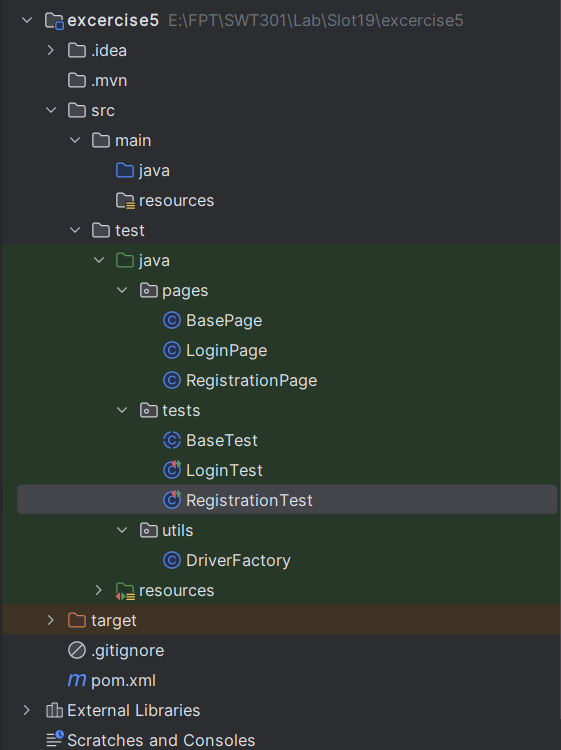
# **Exercise 5: Testing Registration Page Using Page Object Model**

**Implementation Steps**

## **Step 1: Set up project structure according to Page Object Model**

* Create pages directory to contain Page Object classes
* Create tests directory to contain Test classes
* Create utils directory to contain utility classes
* Create file pom.xml

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>

    <artifactId>exercise5</artifactId>

    <version>1.0-SNAPSHOT</version>

    <properties>

        <maven.compiler.source>17</maven.compiler.source>

        <maven.compiler.target>17</maven.compiler.target>

        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

    </properties>

    <dependencies>

*<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->*

        <dependency>

            <groupId>org.seleniumhq.selenium</groupId>

            <artifactId>selenium-java</artifactId>

            <version>4.21.0</version>

        </dependency>

*<!-- https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->*

        <dependency>

            <groupId>io.github.bonigarcia</groupId>

            <artifactId>webdrivermanager</artifactId>

            <version>5.8.0</version>

        </dependency>

        <dependency>

            <groupId>org.junit.jupiter</groupId>

            <artifactId>junit-jupiter-engine</artifactId>

            <version>5.10.2</version>

            <scope>test</scope>

        </dependency>

*<!-- JUnit 5 Parameterized Test -->*

        <dependency>

            <groupId>org.junit.jupiter</groupId>

            <artifactId>junit-jupiter-params</artifactId>

            <version>5.10.2</version>

            <scope>test</scope>

        </dependency>

        <dependency>

            <groupId>io.github.bonigarcia</groupId>

            <artifactId>webdrivermanager</artifactId>

            <version>5.7.0</version>

        </dependency>

    </dependencies>

</project>

## **Step 2: Create DriverFactory class**

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;  
  
import java.util.HashMap;  
import java.util.Map;  
  
public class DriverFactory {  
 public static WebDriver createDriver() {  
 WebDriverManager.*chromedriver*().setup();  
  
 ChromeOptions options = new ChromeOptions();  
 Map<String, Object> prefs = new HashMap<>();  
 prefs.put("profile.managed\_default\_content\_settings.javascript", 1); // Allow JavaScript  
 prefs.put("profile.default\_content\_setting\_values.ads", 2); // Block ads  
 options.setExperimentalOption("prefs", prefs);  
   
 // Add additional options  
 options.addArguments("--incognito"); // Incognito mode  
 options.addArguments("--disable-notifications"); // Disable notifications  
 options.addArguments("--disable-popup-blocking"); // Disable popup blocking  
 options.addArguments("--disable-extensions"); // Disable extensions  
 options.addArguments("--disable-infobars"); // Disable infobars  
 options.addArguments("--disable-gpu"); // Disable GPU acceleration  
 options.addArguments("--disable-dev-shm-usage"); // Overcome limited resource problems  
 options.addArguments("--no-sandbox"); // Bypass OS security model  
  
 return new ChromeDriver(options);  
 }  
}

**Explanation**: This class helps initialize WebDriver with appropriate configurations, separating browser initialization logic from test code.

