Exercise 5 :

Cấu trúc dự án

Ảnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

Code của class basepage

package pages;  
  
import org.openqa.selenium.\*;  
import org.openqa.selenium.support.ui.\*;  
  
import java.time.Duration;  
  
public class BasePage {  
 protected WebDriver driver;  
 protected WebDriverWait wait;  
  
 public BasePage(WebDriver driver) {  
 this.driver = driver;  
 this.wait = new WebDriverWait(driver, Duration.*ofSeconds*(10));  
 }  
  
 // Click using WebElement (with scroll)  
 public void click(WebElement element) {  
 scrollIntoView(element);  
 try {  
 element.click();  
 } catch (ElementClickInterceptedException e) {  
 jsClick(element); // fallback  
 }  
 }  
  
 // Click using By locator (with wait and scroll)  
 protected void click(By locator) {  
 WebElement element = waitForVisibility(locator);  
 scrollIntoView(element);  
 try {  
 element.click();  
 } catch (ElementClickInterceptedException e) {  
 jsClick(element); // fallback  
 }  
 }  
  
 // JavaScript click  
 protected void jsClick(WebElement element) {  
 ((JavascriptExecutor) driver).executeScript("arguments[0].click();", element);  
 }  
  
 // JavaScript click with By  
 protected void jsClick(By locator) {  
 WebElement element = waitForVisibility(locator);  
 ((JavascriptExecutor) driver).executeScript("arguments[0].click();", element);  
 }  
  
 // Scroll into view  
 protected void scrollIntoView(WebElement element) {  
 ((JavascriptExecutor) driver).executeScript("arguments[0].scrollIntoView(true);", element);  
 }  
  
 // Wait for visibility  
 protected WebElement waitForVisibility(By locator) {  
 return wait.until(ExpectedConditions.*visibilityOfElementLocated*(locator));  
 }  
  
 // Wait for clickability  
 protected WebElement waitForClickability(By locator) {  
 return wait.until(ExpectedConditions.*elementToBeClickable*(locator));  
 }  
  
 // Send keys safely  
 protected void type(By locator, String text) {  
 WebElement element = waitForVisibility(locator);  
 element.clear();  
 element.sendKeys(text);  
 }  
  
 // Get text safely  
 protected String getText(By locator) {  
 return waitForVisibility(locator).getText();  
 }  
  
 // Navigate to a URL  
 public void navigateTo(String url) {  
 driver.get(url);  
 }  
  
 // Check if element is visible  
 protected boolean isElementVisible(By locator) {  
 try {  
 return waitForVisibility(locator).isDisplayed();  
 } catch (TimeoutException e) {  
 return false;  
 }  
 }  
}

code của loginpage:

package pages;  
  
import org.openqa.selenium.By;  
import org.openqa.selenium.WebDriver;  
  
public class LoginPage extends BasePage {  
  
 public LoginPage(WebDriver driver) {  
 super(driver);  
 }  
  
 // Locators  
 private By usernameField = By.*id*("username");  
 private By passwordField = By.*id*("password");  
 private By loginButton = By.*cssSelector*("button[type='submit']");  
 private By successMsg = By.*cssSelector*(".flash.success");  
 private By errorMsg = By.*cssSelector*(".flash.error");  
  
 // Actions  
 public void navigate() {  
 navigateTo("https://the-internet.herokuapp.com/login");  
 }  
  
 public void login(String username, String password) {  
 type(usernameField, username);  
 type(passwordField, password);  
 click(loginButton);  
 }  
  
 public By getSuccessLocator() {  
 return successMsg;  
 }  
  
 public By getErrorLocator() {  
 return errorMsg;  
 }  
  
 public String getMessageText(By locator) {  
 return getText(locator);  
 }  
}

code của registrasionpage:

package pages;  
  
import org.openqa.selenium.By;  
import org.openqa.selenium.JavascriptExecutor;  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.WebElement;  
  
import java.util.List;  
  
import static java.lang.Thread.*sleep*;  
  
public class RegistrationPage extends BasePage {  
  
 public RegistrationPage(WebDriver driver) {  
 super(driver);  
 }  
  
 private By firstNameField = By.*id*("firstName");  
 private By lastNameField = By.*id*("lastName");  
 private By emailField = By.*id*("userEmail");  
 private By genderRadio = By.*xpath*("//label[text()='Male']");  
 private By mobileField = By.*id*("userNumber");  
 private By submitButton = By.*id*("submit");  
 private By successModal = By.*id*("example-modal-sizes-title-lg");  
  
 public void navigate() {  
 navigateTo("https://demoqa.com/automation-practice-form");  
 }  
  
