đừng retry mà không log — không biết flaky đến mức nào
```

Câu 17. Logging với SLF4J + Log4j2

log4j2.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="WARN">
    <Appenders>
        <Console name="Console" target="SYSTEM_OUT">
            <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"/>
        </Console>
        <RollingFile name="RollingFile"
                     fileName="target/logs/test.log"
                     filePattern="target/logs/test-%d{yyyy-MM-dd}.log">
            <PatternLayout pattern="%d %p %c{1.} [%t] %m%n"/>
            <Policies>
                <TimeBasedTriggeringPolicy/>
            </Policies>
        </RollingFile>
    </Appenders>
    <Loggers>
        <Logger name="com.company.automation" level="DEBUG"/>
        <Logger name="org.openqa.selenium" level="WARN"/>
        <Root level="INFO">
            <AppenderRef ref="Console"/>
            <AppenderRef ref="RollingFile"/>
        </Root>
    </Loggers>
</Configuration>
```

Sử dụng trong Page Object:

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class LoginPage extends BasePage {
    private static final Logger log = LoggerFactory.getLogger(LoginPage.class);

    public DashboardPage loginAs(String email, String password) {
        log.info("Attempting login with email: {}", email);

        try {
            type(emailInput, email);
            type(passwordInput, "***");  // không log password
            log.debug("Credentials entered, clicking submit");

            click(submitBtn);
            log.info("Login successful, navigating to dashboard");

            return new DashboardPage(driver);
        } catch (Exception e) {
            log.error("Login failed for email: {}", email, e);
            throw e;
        }
    }
}
```

Câu 18. Allure Report — annotation và setup

```
import io.qameta.allure.*;

@Epic("E-commerce")
@Feature("Checkout")
public class CheckoutTests {

    @Test
    @Story("Apply discount code")
    @Severity(SeverityLevel.CRITICAL)
    @Description("User can apply valid discount code and see updated total")
    @Issue("JIRA-1234")
    @TmsLink("TEST-5678")
    void testApplyValidDiscountCode() {
        applyDiscount("SAVE10");
        verifyDiscountApplied();
    }

    @Step("Apply discount code: {code}")
    public void applyDiscount(String code) {
        // step sẽ hiển thị trong Allure report
        new CartPage(driver).enterDiscount(code).apply();
    }

    @Step("Verify discount was applied successfully")
    public void verifyDiscountApplied() {
        Assertions.assertTrue(new CartPage(driver).isDiscountVisible());
    }

    // Attachment
    @Attachment(value = "Screenshot", type = "image/png")
    public byte[] takeScreenshot() {
        return ((TakesScreenshot) driver).getScreenshotAs(OutputType.BYTES);
    }

    @Attachment(value = "Page source", type = "text/html")
    public String capturePageSource() {
        return driver.getPageSource();
    }
}
```

Chạy và xem report:

```
# Chạy test
mvn clean test

# Sinh report
mvn allure:report

# Serve report local
mvn allure:serve
```

Câu 19. CI/CD với GitHub Actions

.github/workflows/selenium-tests.yml:

```yaml
name: Selenium Tests

on:
  pull_request:
    branches: [main]
  schedule:
    - cron: '0 2 * * *'  # 2 AM mỗi ngày

jobs:
  smoke-tests:
    runs-on: ubuntu-latest
    timeout-minutes: 15
    steps:
      - uses: actions/checkout@v4

      - name: Setup JDK 17
        uses: actions/setup-java@v4
        with:
          java-version: '17'
          distribution: 'temurin'
          cache: maven

      - name: Run smoke tests
        run: mvn test -Dgroups=smoke -Dbrowser=chrome -Dheadless=true

      - name: Upload Allure results
        if: always()
        uses: actions/upload-artifact@v4
        with:
          name: allure-results
          path: target/allure-results