## **Step 3:** Create BasePage class

package pages;  
  
import org.openqa.selenium.\*;  
import org.openqa.selenium.support.ui.\*;  
import org.openqa.selenium.interactions.Actions;  
  
import java.time.Duration;  
  
public class BasePage {  
 protected WebDriver driver;  
 protected WebDriverWait wait;  
 protected Actions actions;  
  
 public BasePage(WebDriver driver) {  
 this.driver = driver;  
 this.wait = new WebDriverWait(driver, Duration.*ofSeconds*(10));  
 this.actions = new Actions(driver);  
 }  
  
 // 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));  
 }  
  
 // Click safely  
 protected void click(By locator) {  
 waitForClickability(locator).click();  
 }  
  
 // 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 present  
 protected boolean isElementVisible(By locator) {  
 try {  
 return waitForVisibility(locator).isDisplayed();  
 } catch (TimeoutException e) {  
 return false;  
 }  
 }  
   
 // Scroll to element  
 protected void scrollToElement(By locator) {  
 WebElement element = waitForVisibility(locator);  
 ((JavascriptExecutor) driver).executeScript("arguments[0].scrollIntoView(true);", element);  
 }  
   
 // Select from dropdown by visible text  
 protected void selectByVisibleText(By locator, String text) {  
 WebElement element = waitForVisibility(locator);  
 Select select = new Select(element);  
 select.selectByVisibleText(text);  
 }  
   
 // Select radio button  
 protected void selectRadioButton(By locator) {  
 WebElement radio = waitForClickability(locator);  
 if (!radio.isSelected()) {  
 radio.click();  
 }  
 }  
   
 // Check checkbox  
 protected void checkCheckbox(By locator) {  
 WebElement checkbox = waitForClickability(locator);  
 if (!checkbox.isSelected()) {  
 checkbox.click();  
 }  
 }  
   
 // Upload file  
 protected void uploadFile(By locator, String filePath) {  
 WebElement fileInput = driver.findElement(locator);  
 fileInput.sendKeys(filePath);  
 }  
}

**Explanation**: BasePage is the base class containing common methods for all pages, helping to reuse code and minimize duplication.

## **Step 4:** Create BaseTest class

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();  
 }  
 }  
}

**Explanation**: BaseTest manages the WebDriver lifecycle, ensuring proper initialization and closure of the browser.

## **Step 5:** Create RegistrationPage class extending BasePage

package pages;

public class RegistrationPage extends BasePage {

*// Locators*

    private final By firstNameField = By.id("firstName");

    private final By lastNameField = By.id("lastName");

*// ...*

    public RegistrationPage(WebDriver *driver*) {

        super(driver);

    }

*// Actions*

    public void navigate() {

        navigateTo("https://demoqa.com/automation-practice-form");

    }

    public void setFirstName(String *firstName*) {

        type(firstNameField, firstName);

    }

*// Other methods...*

*// Validation methods*

    public boolean isFirstNameValidationDisplayed() {

        return isElementVisible(firstNameValidation);

    }

*// Other validation methods...*

}

**Explanation**: RegistrationPage represents the registration page, containing elements, actions, and validation methods related to this page.

## **Step 6:** Create comprehensive test data in CSV file

firstName,lastName,email,gender,mobile,day,month,year,subjects,hobbies,address,state,city

John,Doe,john.doe@example.com,Male,1234567890,15,May,1990,Maths,Sports,123 Main Street,NCR,Delhi

Jane,Smith,jane.smith@example.com,Female,9876543210,25,June,1995,English,Reading,456 Oak Avenue,Uttar Pradesh,Agra

Alex,Johnson,alex@example.com,Other,5556667777,10,January,2000,Computer Science,Music,789 Pine Road,Haryana,Karnal

,,invalid@email,Male,12345,1,April,1985,Physics,Sports,Short Address,Rajasthan,Jaipur

" ",Smith,space.first@example.com,Female,9090909090,5,March,1992,Chemistry,Reading,With Space,NCR,Delhi

John," ",space.last@example.com,Male,8080808080,12,July,1988,Biology,Music,Last Space,Haryana,Panipat

"J@#$%",Doe,special.first@example.com,Other,7070707070,20,August,1997,Maths,Sports,Special Chars,Uttar Pradesh,Agra

John,"D@#$%",special.last@example.com,Male,6060606060,18,September,1993,English,Reading,Special Chars,Rajasthan,Jaipur

John123,Doe456,numeric.name@example.com,Female,5050505050,22,October,1991,Computer Science,Music,Numeric Name,Haryana,Karnal

John,Doe,invalid-email,Male,4040404040,30,November,1989,Physics,Sports,Invalid Email,NCR,Delhi

John,Doe,valid@email.com,Male,40404,30,November,1989,Physics,Sports,Short Mobile,Uttar Pradesh,Lucknow

John,Doe,valid@email.com,Male,40404040404040404040,30,November,1989,Physics,Sports,Long Mobile,Rajasthan,Jaipur

" ",Doe,long.space@email.com,Female,1212121212,2,December,1994,Chemistry,Reading,Long Space,Haryana,Panipat

**Explanation**: The CSV file contains multiple test data sets covering various edge cases like special characters, whitespace, invalid formats, etc.