// public void fillForm(String firstName, String lastName, String email, String mobile) {  
// removeAds(); // ✅ Thêm dòng này để xóa iframe quảng cáo  
//  
// type(firstNameField, firstName);  
// type(lastNameField, lastName);  
// type(emailField, email);  
// click(genderRadio);  
// type(mobileField, mobile);  
// }  
public void fillForm(String firstName, String lastName, String email, String mobile) {  
 type(firstNameField, firstName);  
 sleep(500);  
 type(lastNameField, lastName);  
 sleep(500);  
 type(emailField, email);  
 sleep(500);  
  
 // Wait + JS Click for radio button  
 waitForClickability(genderRadio);  
 jsClick(genderRadio); // ✅ this avoids intercept issue  
 sleep(500);  
 type(mobileField, mobile);  
 sleep(500);  
}  
  
 private void sleep(long millis) {  
 try {  
 Thread.*sleep*(millis);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void submitForm() {  
 WebElement submitBtn = driver.findElement(By.*id*("submit"));  
 click(submitBtn);  
 }  
  
 public By getSuccessLocator() {  
 return successModal;  
 }  
  
 // ✅ Hàm xóa iframe quảng cáo để tránh lỗi click bị chặn  
 private void removeAds() {  
 List<WebElement> iframes = driver.findElements(By.*tagName*("iframe"));  
 for (WebElement iframe : iframes) {  
 ((JavascriptExecutor) driver).executeScript("arguments[0].remove();", iframe);  
 }  
 }  
  
}

code của driverfacroty : package utils;  
  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.chrome.ChromeDriver;  
import org.openqa.selenium.chrome.ChromeOptions;  
  
public class DriverFactory {  
 private static WebDriver *driver*;  
  
 public static WebDriver getDriver() {  
 if (*driver* == null) {  
 ChromeOptions options = new ChromeOptions();  
 options.addArguments("--remote-allow-origins=\*");  
 options.addArguments("--disable-blink-features=AutomationControlled");  
 options.addArguments("--start-maximized");  
  
 *driver* = new ChromeDriver(options);  
 }  
 return *driver*;  
 }  
  
 public static void quitDriver() {  
 if (*driver* != null) {  
 *driver*.quit();  
 *driver* = null;  
 }  
 }  
}

code của base test :

package tests;  
  
import org.junit.jupiter.api.AfterAll;  
import org.junit.jupiter.api.BeforeAll;  
import org.openqa.selenium.WebDriver;  
import utils.DriverFactory;  
  
public abstract class BaseTest {  
 protected static WebDriver *driver*;  
  
 @BeforeAll  
 public static void setUpBase() {  
 *driver* = DriverFactory.*getDriver*();  
// driver = DriverFactory.createDriver();  
 *driver*.manage().window().maximize();  
 }  
  
 @AfterAll  
 public static void tearDownBase() {  
 if (*driver* != null) {  
 *driver*.quit();  
 }  
 }  
}

code của login test : package tests;  
  
import org.junit.jupiter.api.\*;  
import org.junit.jupiter.params.ParameterizedTest;  
import org.junit.jupiter.params.provider.CsvFileSource;  
import org.junit.jupiter.params.provider.CsvSource;  
import org.openqa.selenium.\*;  
import org.openqa.selenium.support.ui.\*;  
import pages.LoginPage;  
  
import java.time.Duration;  
  
import static org.junit.jupiter.api.Assertions.*assertTrue*;  
  
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)  
@DisplayName("Login Tests using Page Object Model")  
public class LoginTest extends BaseTest{  
 //static WebDriver driver;  
 static WebDriverWait *wait*;  
 static LoginPage *loginPage*;  
  
