Execise 5:

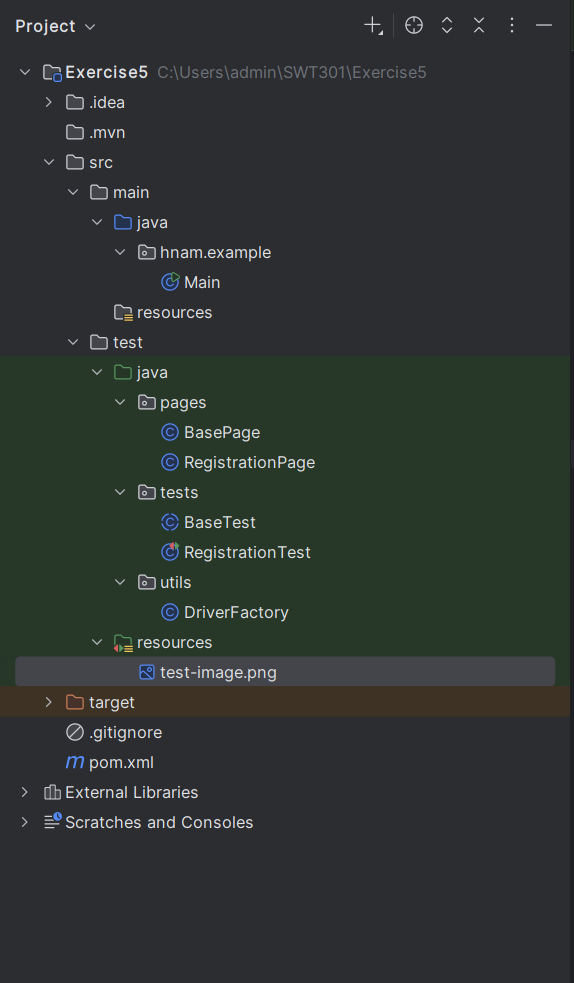
 Mục tiêu: Tự động kiểm thử việc điền và gửi form đăng ký người dùng.

 Công nghệ:

* Java
* Selenium WebDriver
* JUnit 5
* WebDriverManager (tự động quản lý driver)

 Kiểm thử thực hiện trên trình duyệt Chrome.

Cấu trúc dự án



Pom:

<dependencies>  
 <dependency>  
 <groupId>org.seleniumhq.selenium</groupId>  
 <artifactId>selenium-java</artifactId>  
 <version>4.21.0</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter-engine</artifactId>  
 <version>5.13.0-M3</version>  
 <scope>test</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter-params</artifactId>  
 <version>5.13.0-M3</version>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>io.github.bonigarcia</groupId>  
 <artifactId>webdrivermanager</artifactId>  
 <version>5.7.0</version>  
 </dependency>  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-simple</artifactId>  
 <version>2.0.13</version>  
 </dependency>  
  
</dependencies>

BasePage.java

package pages;  
  
import org.openqa.selenium.\*;  
import org.openqa.selenium.support.ui.ExpectedConditions;  
import org.openqa.selenium.support.ui.WebDriverWait;  
  
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));  
 }  
  
 public WebElement waitForVisibility(By locator) {  
 return wait.until(ExpectedConditions.*visibilityOfElementLocated*(locator));  
 }  
  
 public void click(By locator) {  
 WebElement element = waitForVisibility(locator);  
  
 try {  
 ((JavascriptExecutor) driver).executeScript("arguments[0].scrollIntoView({block: 'center'});", element);  
 element.click();  
 } catch (ElementClickInterceptedException e) {  
 ((JavascriptExecutor) driver).executeScript("arguments[0].click();", element);  
 }  
 }  
  
 public void type(By locator, String text) {  
 WebElement element = waitForVisibility(locator);  
 element.clear();  
 element.sendKeys(text);  
 }  
  
 public String getText(By locator) {  
 return waitForVisibility(locator).getText();  
 }  
  
 public void navigateTo(String url) {  
 driver.get(url);  
 }  
 public boolean isElementVisible(By locator) {  
 try {  
 return wait.until(ExpectedConditions.*visibilityOfElementLocated*(locator)).isDisplayed();  
 } catch (TimeoutException e) {  
 return false;  
 }  
 }  
}

RegistrationPage.java (class đại diện cho trang form đăng ký, theo mô hình Page Object Model.)

package pages;  
  
import org.openqa.selenium.By;  
import org.openqa.selenium.JavascriptExecutor;  
import org.openqa.selenium.Keys;  
import org.openqa.selenium.WebDriver;  
  
public class RegistrationPage extends BasePage {  
  
 public RegistrationPage(WebDriver driver) {  
 super(driver);  
 }  
  
 private final By firstName = By.*id*("firstName");  
 private final By lastName = By.*id*("lastName");  
 private final By email = By.*id*("userEmail");  
 private final By genderMale = By.*xpath*("//label[text()='Male']");  
 private final By mobile = By.*id*("userNumber");  
 private final By hobbiesSports = By.*xpath*("//label[text()='Sports']");  
 private final By uploadPicture = By.*id*("uploadPicture");  
 private final By currentAddress = By.*id*("currentAddress");  
 private final By stateDropdown = By.*id*("react-select-3-input");  
 private final By cityDropdown = By.*id*("react-select-4-input");  
 private final By submitButton = By.*id*("submit");  
  
 private final By confirmationModal = By.*id*("example-modal-sizes-title-lg");  
 private final By closePopupAd = By.*id*("close-fixedban");  
  
 public void navigate() {  
 navigateTo("https://demoqa.com/automation-practice-form");  
  