## **Step 7:** Create comprehensive RegistrationTest class

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.Arrays;

import java.util.List;

import java.util.stream.Collectors;

public class RegistrationPage extends BasePage {

*// Locators*

    private final By firstNameField = By.id("firstName");

    private final By lastNameField = By.id("lastName");

    private final By emailField = By.id("userEmail");

    private final By maleGenderRadio = By.xpath("//label[text()='Male']");

    private final By femaleGenderRadio = By.xpath("//label[text()='Female']");

    private final By otherGenderRadio = By.xpath("//label[text()='Other']");

    private final By mobileNumberField = By.id("userNumber");

    private final By dateOfBirthField = By.id("dateOfBirthInput");

    private final By subjectsInput = By.id("subjectsInput");

    private final By sportsHobby = By.xpath("//label[text()='Sports']");

    private final By readingHobby = By.xpath("//label[text()='Reading']");

    private final By musicHobby = By.xpath("//label[text()='Music']");

    private final By uploadPictureBtn = By.id("uploadPicture");

    private final By currentAddressField = By.id("currentAddress");

    private final By stateDropdown = By.id("state");

    private final By stateInput = By.id("react-select-3-input");

    private final By cityDropdown = By.id("city");

    private final By cityInput = By.id("react-select-4-input");

    private final By submitButton = By.id("submit");

*// Form validation elements*

    private final By firstNameValidation = By.xpath("//input[@id='firstName']/following-sibling::div[contains(@class, 'error')]");

    private final By lastNameValidation = By.xpath("//input[@id='lastName']/following-sibling::div[contains(@class, 'error')]");

    private final By emailValidation = By.xpath("//input[@id='userEmail']/following-sibling::div[contains(@class, 'error')]");

    private final By mobileValidation = By.xpath("//input[@id='userNumber']/following-sibling::div[contains(@class, 'error')]");

*// Confirmation elements*

    private final By confirmationModal = By.className("modal-content");

    private final By confirmationModalTitle = By.id("example-modal-sizes-title-lg");

    private final By confirmationTable = By.className("table-responsive");

    public RegistrationPage(WebDriver *driver*) {

        super(*driver*);

    }

    public void **navigate**() {

        navigateTo("https://demoqa.com/automation-practice-form");

    }

    public void **setFirstName**(String *firstName*) {

        type(firstNameField, *firstName*);

    }

    public void **setLastName**(String *lastName*) {

        type(lastNameField, *lastName*);

    }

    public void **setEmail**(String *email*) {

        type(emailField, *email*);

    }

    public void **selectGender**(String *gender*) {

        switch (*gender*.toLowerCase()) {

            case "male":

                clickWithJS(maleGenderRadio);

                break;

            case "female":

                clickWithJS(femaleGenderRadio);

                break;

            case "other":

                clickWithJS(otherGenderRadio);

                break;

            default:

                throw new IllegalArgumentException("Invalid gender: " + *gender*);

        }

    }

    public void **setMobileNumber**(String *mobileNumber*) {

        type(mobileNumberField, *mobileNumber*);

    }

    public void **setDateOfBirth**(String *day*, String *month*, String *year*) {

*// Sử dụng JavaScript để mở date picker thay vì click trực tiếp*

        WebElement dateElement = waitForVisibility(dateOfBirthField);

        JavascriptExecutor js = (JavascriptExecutor) driver;

        js.executeScript("arguments[0].click();", dateElement);

        try {

*// Chờ một chút để date picker hiển thị*

            Thread.sleep(500);

*// Select year*

            By yearDropdown = By.className("react-datepicker\_\_year-select");

            selectByVisibleText(yearDropdown, *year*);

*// Select month*

            By monthDropdown = By.className("react-datepicker\_\_month-select");

            selectByVisibleText(monthDropdown, *month*);

*// Select day bằng JavaScript*

            By daySelector = By.xpath("//div[contains(@class, 'react-datepicker\_\_day') and text()='" + *day* + "']");

            WebElement dayElement = waitForVisibility(daySelector);

            js.executeScript("arguments[0].click();", dayElement);

        } catch (InterruptedException *e*) {

            Thread.currentThread().interrupt();

        }

    }

    public void **setSubjects**(String... *subjects*) {

        for (String subject : *subjects*) {

            WebElement subjectField = waitForVisibility(subjectsInput);

            subjectField.sendKeys(subject);

            try {

                Thread.sleep(500); *// Chờ để suggestions hiển thị*

                By option = By.xpath("//div[contains(@class, 'subjects-auto-complete\_\_option') and contains(text(), '" + subject + "')]");

                clickWithJS(option);