 @BeforeAll  
 static void initPage() {  
 *loginPage* = new LoginPage(*driver*);  
 *wait* = new WebDriverWait(*driver*, Duration.*ofSeconds*(10));  
 }  
  
 @Test  
 @Order(1)  
 @DisplayName("Should login successfully with valid credentials")  
 void testLoginSuccess() {  
 *loginPage*.navigate();  
 *loginPage*.login("tomsmith", "SuperSecretPassword!");  
 WebElement success = *wait*.until(ExpectedConditions.*visibilityOfElementLocated*(*loginPage*.getSuccessLocator()));  
 *assertTrue*(success.getText().contains("You logged into a secure area!"));  
 }  
  
 @Test  
 @Order(2)  
 @DisplayName("Should show error for invalid credentials")  
 void testLoginFail() {  
 *loginPage*.navigate();  
 *loginPage*.login("wronguser", "wrongpassword");  
 WebElement error = *wait*.until(ExpectedConditions.*visibilityOfElementLocated*(*loginPage*.getErrorLocator()));  
 *assertTrue*(error.getText().toLowerCase().contains("invalid"));  
 }  
  
 @ParameterizedTest(name = "CSV Inline: {0} / {1}")  
 @Order(3)  
 @CsvSource({  
 "tomsmith, SuperSecretPassword!, success",  
 "wronguser, SuperSecretPassword!, error",  
 "tomsmith, wrongpassword, error",  
 "'', '', error"  
 })  
 void testLoginCsvInline(String username, String password, String expected) {  
 *loginPage*.navigate();  
 username = (username == null) ? "" : username.trim();  
 password = (password == null) ? "" : password.trim();  
  
 *loginPage*.login(username, password);  
 By resultLocator = expected.equals("success") ? *loginPage*.getSuccessLocator() : *loginPage*.getErrorLocator();  
 WebElement result = *wait*.until(ExpectedConditions.*visibilityOfElementLocated*(resultLocator));  
  
 if (expected.equals("success")) {  
 *assertTrue*(result.getText().contains("You logged into a secure area!"));  
 } else {  
 *assertTrue*(result.getText().toLowerCase().contains("invalid"));  
 }  
 }  
  
 @ParameterizedTest(name = "CSV File: {0} / {1}")  
 @Order(4)  
 @CsvFileSource(resources = "/login-data.csv", numLinesToSkip = 1)  
 void testLoginFromCSV(String username, String password, String expected) {  
 *loginPage*.navigate();  
 username = (username == null) ? "" : username.trim();  
 password = (password == null) ? "" : password.trim();  
  
 *loginPage*.login(username, password);  
 By resultLocator = expected.equals("success") ? *loginPage*.getSuccessLocator() : *loginPage*.getErrorLocator();  
 WebElement result = *wait*.until(ExpectedConditions.*visibilityOfElementLocated*(resultLocator));  
  
 if (expected.equals("success")) {  
 *assertTrue*(result.getText().contains("You logged into a secure area!"));  
 } else {  
 *assertTrue*(result.getText().toLowerCase().contains("invalid"));  
 }  
 }  
  
 @AfterAll  
 static void tearDown() {  
 *driver*.quit();  
 }  
}

code của registrationtest:

package tests;  
  
import org.junit.jupiter.api.\*;  
import org.openqa.selenium.WebElement;  
import org.openqa.selenium.support.ui.ExpectedConditions;  
import org.openqa.selenium.support.ui.WebDriverWait;  
import pages.RegistrationPage;  
  
import java.time.Duration;  
  
import static org.junit.jupiter.api.Assertions.*assertTrue*;  
  
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)  
@DisplayName("Registration Form Tests")  
public class RegistrationTest extends BaseTest {  
  
 static RegistrationPage *regPage*;  
 static WebDriverWait *wait*;  
  
 @BeforeAll  
 static void setup() {  
 *regPage* = new RegistrationPage(*driver*);  
 *wait* = new WebDriverWait(*driver*, Duration.*ofSeconds*(10));  
 }  
  