      - name: Upload screenshots on failure
        if: failure()
        uses: actions/upload-artifact@v4
        with:
          name: screenshots
          path: target/screenshots

  full-regression:
    if: github.event_name == 'schedule'
    runs-on: ubuntu-latest
    timeout-minutes: 60

    services:
      selenium-hub:
        image: selenium/hub:4.18.1
        ports:
          - 4444:4444
      chrome:
        image: selenium/node-chrome:4.18.1
        env:
          SE_EVENT_BUS_HOST: selenium-hub
          SE_EVENT_BUS_PUBLISH_PORT: 4442
          SE_EVENT_BUS_SUBSCRIBE_PORT: 4443
```

```
    steps:
      - uses: actions/checkout@v4
      - name: Run regression
        run: mvn test -Dremote=true -DhubUrl=http://localhost:4444

      - name: Notify Slack on failure
        if: failure()
        uses: slackapi/slack-github-action@v1
        with:
          payload: |
            {
              "text": "🚨 Nightly regression failed",
              "url": "${{ github.server_url }}/${{ github.repository }}/actions/runs/${{ github
            }
        env:
          SLACK_WEBHOOK_URL: ${{ secrets.SLACK_WEBHOOK_URL }}
```

Câu 20. Cross-browser config qua System Properties

DriverFactory mở rộng:

```
public class DriverFactory {

    public static WebDriver create() {
        String browser = System.getProperty("browser", "chrome").toLowerCase();
        boolean headless = Boolean.parseBoolean(System.getProperty("headless", "false"));
        boolean remote = Boolean.parseBoolean(System.getProperty("remote", "false"));

        if (remote) {
            return createRemote(browser, headless);
        }
        return createLocal(browser, headless);
    }

    private static WebDriver createLocal(String browser, boolean headless) {
        switch (browser) {
            case "chrome": return createChrome(headless);
            case "firefox": return createFirefox(headless);
            case "safari": return new SafariDriver();
            case "edge": return createEdge(headless);
            default: throw new IllegalArgumentException("Unknown browser: " + browser);
        }
    }

    private static WebDriver createChrome(boolean headless) {
        ChromeOptions options = new ChromeOptions();
        if (headless) options.addArguments("--headless=new");
        options.addArguments("--window-size=1920,1080");
        options.addArguments("--disable-dev-shm-usage");
        options.addArguments("--no-sandbox");
        return new ChromeDriver(options);
    }

    private static WebDriver createFirefox(boolean headless) {
        FirefoxOptions options = new FirefoxOptions();
        if (headless) options.addArguments("-headless");
        options.addArguments("-width=1920");
        options.addArguments("-height=1080");
        return new FirefoxDriver(options);
    }

    private static WebDriver createEdge(boolean headless) {
        EdgeOptions options = new EdgeOptions();
        if (headless) options.addArguments("--headless=new");
        return new EdgeDriver(options);
    }

    private static WebDriver createRemote(String browser, boolean headless) {
        String hubUrl = System.getProperty("hubUrl", "http://localhost:4444/wd/hub");
        try {
            return new RemoteWebDriver(new URL(hubUrl), getOptions(browser, headless));
        } catch (Exception e) {
            throw new RuntimeException("Cannot connect to grid: " + hubUrl, e);
        }
    }
}
```

Chạy với browser khác nhau:

```
mvn test -Dbrowser=chrome
mvn test -Dbrowser=firefox -Dheadless=true
mvn test -Dbrowser=edge -Dremote=true -DhubUrl=http://grid:4444/wd/hub
```

Phần 3 — Nâng cao và kỹ thuật đặc biệt (10 câu)

Câu 21. Selenium 4 Relative Locators

```
import static org.openqa.selenium.support.locators.RelativeLocator.with;

// "Tìm input nằm BÊN PHẢI label 'Email'"
WebElement emailInput = driver.findElement(
    with(By.tagName("input")).toRightOf(By.xpath("//label[text()='Email']"))
);

// "Tìm checkbox NẰM DƯỚI label 'Remember me'"
WebElement rememberMe = driver.findElement(
    with(By.tagName("input")).below(By.xpath("//label[text()='Remember me']"))
);

// "Tìm button 'Save' GẦN nhất với 'Cancel'"
WebElement saveBtn = driver.findElement(
    with(By.tagName("button")).near(By.xpath("//button[text()='Cancel']"))
);