 // Thử đóng popup nếu có  
 try {  
 driver.switchTo().frame("google\_ads\_iframe\_/21849154601,22343295815/Ad.Plus-Anchor\_0");  
 ((JavascriptExecutor) driver).executeScript("arguments[0].style.display='none';", driver.findElement(By.*tagName*("iframe")));  
 driver.switchTo().defaultContent();  
 } catch (Exception ignored) {}  
  
 try {  
 click(closePopupAd);  
 } catch (Exception ignored) {}  
 }  
  
 public void fillForm(String fName, String lName, String emailAddr, String phone, String address, String filePath) {  
 type(firstName, fName);  
 type(lastName, lName);  
 type(email, emailAddr);  
 click(genderMale);  
 type(mobile, phone);  
 click(hobbiesSports);  
 type(uploadPicture, filePath);  
 type(currentAddress, address);  
 selectDropdown(stateDropdown, "NCR");  
 selectDropdown(cityDropdown, "Delhi");  
 }  
  
 public void submit() {  
 click(submitButton);  
 }  
  
 public boolean isSubmissionSuccessful() {  
 return isElementVisible(confirmationModal);  
 }  
  
 private void selectDropdown(By locator, String value) {  
 waitForVisibility(locator).sendKeys(value);  
 waitForVisibility(locator).sendKeys(Keys.*ENTER*);  
 }  
}

BaseTest.java

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.*createDriver*();  
 *driver*.manage().window().maximize();  
 }  
  
 @AfterAll  
 public static void tearDownBase() {  
 if (*driver* != null) {  
 *driver*.quit();  
 }  
 }  
}

 click(): click an toàn với scroll + fallback bằng JavaScript.

 type(): nhập dữ liệu vào input.

 getText(): lấy nội dung text từ phần tử.

 navigateTo(url): điều hướng tới URL.

 isElementVisible(): kiểm tra element có hiển thị không.

**Mục đích:** Tránh lặp lại code, giúp các Page cụ thể (RegistrationPage) sử dụng dễ dàng.

RegistrationTest.java

package tests;  
  
import org.junit.jupiter.api.\*;  
import org.openqa.selenium.WebDriver;  
import pages.RegistrationPage;  
  
import static org.junit.jupiter.api.Assertions.*assertTrue*;  
  
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)  
@DisplayName("Test Registration Form - demoqa.com")  
public class RegistrationTest extends BaseTest {  
 static RegistrationPage *formPage*;  
  
 @BeforeAll  
 static void init() {  
 *formPage* = new RegistrationPage(*driver*);  
 }  
  
 @Test  
 @Order(1)  
 @DisplayName("Should submit registration form successfully")  
 void testSubmitRegistrationForm() {  
 *formPage*.navigate();  
  
 *formPage*.fillForm(  
 "John",  
 "Doe",  
 "john.doe@example.com",  
 "0123456789",  
 "123 Sample Street",  
 System.*getProperty*("user.dir") + "/src/test/resources/test-image.png"  
 );  
  
 *formPage*.submit();  
  
 *assertTrue*(*formPage*.isSubmissionSuccessful(), "Form was not submitted successfully.");  
 }  
 @AfterAll  
 static void tearDown() {  
 try {  
 Thread.*sleep*(5000);   
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 *driver*.quit();  
 }  
}