 @Test  
 @Order(1)  
 @DisplayName("Should register successfully")  
 void testValidRegistration() {  
 *regPage*.navigate();  
 *regPage*.fillForm("Khai", "Nguyen", "khai@test.com", "0987654321");  
 *regPage*.submitForm();  
  
 WebElement modal = *wait*.until(ExpectedConditions.*visibilityOfElementLocated*(*regPage*.getSuccessLocator()));  
 *assertTrue*(modal.getText().contains("Thanks for submitting the form"));  
 }  
}

file data:

username,password,expectedResult  
tomsmith,SuperSecretPassword!,success  
wronguser,SuperSecretPassword!,error  
tomsmith,wrongpassword,error  
"","",error

**1. RegistrationTest.java**

**Mục đích:**

Kiểm tra chức năng đăng ký tài khoản trên trang web.

**Quy trình test:**

* **setup()**: Khởi tạo trang đăng ký (RegistrationPage) và WebDriverWait.
* **testValidRegistration()**:

1. Điều hướng đến trang đăng ký.
2. Điền form với thông tin hợp lệ: tên, họ, email, số điện thoại.
3. Gửi form.
4. Đợi modal xác nhận hiện ra.
5. Kiểm tra nội dung modal có chứa "Thanks for submitting the form".

**Kết quả khi chạy:**

* Nếu form đăng ký hoạt động đúng, test sẽ **PASS** (thành công).
* Nếu có lỗi (form không gửi được, modal không hiện, hoặc nội dung không đúng), test sẽ **FAIL**.
* Ảnh có chứa văn bản, phần mềm, Phần mềm đa phương tiện, Phần mềm đồ họa

  Mô tả được tạo tự động

**Ý nghĩa:**

* Đảm bảo chức năng đăng ký hoạt động đúng với dữ liệu hợp lệ.
* Nếu test fail, có thể do bug ở UI, backend, hoặc locator không đúng.

**2. LoginTest.java**

**Mục đích:**

Kiểm tra chức năng đăng nhập với nhiều trường hợp khác nhau (thành công, thất bại, dữ liệu từ file CSV).

**Quy trình test:**

* **initPage()**: Khởi tạo trang đăng nhập (LoginPage) và WebDriverWait.
* **testLoginSuccess()**:

1. Điều hướng đến trang login.
2. Đăng nhập với tài khoản hợp lệ.
3. Đợi thông báo thành công hiện ra.
4. Kiểm tra nội dung thông báo.

* **testLoginFail()**:

1. Đăng nhập với tài khoản sai.
2. Đợi thông báo lỗi hiện ra.
3. Kiểm tra nội dung thông báo lỗi.

* **testLoginCsvInline()**:
* Chạy nhiều trường hợp đăng nhập với dữ liệu inline (trong code).
* Kiểm tra kết quả mong đợi (thành công hoặc lỗi).
* **testLoginFromCSV()**:
* Chạy nhiều trường hợp đăng nhập với dữ liệu từ file CSV (login-data.csv).
* Kiểm tra kết quả mong đợi.
* **tearDown()**: Đóng trình duyệt sau khi chạy xong tất cả test.

**Kết quả khi chạy:**

* Nếu các chức năng login hoạt động đúng (thành công với tài khoản hợp lệ, báo lỗi với tài khoản sai), các test sẽ **PASS**.
* Nếu có lỗi (login sai vẫn vào được, hoặc login đúng mà báo lỗi), test sẽ **FAIL**.
* Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

  Mô tả được tạo tự động

**Ý nghĩa:**

* Đảm bảo chức năng đăng nhập hoạt động đúng với nhiều trường hợp.
* Kiểm tra được cả UI, logic xử lý, và thông báo trả về cho người dùng.