// CHAIN nhiều relative
WebElement target = driver.findElement(
    with(By.tagName("input"))
        .toRightOf(By.id("label-email"))
        .above(By.id("submit-btn"))
);
```

Khi nào dùng:

```
// ✅ Visual layout testing
WebElement priceLabel = driver.findElement(By.cssSelector(".price"));
WebElement discountBadge = driver.findElement(
    with(By.cssSelector(".badge")).toLeftOf(priceLabel)
);
// Verify discount badge nằm bên trái price

// ❌ KHÔNG dùng cho test functional thông thường
// Layout đổi nhẹ → test vỡ
// CSS Selector ổn định hơn cho test thường
```

Câu 22. Chrome DevTools Protocol — Network Interception

```
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.devtools.DevTools;
import org.openqa.selenium.devtools.v120.network.Network;
import org.openqa.selenium.devtools.v120.network.model.RequestId;

ChromeDriver driver = new ChromeDriver();
DevTools devTools = driver.getDevTools();
devTools.createSession();
devTools.send(Network.enable(Optional.empty(), Optional.empty(), Optional.empty()));

// 1. LOG mọi network request
devTools.addListener(Network.requestWillBeSent(), request -> {
    System.out.println("➡️ " + request.getRequest().getMethod() + " " +
                       request.getRequest().getUrl());
});

devTools.addListener(Network.responseReceived(), response -> {
    System.out.println("⬅️ " + response.getResponse().getStatus() + " " +
                       response.getResponse().getUrl());
});

// 2. BLOCK request
devTools.send(Network.setBlockedURLs(List.of(
    "*google-analytics*",
    "*facebook.com/tr*",
    "*.png"  // chặn ảnh để test load nhanh hơn
)));

// 3. GIẢ LẬP network chậm (3G)
devTools.send(Network.emulateNetworkConditions(
    false,           // offline
    100,             // latency ms
    50 * 1024,       // download bytes/s
    20 * 1024,       // upload bytes/s
    Optional.of(ConnectionType.CELLULAR3G)
));

// 4. MOCK response
devTools.send(Fetch.enable(...));
devTools.addListener(Fetch.requestPaused(), request -> {
    if (request.getRequest().getUrl().contains("/api/user")) {
        // Trả về fake response
        devTools.send(Fetch.fulfillRequest(
            request.getRequestId(),
            200,
            Optional.empty(),
            Optional.empty(),
            Optional.of(Base64.getEncoder().encodeToString(
                "{\"name\":\"Mocked User\"}".getBytes()
            )),
            Optional.empty()
        ));
```

```
    }
});
```

Câu 23. CDP — Console logs, performance metrics

```
DevTools devTools = ((ChromeDriver) driver).getDevTools();
devTools.createSession();

// 1. CAPTURE console logs từ trang
devTools.send(Log.enable());
devTools.addListener(Log.entryAdded(), entry -> {
    System.out.println("[" + entry.getLevel() + "] " + entry.getText());
});

// 2. CAPTURE JS exceptions
devTools.send(Runtime.enable());
devTools.addListener(Runtime.exceptionThrown(), exc -> {
    System.err.println("❌ JS Exception: " + exc.getExceptionDetails().getText());
});

// 3. PERFORMANCE METRICS
devTools.send(Performance.enable(Optional.empty()));

driver.get("https://example.com");

List<Metric> metrics = devTools.send(Performance.getMetrics());
for (Metric m : metrics) {
    System.out.println(m.getName() + ": " + m.getValue());
}

// Đo Largest Contentful Paint, First Contentful Paint
String perfData = (String) ((JavascriptExecutor) driver).executeScript(
    "var perfData = window.performance.getEntriesByType('paint');" +
    "return JSON.stringify(perfData);"
);
System.out.println("Paint metrics: " + perfData);