 **navigateTo(url)**: Truy cập vào trang form.

 **fillForm()**: Nhập dữ liệu vào các trường của form.

 **submitForm()**: Nhấn nút Submit để gửi form.

 **assertTrue(...)**: Kiểm tra xem thông báo thành công hiển thị hay không.

@afterall để quan sát kết quả điền thông tin

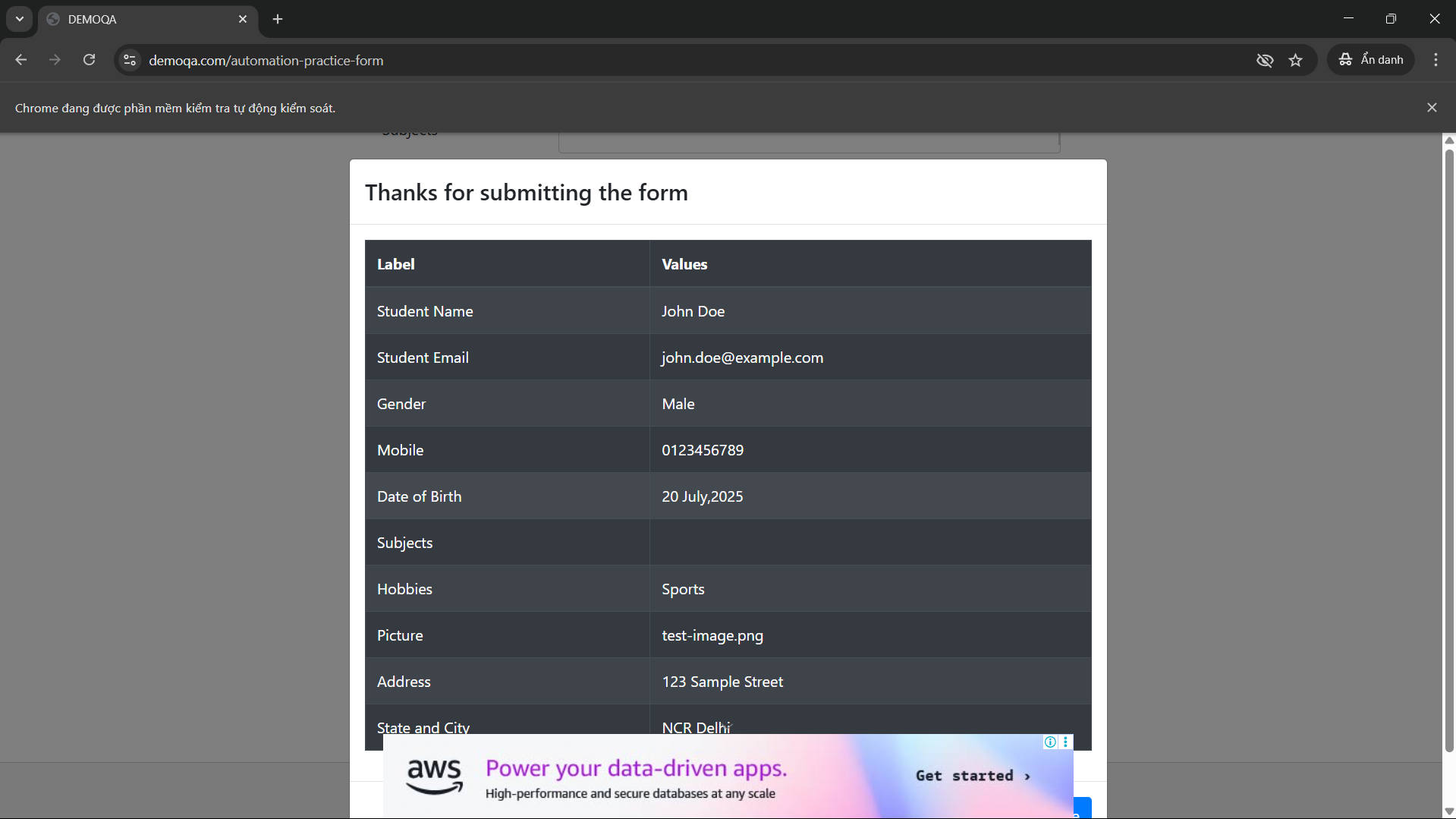
DriverFactory.java

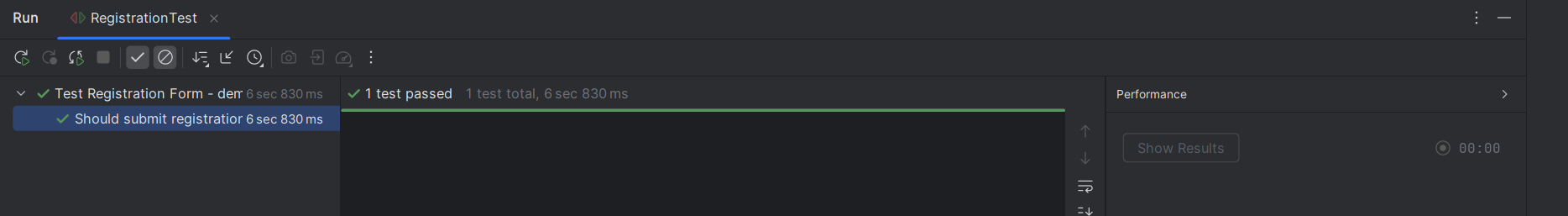
package utils;  
  
import io.github.bonigarcia.wdm.WebDriverManager;  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.chrome.ChromeDriver;  
import org.openqa.selenium.chrome.ChromeOptions;  
  
public class DriverFactory {  
 public static WebDriver createDriver() {  
 WebDriverManager.*chromedriver*().setup();  
  
 ChromeOptions options = new ChromeOptions();  
 options.addArguments("--incognito");  
 options.addArguments("--start-maximized");  
  
 return new ChromeDriver(options);  
 }  
}

**Quy trình kiểm thử tự động**

1. Trình duyệt Chrome được mở bằng WebDriverManager.
2. Truy cập trang form.
3. Nhập liệu tự động từng trường.
4. Upload ảnh.
5. Nhấn Submit.
6. Xác nhận form đã gửi thành công qua message.

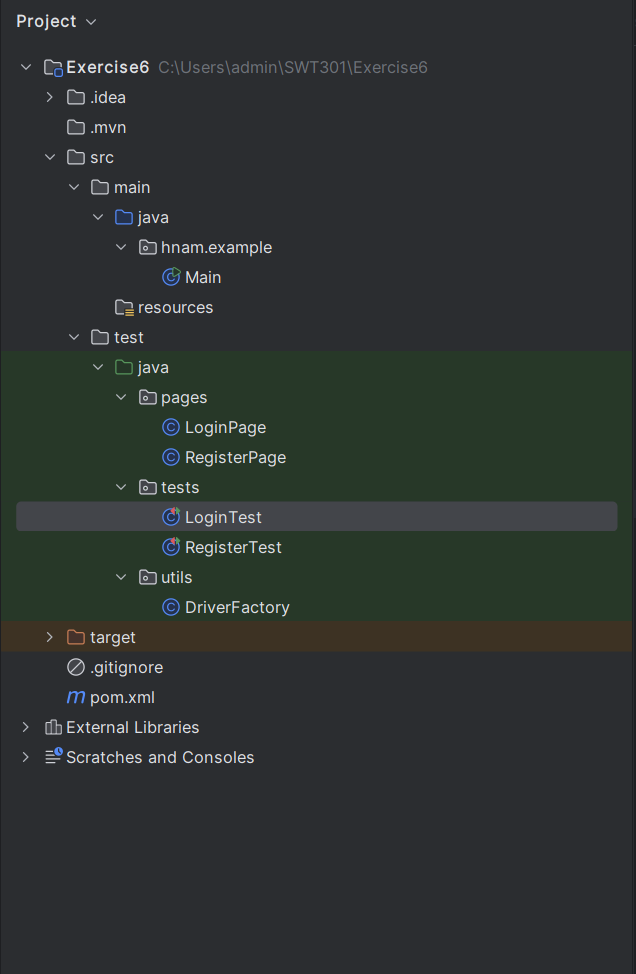
Chạy, thông tin tự động điền vào





Exercise 6 :

Cấu trúc dự án:



Pom:

<dependencies>  
 <dependency>  
 <groupId>org.seleniumhq.selenium</groupId>  
 <artifactId>selenium-java</artifactId>  
 <version>4.21.0</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter-engine</artifactId>  
 <version>5.13.0-M3</version>  
 <scope>test</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter-params</artifactId>  
 <version>5.13.0-M3</version>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>io.github.bonigarcia</groupId>  
 <artifactId>webdrivermanager</artifactId>  
 <version>5.7.0</version>  
 </dependency>  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-simple</artifactId>  
 <version>2.0.13</version>  
 </dependency>  
  
</dependencies>

DriverFactory

package utils;  
  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.edge.EdgeDriver;  
import org.openqa.selenium.edge.EdgeOptions;  
  
public class DriverFactory {  
 public static WebDriver createDriver() {  
  
 System.*setProperty*("webdriver.edge.driver", "C:\\Users\\admin\\Downloads\\edgedriver\_win64\\msedgedriver.exe");  
  
 EdgeOptions options = new EdgeOptions();  
 options.addArguments("--start-maximized");  
 options.addArguments("--disable-notifications");  
  
 return new EdgeDriver(options);  
 }  
}

* System.setProperty(...): thiết lập đường dẫn đến ChromeDriver.
* ChromeOptions: thêm tùy chọn khởi chạy trình duyệt.
* --start-maximized: mở trình duyệt ở chế độ toàn màn hình.
* --disable-notifications: tắt thông báo của trình duyệt.
* Trả về đối tượng ChromeDriver đã cấu hình sẵn.