Exercise 6 :

Test chức năng login và tạo mới user bên trang admin :

Cấu trúc dự án : Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

Code của : BasePage

package pages;  
  
import org.openqa.selenium.\*;  
import org.openqa.selenium.support.ui.\*;  
  
import java.time.Duration;  
  
public class BasePage {  
 protected WebDriver driver;  
 protected WebDriverWait wait;  
  
 public BasePage(WebDriver driver) {  
 this.driver = driver;  
 this.wait = new WebDriverWait(driver, Duration.*ofSeconds*(10));  
 }  
  
 // Click using WebElement (with scroll)  
 public void click(WebElement element) {  
 scrollIntoView(element);  
 try {  
 element.click();  
 } catch (ElementClickInterceptedException e) {  
 jsClick(element); // fallback  
 }  
 }  
  
 // Click using By locator (with wait and scroll)  
 protected void click(By locator) {  
 WebElement element = waitForVisibility(locator);  
 scrollIntoView(element);  
 try {  
 element.click();  
 } catch (ElementClickInterceptedException e) {  
 jsClick(element); // fallback  
 }  
 }  
  
 // JavaScript click  
 protected void jsClick(WebElement element) {  
 ((JavascriptExecutor) driver).executeScript("arguments[0].click();", element);  
 }  
  
 // JavaScript click with By  
 protected void jsClick(By locator) {  
 WebElement element = waitForVisibility(locator);  
 ((JavascriptExecutor) driver).executeScript("arguments[0].click();", element);  
 }  
  
 // Scroll into view  
 protected void scrollIntoView(WebElement element) {  
 ((JavascriptExecutor) driver).executeScript("arguments[0].scrollIntoView(true);", element);  
 }  
  
 // Wait for visibility  
 protected WebElement waitForVisibility(By locator) {  
 return wait.until(ExpectedConditions.*visibilityOfElementLocated*(locator));  
 }  
  
 // Wait for clickability  
 protected WebElement waitForClickability(By locator) {  
 return wait.until(ExpectedConditions.*elementToBeClickable*(locator));  
 }  
  
 // Send keys safely  
 protected void type(By locator, String text) {  
 WebElement element = waitForVisibility(locator);  
 element.clear();  
 element.sendKeys(text);  
 }  
  
 // Get text safely  
 protected String getText(By locator) {  
 return waitForVisibility(locator).getText();  
 }  
  
 // Navigate to a URL  
 public void navigateTo(String url) {  
 driver.get(url);  
 }  
  
 // Check if element is visible  
 protected boolean isElementVisible(By locator) {  
 try {  
 return waitForVisibility(locator).isDisplayed();  
 } catch (TimeoutException e) {  
 return false;  
 }  
 }  
}

code cuả LoginPage :

package pages;  
  
import org.openqa.selenium.By;  
import org.openqa.selenium.WebDriver;  
  
public class LoginPage extends BasePage {  
  
 public LoginPage(WebDriver driver) {  
 super(driver);  
 }  
  
 // Locators theo HTML thực tế  
 private By emailField = By.*name*("Login.Email");  
 private By passwordField = By.*name*("Login.Password");  
 private By loginButton = By.*xpath*("//button[contains(text(),'Đăng Nhập')]");  
 private By errorMsg = By.*cssSelector*(".text-danger");  
 // Nếu có thông báo thành công, bổ sung locator phù hợp  
 // private By successMsg = By.cssSelector(".alert-success");  
  
 // Actions  
 public void navigate() {  
 navigateTo("https://localhost:7231/Login/Login");  
 }  
  
 public void login(String email, String password) {  
 type(emailField, email);  
 type(passwordField, password);  
 click(loginButton);  
 }  
  
 public By getErrorLocator() {  
 return errorMsg;  
 }  
  
 // Nếu có thông báo thành công, bổ sung hàm này:  
 // public By getSuccessLocator() { return successMsg; }  
  
 public String getMessageText(By locator) {  
 return getText(locator);  
 }  
}

code của RegistrationPage :

package pages;  
  
import org.openqa.selenium.By;  
import org.openqa.selenium.WebDriver;  
  
public class RegistrationPage extends BasePage {  
  
 public RegistrationPage(WebDriver driver) {  
 super(driver);  
 }  
  
 // Locators cho hệ thống thực tế  
 private By nameField = By.*name*("Register.Name");  
 private By emailField = By.*name*("Register.Email");  
 private By passwordField = By.*name*("Register.Password");  
 private By registerButton = By.*xpath*("//button[contains(text(),'Đăng Ký')]");  
 private By errorMsg = By.*cssSelector*(".text-danger");  
 // Nếu có thông báo thành công, bổ sung locator phù hợp  
 // private By successMsg = By.cssSelector(".alert-success");  
  