// 4. EMULATE device
devTools.send(Emulation.setDeviceMetricsOverride(
    375, 667, 2.0, true,
    Optional.empty(), Optional.empty(), Optional.empty(),
    Optional.empty(), Optional.empty(), Optional.empty(),
    Optional.empty(), Optional.empty(), Optional.empty()
));
```

Câu 24. Visual Regression Testing

Cách 1: Compare screenshots với AShot:

```
import ru.yandex.qatools.ashot.AShot;
import ru.yandex.qatools.ashot.Screenshot;
import ru.yandex.qatools.ashot.comparison.ImageDiff;
import ru.yandex.qatools.ashot.comparison.ImageDiffer;
import ru.yandex.qatools.ashot.shooting.ShootingStrategies;

public class VisualRegression {

    public static void compareWithBaseline(WebDriver driver, String testName) throws IOExceptio
        Screenshot actual = new AShot()
            .shootingStrategy(ShootingStrategies.viewportPasting(100))
            .takeScreenshot(driver);

        BufferedImage actualImage = actual.getImage();
        File baselineFile = new File("baselines/" + testName + ".png");

        if (!baselineFile.exists()) {
            // Lần đầu — lưu làm baseline
            ImageIO.write(actualImage, "PNG", baselineFile);
            System.out.println("✅ Baseline created: " + baselineFile);
            return;
        }

        BufferedImage expected = ImageIO.read(baselineFile);

        ImageDiff diff = new ImageDiffer()
            .withDiffSizeTrigger(20)  // tolerance pixel
            .makeDiff(expected, actualImage);

        if (diff.hasDiff()) {
            File diffFile = new File("diffs/" + testName + ".png");
            ImageIO.write(diff.getMarkedImage(), "PNG", diffFile);
            throw new AssertionError("Visual diff detected. See: " + diffFile);
        }
    }
}

// Sử dụng:
@Test
void homePageShouldMatchBaseline() throws IOException {
    driver.get("https://example.com");
    wait.until(d -> ((JavascriptExecutor) d)
        .executeScript("return document.readyState").equals("complete"));

    VisualRegression.compareWithBaseline(driver, "home-page");
}
```

Cách 2: Tích hợp Applitools / Percy:

```
// Applitools example
Eyes eyes = new Eyes();
eyes.setApiKey(System.getenv("APPLITOOLS_API_KEY"));

try {
    eyes.open(driver, "MyApp", "Login Page Test");
    driver.get("https://example.com/login");
    eyes.checkWindow("Login Page");
    eyes.close();
} finally {
    eyes.abortIfNotClosed();
}
```

Câu 25. Mobile Web Testing với Chrome Emulation

```
import org.openqa.selenium.chrome.ChromeOptions;
import java.util.Map;

// CÁCH 1: Emulate device có sẵn
ChromeOptions options = new ChromeOptions();
Map<String, String> mobileEmulation = Map.of("deviceName", "iPhone 12 Pro");
options.setExperimentalOption("mobileEmulation", mobileEmulation);
WebDriver driver = new ChromeDriver(options);

// CÁCH 2: Custom device
Map<String, Object> deviceMetrics = Map.of(
    "width", 375,
    "height", 812,
    "pixelRatio", 3.0
);
Map<String, Object> mobileSettings = Map.of(
    "deviceMetrics", deviceMetrics,
    "userAgent", "Mozilla/5.0 (iPhone; CPU iPhone OS 15_0 like Mac OS X)..."
);
options.setExperimentalOption("mobileEmulation", mobileSettings);

// CÁCH 3: Test responsive bằng resize window
driver.manage().window().setSize(new Dimension(375, 667));
driver.get("https://example.com");

// Check mobile menu xuất hiện
assertTrue(driver.findElement(By.cssSelector(".hamburger-menu")).isDisplayed());

// Test khác viewport sizes
int[][] viewports = {
    {320, 568},   // iPhone SE
    {375, 667},   // iPhone 8
    {768, 1024},  // iPad
    {1920, 1080}  // Desktop
};

for (int[] vp : viewports) {
    driver.manage().window().setSize(new Dimension(vp[0], vp[1]));
    System.out.println("Testing at " + vp[0] + "x" + vp[1]);
    // chạy assertion responsive
}
```

Câu 26. Database verification trong test