LoginPage:

package pages;  
  
import org.openqa.selenium.By;  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.support.ui.ExpectedConditions;  
import org.openqa.selenium.support.ui.WebDriverWait;  
import org.openqa.selenium.WebElement;  
  
import java.time.Duration;  
  
public class LoginPage {  
 private WebDriver driver;  
  
 public LoginPage(WebDriver driver) {  
 this.driver = driver;  
 }  
  
 private By emailInput = By.*name*("email");  
 private By passwordInput = By.*name*("password");  
 private By loginButton = By.*cssSelector*("button[type='submit']");  
 private By alertBox = By.*className*("alert");  
  
 // Nút đăng nhập Google dựa trên nội dung văn bản  
 private By googleLoginButton = By.*xpath*("//span[contains(text(), 'Google') or contains(text(), 'Đăng nhập bằng Google')]");  
  
 public void navigate() {  
 driver.get("http://localhost:3000/auth/login");  
 }  
  
 public void login(String email, String password) {  
 driver.findElement(emailInput).sendKeys(email);  
 driver.findElement(passwordInput).sendKeys(password);  
 driver.findElement(loginButton).click();  
 }  
  
 public void clickGoogleLogin(WebDriverWait wait) {  
 wait.until(ExpectedConditions.*elementToBeClickable*(googleLoginButton)).click();  
  
 try {  
 Thread.*sleep*(10000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
  
 }  
  
 public By getAlertBox() {  
 return alertBox;  
 }  
}

Các By định vị các element như input email, password, button login, cảnh báo lỗi, nút login bằng Google.

 login(...): thực hiện nhập thông tin và click login.

 clickGoogleLogin(...): chờ nút Google khả dụng, sau đó click và tạm dừng 10s để người dùng tự đăng nhập Google.

LoginTest:

package tests;  
  
import org.junit.jupiter.api.\*;  
import org.openqa.selenium.\*;  
import org.openqa.selenium.support.ui.\*;  
import pages.LoginPage;  
import utils.DriverFactory;  
  
import java.time.Duration;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
public class LoginTest {  
 static WebDriver *driver*;  
 static LoginPage *loginPage*;  
 static WebDriverWait *wait*;  
  
 @BeforeAll  
 static void setup() {  
 *driver* = DriverFactory.*createDriver*();  
 *loginPage* = new LoginPage(*driver*);  
 *wait* = new WebDriverWait(*driver*, Duration.*ofSeconds*(15)); // tăng thời gian đợi  
 }  
  
 @Test  
 void testLoginFail() {  
 *loginPage*.navigate();  
 *loginPage*.login("wrong@email.com", "wrongpass");  
  
 WebElement alert = *wait*.until(ExpectedConditions  
 .*visibilityOfElementLocated*(*loginPage*.getAlertBox()));  
  
 *assertTrue*(alert.getText().toLowerCase().contains("mật khẩu")  
 || alert.getText().toLowerCase().contains("sai"));  
 }

@Test  
 void testLoginWithGoogle() {  
 *loginPage*.navigate();  
  
 *loginPage*.clickGoogleLogin(*wait*);  
  
 System.*out*.println("🔒 Vui lòng hoàn tất đăng nhập Google thủ công trong vòng 60 giây...");  
 new WebDriverWait(*driver*, Duration.*ofSeconds*(60))  
 .until(ExpectedConditions.*urlContains*("localhost:3000"));  
  
 *assertTrue*(*driver*.getCurrentUrl().contains("localhost:3000"));  
 }  
  
  
 @AfterAll  
 static void tearDown() {  
 if (*driver* != null) *driver*.quit();  
 }  
}

 Kiểm thử đăng nhập sai.

 Kiểm tra thông báo lỗi có xuất hiện hay không.

* Kiểm thử đăng nhập bằng Google: mở popup → người dùng tự đăng nhập → kiểm tra URL sau khi đăng nhập.

RegisterPage

package pages;  
  
import org.openqa.selenium.By;  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.support.ui.Select;  
  
public class RegisterPage {  
 private WebDriver driver;  
  
 public RegisterPage(WebDriver driver) {  
 this.driver = driver;  
 }  
  
 // OTP input step  
 private By emailInput = By.*name*("email");  
 private By submitButton = By.*cssSelector*("button[type='submit']");  
 private By toastMessage = By.*className*("Toastify\_\_toast");  
  
 // Registration form selectors  
 private By nameInput = By.*name*("name");  
 private By phoneInput = By.*name*("phone");  
 private By dobInput = By.*name*("date\_of\_birth");  
 private By genderSelect = By.*name*("gender");  
 private By passwordInput = By.*name*("password");  
 private By confirmPasswordInput = By.*name*("confirmPassword");  
 private By registerSubmitButton = By.*cssSelector*("button[type='submit']");  
  
 public void navigate() {  
 driver.get("http://localhost:3000/auth/user-register");  
 }  
  
 public void sendOtp(String email) {  
 driver.findElement(emailInput).sendKeys(email);  
 driver.findElement(submitButton).click();  
 }  
  
 public void fillRegistrationForm() {  
 driver.findElement(nameInput).sendKeys("ahihihi");  
 driver.findElement(phoneInput).sendKeys("0912345678");  
 driver.findElement(dobInput).sendKeys("2000-01-01");  
 new Select(driver.findElement(genderSelect)).selectByValue("male");  
 driver.findElement(passwordInput).sendKeys("Test@1234");  
 driver.findElement(confirmPasswordInput).sendKeys("Test@1234");  
 }  
  
 public void submitRegistrationForm() {  
 driver.findElement(registerSubmitButton).click();  
 }  
  