 public void navigate() {  
 navigateTo("https://localhost:7231/Login/Login");  
 }  
  
 public void fillForm(String name, String email, String password) {  
 type(nameField, name);  
 type(emailField, email);  
 type(passwordField, password);  
 }  
  
 public void submitForm() {  
 click(registerButton);  
 }  
  
 public By getErrorLocator() {  
 return errorMsg;  
 }  
  
 // Nếu có thông báo thành công, bổ sung hàm này:  
 // public By getSuccessLocator() { return successMsg; }  
}

code của LoginTest:

package tests;  
  
import org.junit.jupiter.api.\*;  
import org.junit.jupiter.params.ParameterizedTest;  
import org.junit.jupiter.params.provider.CsvFileSource;  
import org.openqa.selenium.\*;  
import org.openqa.selenium.support.ui.\*;  
import pages.LoginPage;  
  
import java.time.Duration;  
  
import static org.junit.jupiter.api.Assertions.*assertTrue*;  
  
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)  
@DisplayName("Login Tests for pet\_spa\_system1")  
public class LoginTest extends BaseTest {  
 static WebDriverWait *wait*;  
 static LoginPage *loginPage*;  
  
 @BeforeAll  
 static void initPage() {  
 *loginPage* = new LoginPage(*driver*);  
 *wait* = new WebDriverWait(*driver*, Duration.*ofSeconds*(10));  
 }  
  
 @ParameterizedTest(name = "Login: {0} / {1} => {2}")  
 @CsvFileSource(resources = "/login-data.csv", numLinesToSkip = 1)  
 @Order(1)  
 void testLoginFromCSV(String email, String password, String expected) {  
 *loginPage*.navigate();  
 email = (email == null) ? "" : email.trim();  
 password = (password == null) ? "" : password.trim();  
  
 *loginPage*.login(email, password);  
  
 if ("success".equals(expected)) {  
 *wait*.until(ExpectedConditions.*not*(ExpectedConditions.*urlToBe*("https://localhost:7231/Login/Login")));  
 *assertTrue*(!*driver*.getCurrentUrl().contains("/Login/Login"));  
 } else {  
 try {  
 WebElement error = *wait*.until(ExpectedConditions.*visibilityOfElementLocated*(*loginPage*.getErrorLocator()));  
 *assertTrue*(error.getText().length() > 0);  
 } catch (TimeoutException e) {  
 Assertions.*fail*("Không tìm thấy thông báo lỗi khi đăng nhập sai (email: '" + email + "', password: '" + password + "')");  
 }  
 }  
 }  
}

File data:

email,password,expectedResult  
khailamtest@gmail.com,123456789,success  
khailamtest@gmail.com,wrongpass,error  
wrongemail@gmail.com,123456789,error  
,123456789,error  
khailamtest@gmail.com,,error  
,,

**1. Mục đích của LoginTest.java**

* **Kiểm thử tự động chức năng đăng nhập** của hệ thống pet\_spa\_system1 với nhiều bộ dữ liệu khác nhau (đọc từ file CSV).
* Đảm bảo hệ thống xử lý đúng các trường hợp: đăng nhập thành công, đăng nhập thất bại (sai mật khẩu, thiếu trường, ...).

**2. Cách hoạt động của test**

* **Test sử dụng annotation @ParameterizedTest**: mỗi dòng trong file login-data.csv sẽ được chạy như một test case riêng biệt.
* **Các bước kiểm thử cho mỗi bộ dữ liệu:**

1. **Điền email và password** vào form đăng nhập.
2. **Nhấn nút Đăng Nhập**.
3. **Kiểm tra kết quả:**

* Nếu expected là "success":
* Test sẽ **chờ URL thay đổi** (không còn là /Login/Login), tức là đăng nhập thành công và đã chuyển trang.
* **Kết quả mong đợi:** Test pass nếu URL đã thay đổi.
* Nếu expected là "error":
* Test sẽ **chờ xuất hiện thông báo lỗi** (element có class .text-danger).
* **Kết quả mong đợi:** Test pass nếu tìm thấy thông báo lỗi.
* Nếu không tìm thấy thông báo lỗi trong 10 giây, test sẽ fail với thông báo rõ ràng.