```
import java.sql.*;

public class DatabaseHelper {
    private static final String URL = "jdbc:postgresql://localhost:5432/testdb";
    private static final String USER = "test";
    private static final String PASS = "test";

    public static User getUserByEmail(String email) {
        String sql = "SELECT id, email, full_name, status FROM users WHERE email = ?";

        try (Connection conn = DriverManager.getConnection(URL, USER, PASS);
             PreparedStatement stmt = conn.prepareStatement(sql)) {

            stmt.setString(1, email);
            ResultSet rs = stmt.executeQuery();

            if (rs.next()) {
                return new User(
                    rs.getLong("id"),
                    rs.getString("email"),
                    rs.getString("full_name"),
                    rs.getString("status")
                );
            }
            return null;

        } catch (SQLException e) {
            throw new RuntimeException("DB query failed", e);
        }
    }

    public static void cleanupTestUsers() {
        String sql = "DELETE FROM users WHERE email LIKE 'test_%@example.com'";
        try (Connection conn = DriverManager.getConnection(URL, USER, PASS);
             Statement stmt = conn.createStatement()) {
            int deleted = stmt.executeUpdate(sql);
            System.out.println("Cleaned up " + deleted + " test users");
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }
}

// Sử dụng:
@Test
void testRegistrationCreatesUserInDB() {
    String email = "test_" + UUID.randomUUID() + "@example.com";

    // Register qua UI
    new RegisterPage(driver).open().registerWith(email, "Test@123");

    // Verify trực tiếp ở DB
    User user = DatabaseHelper.getUserByEmail(email);
    assertNotNull(user, "User phải được tạo trong DB");
```

```
    assertEquals("pending", user.getStatus());
}

@AfterEach
void cleanup() {
    DatabaseHelper.cleanupTestUsers();
}
```

Quan điểm: dùng DB query cho **verify**, ưu tiên dùng API cho **setup**. Tránh INSERT trực tiếp vào DB vì gắn test vào schema.

Câu 27. BrowserStack / Sauce Labs integration

```
import org.openqa.selenium.MutableCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;
import java.net.URL;

public class CloudDriverFactory {

    public static WebDriver createBrowserStackDriver(String testName) throws Exception {
        String userName = System.getenv("BROWSERSTACK_USER");
        String accessKey = System.getenv("BROWSERSTACK_KEY");
        URL url = new URL("https://hub-cloud.browserstack.com/wd/hub");

        MutableCapabilities caps = new MutableCapabilities();

        // Browser
        caps.setCapability("browserName", "Chrome");
        caps.setCapability("browserVersion", "latest");

        // OS
        Map<String, Object> bstackOptions = new HashMap<>();
        bstackOptions.put("os", "Windows");
        bstackOptions.put("osVersion", "11");
        bstackOptions.put("buildName", "Selenium Build " + System.getenv("BUILD_NUMBER"));
        bstackOptions.put("sessionName", testName);
        bstackOptions.put("projectName", "MyApp E2E");
        bstackOptions.put("seleniumVersion", "4.18.1");
        bstackOptions.put("debug", true);
        bstackOptions.put("networkLogs", true);
        bstackOptions.put("video", true);
        bstackOptions.put("userName", userName);
        bstackOptions.put("accessKey", accessKey);

        caps.setCapability("bstack:options", bstackOptions);

        return new RemoteWebDriver(url, caps);
    }

    // Test trên thiết bị mobile thật
    public static WebDriver createMobileDriver(String testName) throws Exception {
        MutableCapabilities caps = new MutableCapabilities();

        Map<String, Object> bstackOptions = new HashMap<>();
        bstackOptions.put("deviceName", "iPhone 14 Pro");
        bstackOptions.put("osVersion", "16");
        bstackOptions.put("realMobile", true);
        bstackOptions.put("sessionName", testName);
        bstackOptions.put("userName", System.getenv("BROWSERSTACK_USER"));
        bstackOptions.put("accessKey", System.getenv("BROWSERSTACK_KEY"));

        caps.setCapability("browserName", "Safari");
        caps.setCapability("bstack:options", bstackOptions);

        return new RemoteWebDriver(
            new URL("https://hub-cloud.browserstack.com/wd/hub"), caps
        );
```