 public By getToastMessage() {  
 return toastMessage;  
 }  
}

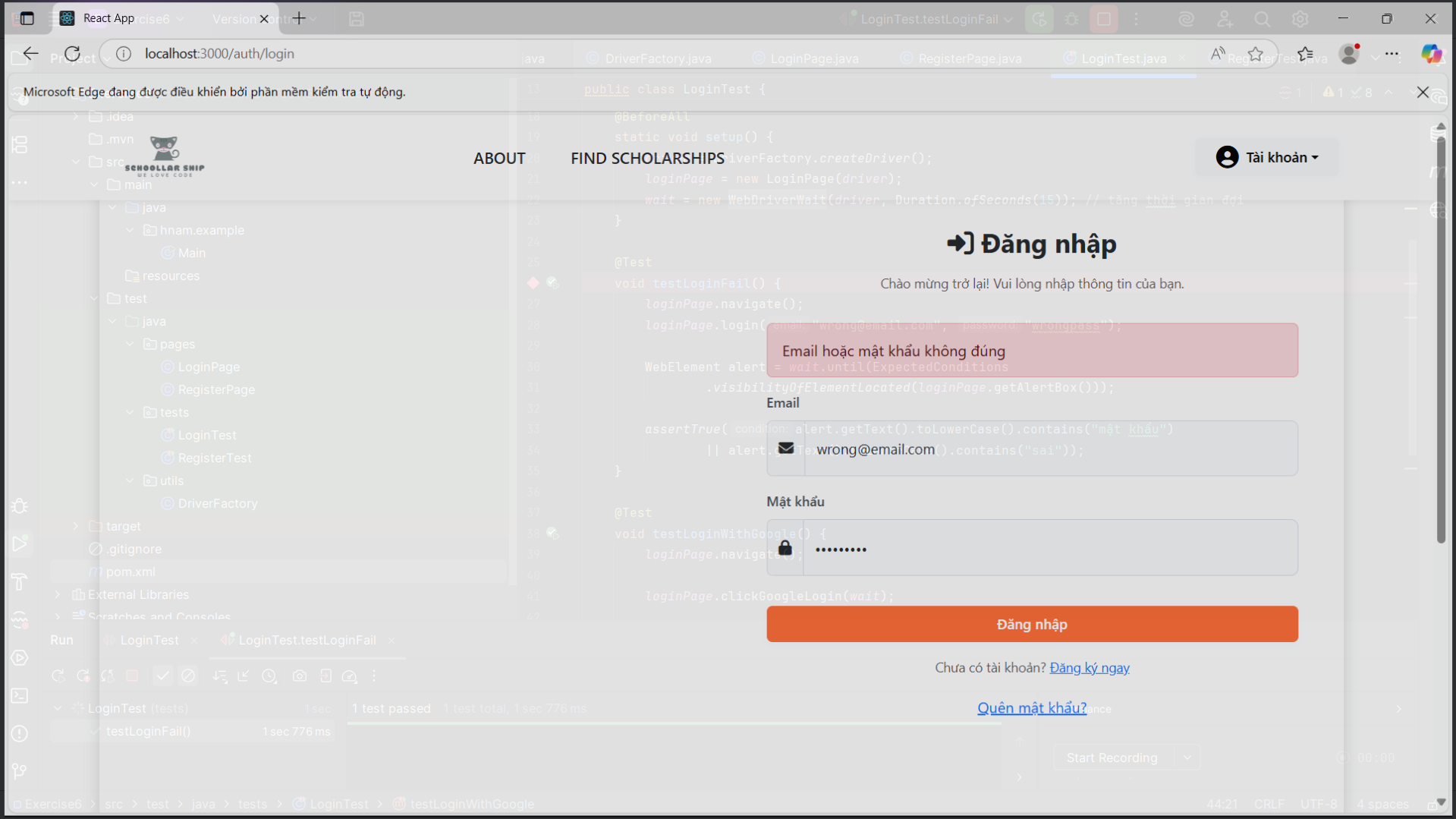
RegisterTest

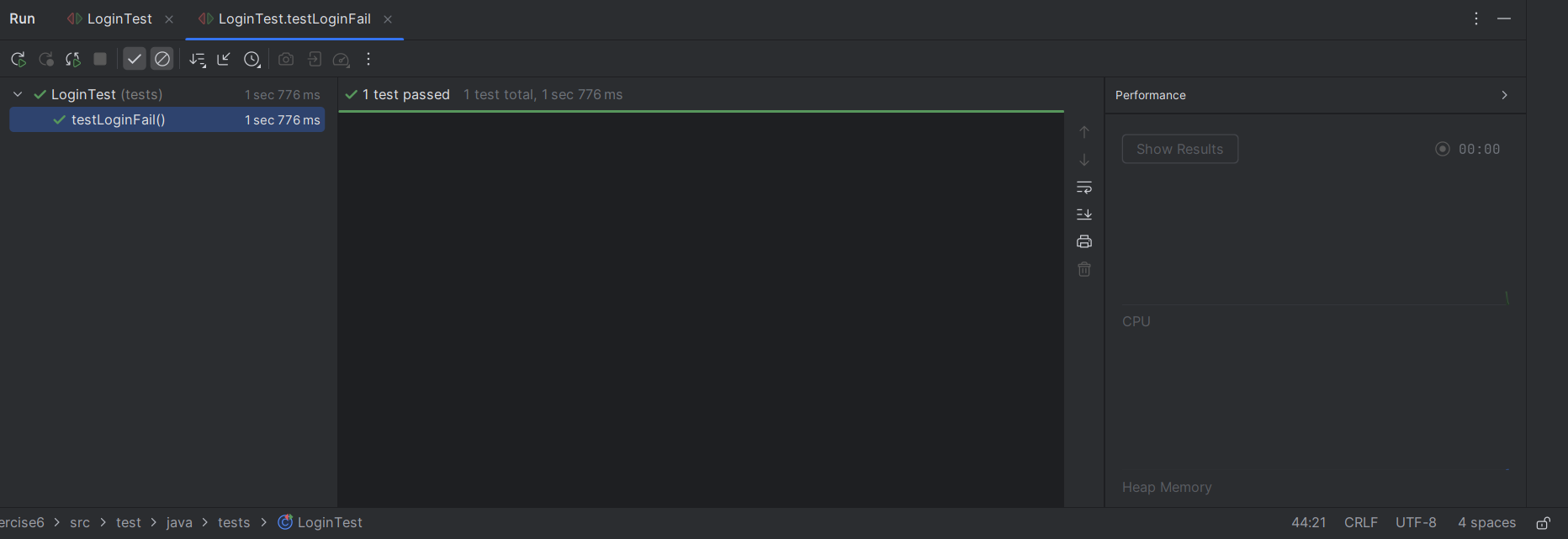
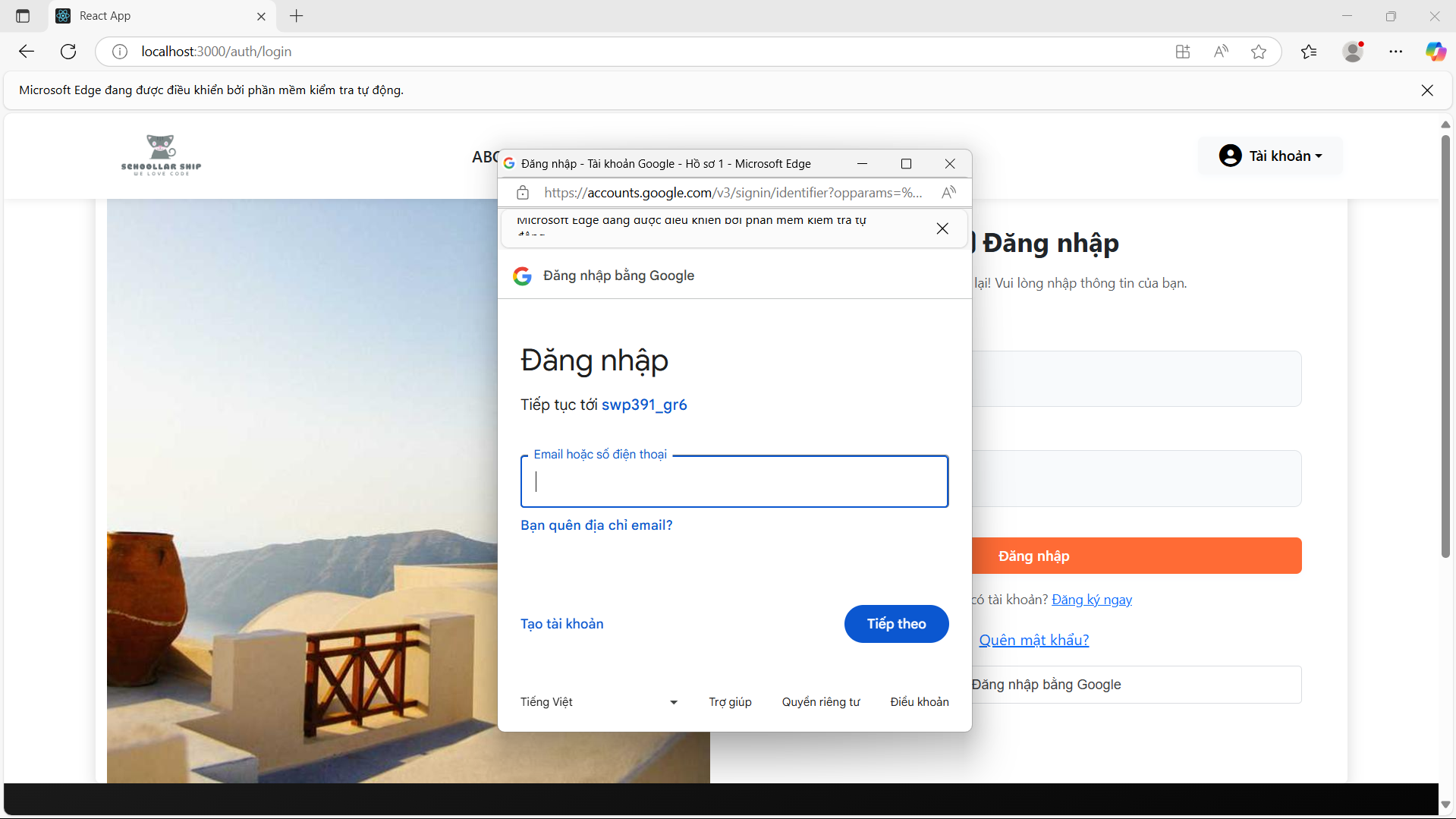
package tests;  
  
import org.junit.jupiter.api.\*;  
import org.openqa.selenium.\*;  
import org.openqa.selenium.support.ui.\*;  
import pages.RegisterPage;  
import utils.DriverFactory;  
  
import java.time.Duration;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
public class RegisterTest {  
 static WebDriver *driver*;  
 static RegisterPage *registerPage*;  
 static WebDriverWait *wait*;  
  
 @BeforeAll  
 static void setup() {  
 *driver* = DriverFactory.*createDriver*();  
 *registerPage* = new RegisterPage(*driver*);  
 *wait* = new WebDriverWait(*driver*, Duration.*ofSeconds*(10));  
 }  
  
 @Test  
 void testFullRegisterFlow() {  
  
 *registerPage*.navigate();  
 *registerPage*.sendOtp("namrp103@gmail.com");  
  
 System.*out*.println("👉 Vui lòng nhập OTP thủ công trong vòng 60 giây...");  
 new WebDriverWait(*driver*, Duration.*ofSeconds*(60))  
 .until(ExpectedConditions.*presenceOfElementLocated*(By.*name*("name"))); // chờ input name  
  
 *registerPage*.fillRegistrationForm();  
 *registerPage*.submitRegistrationForm();  
  
 new WebDriverWait(*driver*, Duration.*ofSeconds*(10))  
 .until(ExpectedConditions.*urlContains*("/auth/login"));  
  