1. **Ý nghĩa kết quả khi chạy test**

**Ảnh có chứa văn bản, Phần mềm đa phương tiện, phần mềm, Phần mềm đồ họa

Mô tả được tạo tự động**

* **Test pass (màu xanh):**
* Hệ thống hoạt động đúng với từng trường hợp kiểm thử.
* Đăng nhập thành công khi nhập đúng tài khoản.
* Hiển thị thông báo lỗi khi nhập sai tài khoản, mật khẩu, hoặc thiếu trường.
* **Test fail (màu đỏ):**
* Có thể do một trong các nguyên nhân:
* Đăng nhập thành công nhưng không chuyển trang (URL vẫn là /Login/Login).
* Đăng nhập sai nhưng **không hiển thị thông báo lỗi** (không có element .text-danger).
* Lỗi logic hoặc bug phía backend/frontend (ví dụ: không validate trường rỗng, không trả về lỗi khi sai mật khẩu).
* Lỗi timeout do trang không phản hồi hoặc chậm.
* **Thông báo lỗi cụ thể:**
* "Không tìm thấy thông báo lỗi khi đăng nhập sai (email: '...', password: '...')":

→ Hệ thống chưa hiển thị thông báo lỗi khi đăng nhập sai hoặc thiếu trường. Đây là bug cần sửa phía backend/frontend.

**Case 1:** Đúng tài khoản → test pass nếu chuyển trang.

**Case 2, 3, 4, 5, 6:** Sai tài khoản, thiếu trường → test pass nếu có thông báo lỗi; test fail nếu không có thông báo lỗi.

+ **Test chức năng add new user**

Code của adminuserpage

package pages;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

public class AdminUserPage extends BasePage {

    public AdminUserPage(WebDriver driver) {

        super(driver);

    }

    // Locators

    private By addUserBtn = By.xpath("//button[contains(.,'Add New User')]");

    private By usernameField = By.id("edit-username");

    private By emailField = By.id("edit-email");

    private By fullNameField = By.id("edit-fullname");

    private By phoneField = By.id("edit-phone");

    private By addressField = By.id("edit-address");

    private By roleSelect = By.id("edit-role");

    private By saveBtn = By.xpath("//button[@type='submit' and contains(.,'Save')]");

    private By modal = By.id("editUserModal");

    private By errorMsg = By.cssSelector(".text-danger, .alert-danger");

    public void navigate() {

        navigateTo("https://localhost:7231/Admin/List\_Customer");

    }

    public void openAddUserModal() {

        click(addUserBtn);

        waitForVisibility(modal);

    }

    public void fillUserForm(String username, String email, String fullName, String phone, String address, String roleValue) {

        type(usernameField, username);

        type(emailField, email);

        type(fullNameField, fullName);

        type(phoneField, phone);

        type(addressField, address);

        selectByValue(roleSelect, roleValue); // "1"=Admin, "2"=Customer, "3"=Employee

    }

    public void submitUserForm() {

        click(saveBtn);

    }

    public By getErrorLocator() {

        return errorMsg;

    }

}

Code của adminuserTest

package tests;

import org.junit.jupiter.api.\*;

import org.junit.jupiter.params.ParameterizedTest;

import org.junit.jupiter.params.provider.CsvFileSource;

import org.openqa.selenium.\*;

import org.openqa.selenium.support.ui.\*;

import pages.AdminUserPage;

import java.time.Duration;

import static org.junit.jupiter.api.Assertions.assertTrue;

@TestMethodOrder(MethodOrderer.OrderAnnotation.class)

@DisplayName("Admin Add User Tests")

public class AdminUserTest extends BaseTest {

    static WebDriverWait wait;

    static AdminUserPage adminUserPage;

    @BeforeAll

    static void setup() {

        adminUserPage = new AdminUserPage(driver);

        wait = new WebDriverWait(driver, Duration.ofSeconds(10));