```
    }
}
```

Mark test status trên BrowserStack:

```
@AfterEach
void markTestStatus(TestInfo testInfo) {
    boolean passed = testInfo.getTags().contains("passed");
    String status = passed ? "passed" : "failed";
    String reason = passed ? "All assertions passed" : "Test failed";

    ((JavascriptExecutor) driver).executeScript(
        "browserstack_executor: {\"action\": \"setSessionStatus\", " +
        "\"arguments\": {\"status\":\"" + status + "\", \"reason\": \"" + reason + "\"}}"
    );
}
```

Câu 28. Custom ExpectedConditions

```
import org.openqa.selenium.support.ui.ExpectedCondition;

public class CustomConditions {

    // Chờ element có specific attribute value
    public static ExpectedCondition<Boolean> attributeToBe(
        WebElement element, String attribute, String expectedValue) {

        return driver -> {
            try {
                return element.getAttribute(attribute).equals(expectedValue);
            } catch (StaleElementReferenceException e) {
                return false;
            }
        };
    }

    // Chờ table có ít nhất N row
    public static ExpectedCondition<List<WebElement>> tableToHaveAtLeastRows(
        By tableLocator, int minRows) {

        return driver -> {
            List<WebElement> rows = driver.findElements(
                new ByChained(tableLocator, By.tagName("tr"))
            );
            return rows.size() >= minRows ? rows : null;
        };
    }

    // Chờ download hoàn tất (file xuất hiện và stable size)
    public static ExpectedCondition<Path> fileToBeDownloaded(Path expectedFile) {
        return driver -> {
            if (!Files.exists(expectedFile)) return null;
            try {
                long size1 = Files.size(expectedFile);
                Thread.sleep(500);
                long size2 = Files.size(expectedFile);
                return size1 == size2 && size1 > 0 ? expectedFile : null;
            } catch (Exception e) {
                return null;
            }
        };
    }

    // Chờ jQuery / Ajax requests done
    public static ExpectedCondition<Boolean> ajaxCompleted() {
        return driver -> {
            JavascriptExecutor js = (JavascriptExecutor) driver;
            try {
                Long active = (Long) js.executeScript("return jQuery.active");
                return active != null && active == 0;
            } catch (Exception e) {
                return true;  // không có jQuery
            }
```

```
        };
    }

    // Chờ React app idle
    public static ExpectedCondition<Boolean> reactToBeIdle() {
        return driver -> {
            JavascriptExecutor js = (JavascriptExecutor) driver;
            return (Boolean) js.executeScript(
                "return document.readyState === 'complete' && " +
                "(!window.React || !window.React.__SECRET_DOM_HANDLERS) "
            );
        };
    }
}

// Sử dụng:
wait.until(CustomConditions.attributeToBe(element, "aria-expanded", "true"));
wait.until(CustomConditions.tableToHaveAtLeastRows(By.id("results"), 10));
wait.until(CustomConditions.fileToBeDownloaded(Path.of("/tmp/report.pdf")));
```

Câu 29. Accessibility Testing với axe-core

```
import com.deque.html.axecore.selenium.AxeBuilder;
import com.deque.html.axecore.results.Results;
import com.deque.html.axecore.results.Rule;

public class AccessibilityHelper {

    public static void checkAccessibility(WebDriver driver, String pageName) {
        Results results = new AxeBuilder()
            .withTags(Arrays.asList("wcag2a", "wcag2aa", "best-practice"))
            .analyze(driver);

        List<Rule> violations = results.getViolations();

        if (!violations.isEmpty()) {
            StringBuilder report = new StringBuilder();
            report.append("♿ Accessibility violations on ").append(pageName).append(":\n\n");

            for (Rule v : violations) {
                report.append("❌ ").append(v.getId())
                    .append(" [").append(v.getImpact()).append("]\n")
                    .append("   ").append(v.getDescription()).append("\n")
                    .append("   Affected: ").append(v.getNodes().size()).append(" elements\n")
                    .append("   Help: ").append(v.getHelpUrl()).append("\n\n");
            }