 *assertTrue*(*driver*.getCurrentUrl().contains("/auth/login"));  
 System.*out*.println("✅ Đăng ký thành công!");  
 }  
  
  
 @AfterAll  
 static void tearDown() {  
 *driver*.quit();  
 }  
}

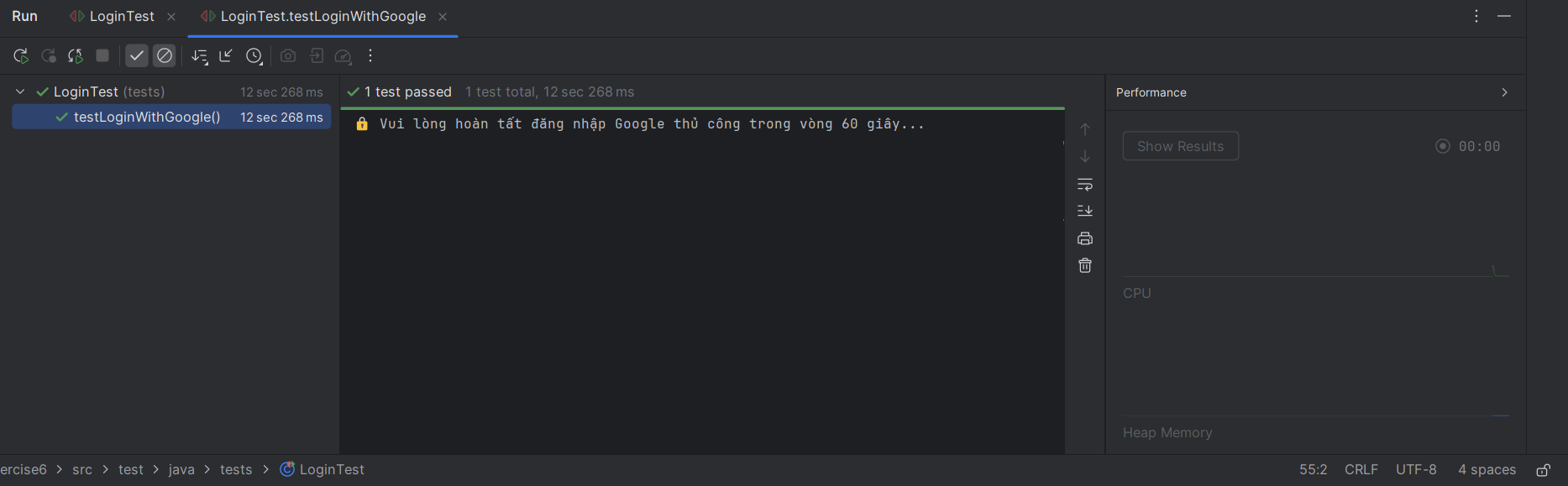
* Gửi OTP đến email thật.
* Chờ người dùng nhập OTP.
* Sau khi hiện ra form, tự động điền form đăng ký và gửi.
* Kiểm tra người dùng được điều hướng sang trang login.