    }

    @ParameterizedTest(name = "Add User: {0} / {1} / {2} => {6}")

    @CsvFileSource(resources = "/add-user-data.csv", numLinesToSkip = 1)

    @Order(1)

    void testAddUser(String username, String email, String fullName, String phone, String address, String role, String expected) {

        adminUserPage.navigate();

        adminUserPage.openAddUserModal();

        adminUserPage.fillUserForm(username, email, fullName, phone, address, role);

        adminUserPage.submitUserForm();

        if ("success".equals(expected)) {

            // TODO: Kiểm tra user mới xuất hiện trong bảng (có thể kiểm tra bằng email hoặc username)

            // Ví dụ: wait.until(ExpectedConditions.textToBePresentInElementLocated(By.id("userTableBody"), email));

        } else {

            try {

                WebElement error = wait.until(ExpectedConditions.visibilityOfElementLocated(adminUserPage.getErrorLocator()));

                assertTrue(error.getText().length() > 0);

            } catch (TimeoutException e) {

                Assertions.fail("Không tìm thấy thông báo lỗi khi thêm user sai (username: '" + username + "', email: '" + email + "')");

            }

        }

    }

}

File data :

username,email,fullName,phone,address,role,expectedResult

testuser1,test1@example.com,Test User 1,0123456789,123 Main St,2,success

,test2@example.com,Test User 2,0123456789,123 Main St,2,error

testuser3,,Test User 3,0123456789,123 Main St,2,error

testuser4,test4@example.com,,0123456789,123 Main St,2,error

**1. Mục đích của AdminUserTest.java**

* **Kiểm thử tự động chức năng Thêm mới User** trên trang Admin /Admin/List\_Customer với nhiều bộ dữ liệu khác nhau (đọc từ file add-user-data.csv).
* Đảm bảo hệ thống xử lý đúng các trường hợp: thêm user thành công, thêm user thất bại (thiếu trường bắt buộc, ...).

**2. Cách hoạt động của test**

* **Test sử dụng @ParameterizedTest**: mỗi dòng trong file CSV là một test case riêng biệt.
* **Các bước kiểm thử cho mỗi bộ dữ liệu:**

1. **Truy cập trang quản lý user**.
2. **Mở modal Thêm User**.
3. **Điền thông tin user** (username, email, fullName, phone, address, role).
4. **Nhấn Save**.
5. **Kiểm tra kết quả:**

* Nếu expected là "success":
* (TODO) Kiểm tra user mới xuất hiện trong bảng (bạn có thể bổ sung kiểm tra này, ví dụ kiểm tra email xuất hiện trong bảng).
* Nếu expected là "error":
* Test sẽ **chờ xuất hiện thông báo lỗi** (element có class .text-danger hoặc .alert-danger).
* Nếu không tìm thấy thông báo lỗi trong 10 giây, test sẽ fail với thông báo rõ ràng.

1. **Ý nghĩa kết quả khi chạy test**

**-**

**Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, phần mềm

Mô tả được tạo tự động**

* **Test pass (màu xanh):**
* Hệ thống hoạt động đúng với từng trường hợp kiểm thử.
* Thêm user thành công khi nhập đủ thông tin hợp lệ.
* Hiển thị thông báo lỗi khi thiếu trường bắt buộc (username, email, fullName).
* **Test fail (màu đỏ):**
* Có thể do một trong các nguyên nhân:
* Thêm user thành công nhưng user không xuất hiện trong bảng (nếu bạn bổ sung kiểm tra này).
* Thêm user lỗi nhưng **không hiển thị thông báo lỗi** (không có element .text-danger hoặc .alert-danger).
* Lỗi logic hoặc bug phía backend/frontend (ví dụ: không validate trường rỗng, không trả về lỗi khi thiếu trường).
* Lỗi timeout do trang không phản hồi hoặc chậm.
* **Thông báo lỗi cụ thể:**
* "Không tìm thấy thông báo lỗi khi thêm user sai (username: '...', email: '...')":

→ Hệ thống chưa hiển thị thông báo lỗi khi thêm user sai hoặc thiếu trường. Đây là bug cần sửa phía backend/frontend.