            // Save JSON for detailed report
            try {
                new AxeReporter().writeResultsToJsonFile(
                    "target/axe-reports/" + pageName, results
                );
            } catch (Exception ignored) {}

            throw new AssertionError(report.toString());
        }
    }
}

// Sử dụng:
@Test
void homepageShouldBeAccessible() {
    driver.get("https://example.com");
    AccessibilityHelper.checkAccessibility(driver, "homepage");
}

// Hoặc check chỉ một region:
Results results = new AxeBuilder()
    .include(Arrays.asList("#main-content"))
    .exclude(Arrays.asList(".ads-banner"))
    .analyze(driver);
```

Câu 30. Page Factory và @FindBy

```
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
import org.openqa.selenium.support.How;

public class LoginPagePageFactory {

    @FindBy(id = "email")
    private WebElement emailInput;

    @FindBy(css = "[data-testid='password']")
    private WebElement passwordInput;

    @FindBy(xpath = "//button[@type='submit']")
    private WebElement submitBtn;

    @FindBys({
        @FindBy(css = ".error-container"),
        @FindBy(tagName = "span")
    })
    private List<WebElement> errorMessages;

    @FindAll({
        @FindBy(className = "warning"),
        @FindBy(className = "alert-danger")
    })
    private List<WebElement> allAlerts;

    private WebDriver driver;

    public LoginPagePageFactory(WebDriver driver) {
        this.driver = driver;
        PageFactory.initElements(driver, this);
    }

    public void login(String email, String password) {
        emailInput.sendKeys(email);
        passwordInput.sendKeys(password);
        submitBtn.click();
    }
}
```

Lazy initialization với AjaxElementLocatorFactory:

```
import org.openqa.selenium.support.pagefactory.AjaxElementLocatorFactory;

public LoginPagePageFactory(WebDriver driver) {
    this.driver = driver;
    // 10s wait built-in cho mọi @FindBy
    PageFactory.initElements(new AjaxElementLocatorFactory(driver, 10), this);
}
```

⚠️ Page Factory có nên dùng không?

```
// Pros:
// - Code clean với @FindBy annotations
// - Lazy initialization

// Cons:
// - Có thể gây StaleElementReferenceException nhiều hơn
// - Khó debug khi locator sai (lỗi xuất hiện ở lúc dùng, không phải khi init)
// - Không support By chains phức tạp
// - Selenium team không khuyến khích từ Selenium 4

// Quan điểm hiện đại:
// Dùng classic POM (như Câu 4) — explicit, dễ debug, dễ refactor
public class ModernLoginPage extends BasePage {
    private final By emailInput = By.id("email");  // explicit By locator

    public void login(String email, String pass) {
        type(emailInput, email);  // find ngay trước khi dùng → tránh stale
    }
}
```

Lời kết — Checklist trước phỏng vấn

Code chuẩn bị mang theo:

- ✅ Một POM project hoàn chỉnh trên GitHub (LoginPage + 1-2 page khác)
- ✅ Một workflow CI/CD `.yml` chạy được
- ✅ Một test có Allure report đẹp, có screenshot khi fail
- ✅ Một test với cookie injection / API login

Câu hỏi hay được hỏi ngẫu nhiên trong live coding:

- "Viết code chờ một element có text 'Loaded'" → CustomConditions
- "Mở 2 tab, switch qua lại" → Window handle pattern
- "Test một form có 10 field" → POM + builder pattern
- "Test một bảng có pagination" → Custom condition + for loop
- "Mock một API response" → CDP Fetch.fulfillRequest

Câu chốt cuối phỏng vấn:

> *"Em không phải là người viết test giỏi nhất. Em là người luôn hỏi: test này có cần thiết không, có chạy nhanh không, có tin được không. Ba câu hỏi đó giúp em luôn refactor được suite tốt hơn theo thời gian."*