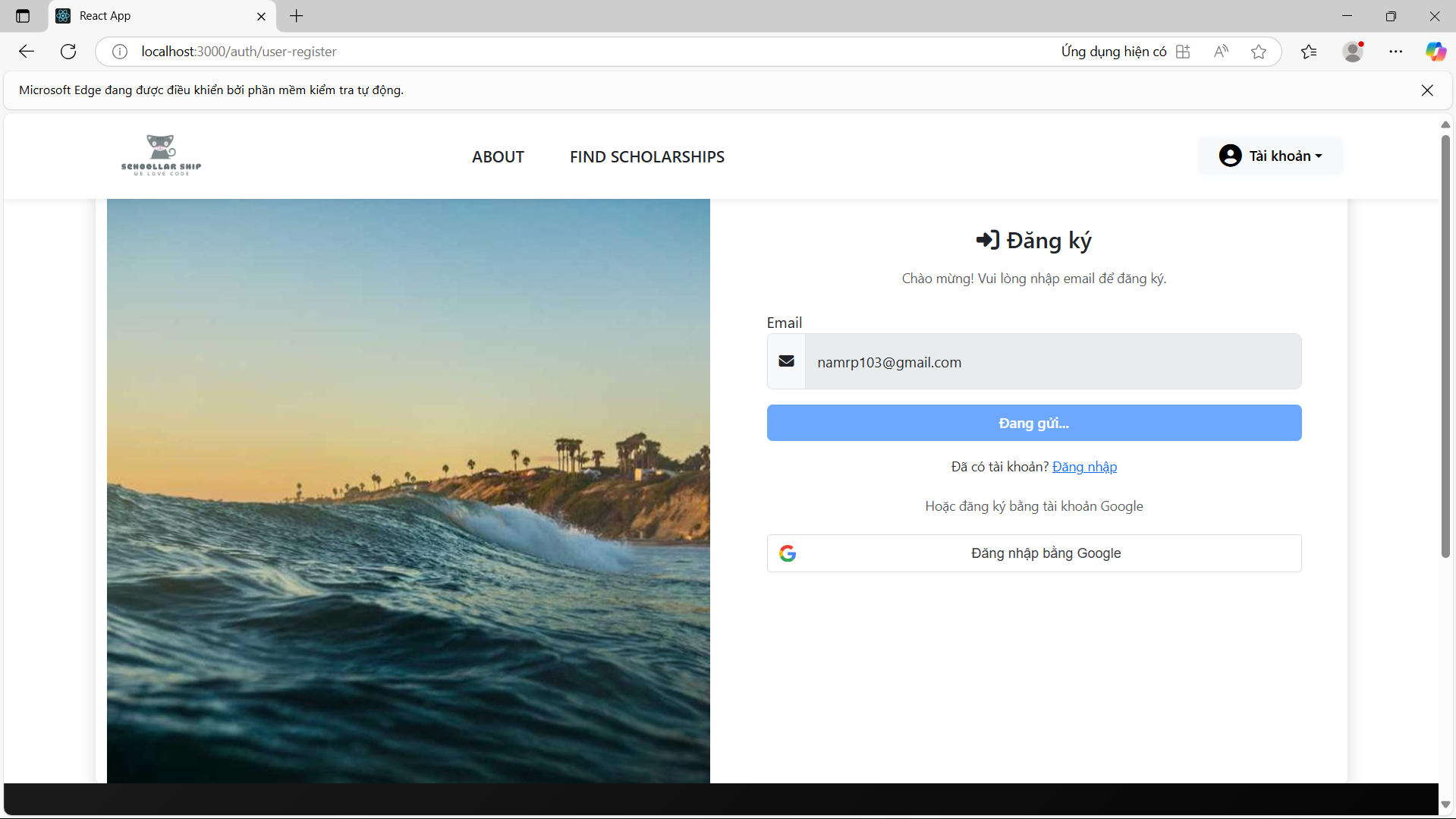
**Tổng kết các kỹ thuật đã dùng:**

| **Kỹ thuật / Thư viện** | **Mục đích** |
| --- | --- |
| Selenium WebDriver | Điều khiển trình duyệt tự động |
| JUnit 5 | Viết unit test |
| WebDriverWait + ExpectedConditions | Chờ đợi phần tử hiển thị hoặc click được |
| Thread.sleep() | Chờ người dùng nhập OTP / Google |
| Select (Selenium) | Tương tác với select dropdown |
| assertTrue(...) | Kiểm tra kết quả đầu ra mong muốn |

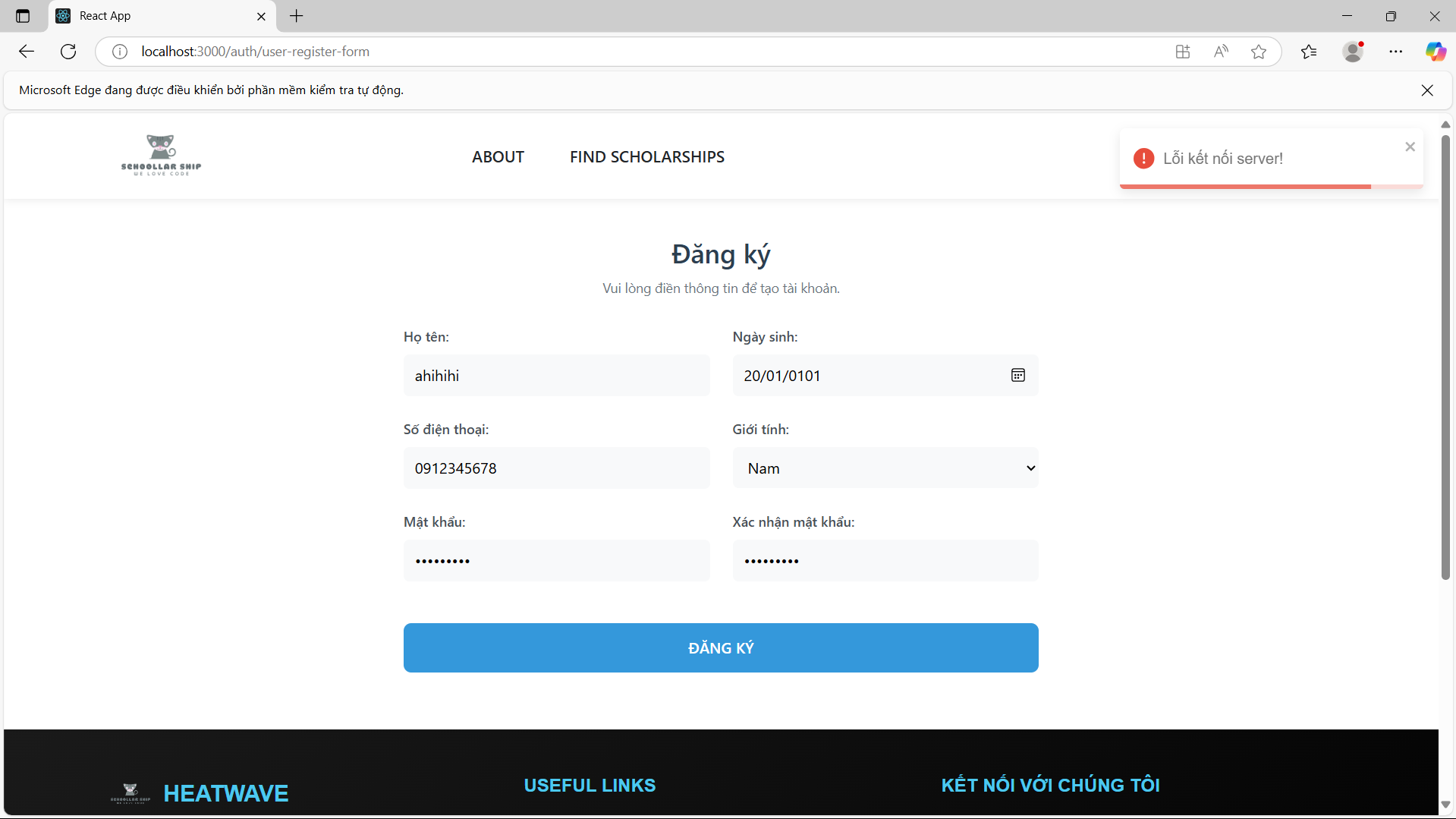
chạy các test  


Tự động nhấn vào nút đăng nhập bằng google để hiện ra popup để nhập thủ công  
  
set time nếu không nhập đúng thì sẽ trả về fail



Sẽ hoàn thành otp được gửi về gmail real.



Sẽ tự động điền thông tin user để đăng ký  
tuy nhiên bị lỗi kết nối Server nhma jdbc vẫn lưu thông tin đăng nhập chính xác

'USER0000000007', '2025-07-10 00:32:13.000000', '0101-01-19', 'namrp103@gmail.com', 'male', NULL, 'ahihihi', '$2a$10$1KWqaAjkVmnH8czYUiR97eJ.HuIO.pvcJJ/DyfdISNZLooeS.fMU6', '0912345678', 'seeker', NULL, '2025-07-20 15:29:13.464793'  
nên em không biết là có lỗi ở case này hay không…