            } catch (InterruptedException *e*) {

                Thread.currentThread().interrupt();

            } catch (Exception *e*) {

                System.out.println("Could not select subject: " + subject + " - " + e.getMessage());

            }

        }

    }

    public void **selectHobbies**(String... *hobbies*) {

        for (String hobby : *hobbies*) {

            switch (hobby.toLowerCase()) {

                case "sports":

                    clickWithJS(sportsHobby);

                    break;

                case "reading":

                    clickWithJS(readingHobby);

                    break;

                case "music":

                    clickWithJS(musicHobby);

                    break;

                default:

                    throw new IllegalArgumentException("Invalid hobby: " + hobby);

            }

        }

    }

    public void **uploadPicture**(String *filePath*) {

        uploadFile(uploadPictureBtn, *filePath*);

    }

    public void **setCurrentAddress**(String *address*) {

        type(currentAddressField, *address*);

    }

    public void **setState**(String *state*) {

        scrollToElement(stateDropdown);

        clickWithJS(stateDropdown);

        try {

            Thread.sleep(500); *// Chờ dropdown hiển thị*

            WebElement input = waitForVisibility(stateInput);

            input.sendKeys(*state*);

            input.sendKeys(org.openqa.selenium.Keys.ENTER);

        } catch (InterruptedException *e*) {

            Thread.currentThread().interrupt();

        }

    }

    public void **setCity**(String *city*) {

        scrollToElement(cityDropdown);

        clickWithJS(cityDropdown);

        try {

            Thread.sleep(500); *// Chờ dropdown hiển thị*

            WebElement input = waitForVisibility(cityInput);

            input.sendKeys(*city*);

            input.sendKeys(org.openqa.selenium.Keys.ENTER);

        } catch (InterruptedException *e*) {

            Thread.currentThread().interrupt();

        }

    }

    public void **submitForm**() {

        scrollToElement(submitButton);

        clickWithJS(submitButton);

    }

    public boolean **isConfirmationDisplayed**() {

        try {

            return isElementVisible(confirmationModal);

        } catch (Exception *e*) {

            return false;

        }

    }

    public String **getConfirmationTitle**() {

        return getText(confirmationModalTitle);

    }

    public boolean **verifySubmittedData**(String *label*, String *expectedValue*) {

        try {

            By rowLocator = By.xpath("//div[@class='table-responsive']//tr[td[text()='" + *label* + "']]/td[2]");

            String actualValue = getText(rowLocator);

            return actualValue.equals(*expectedValue*);

        } catch (Exception *e*) {

            return false;

        }

    }

*// New methods for validation checking*

    public boolean **isFirstNameValidationDisplayed**() {

        return isElementVisible(firstNameValidation);

    }

    public boolean **isLastNameValidationDisplayed**() {

        return isElementVisible(lastNameValidation);

    }

    public boolean **isEmailValidationDisplayed**() {

        return isElementVisible(emailValidation);

    }

    public boolean **isMobileValidationDisplayed**() {

        return isElementVisible(mobileValidation);

    }

    public String **getFirstNameValidationMessage**() {

        if (isFirstNameValidationDisplayed()) {

            return getText(firstNameValidation);

        }

        return "";

    }

    public String **getLastNameValidationMessage**() {

        if (isLastNameValidationDisplayed()) {

            return getText(lastNameValidation);

        }

        return "";

    }

    public String **getEmailValidationMessage**() {

        if (isEmailValidationDisplayed()) {

            return getText(emailValidation);

        }

        return "";

    }

    public String **getMobileValidationMessage**() {

        if (isMobileValidationDisplayed()) {

            return getText(mobileValidation);

        }

        return "";

    }

*// Helper method to get all selected hobbies*

    public List<String> **getSelectedHobbies**() {

        List<String> selectedHobbies = Arrays.asList("Sports", "Reading", "Music").stream()

                .filter(*hobby* -> {

                    By hobbyLocator = null;

                    switch (*hobby*) {

                        case "Sports":

                            hobbyLocator = sportsHobby;

                            break;

                        case "Reading":

                            hobbyLocator = readingHobby;

                            break;

                        case "Music":

                            hobbyLocator = musicHobby;

                            break;

                    }

                    try {

                        WebElement element = driver.findElement(hobbyLocator);

                        return element.isSelected();

                    } catch (Exception *e*) {

                        return false;

                    }

                })

                .collect(Collectors.toList());

        return selectedHobbies;

    }

*// Helper method to click using JavaScript (useful for elements that might be covered by other elements)*

    private void **clickWithJS**(By *locator*) {

        WebElement element = waitForVisibility(*locator*);

        JavascriptExecutor js = (JavascriptExecutor) driver;

        js.executeScript("arguments[0].click();", element);

    }

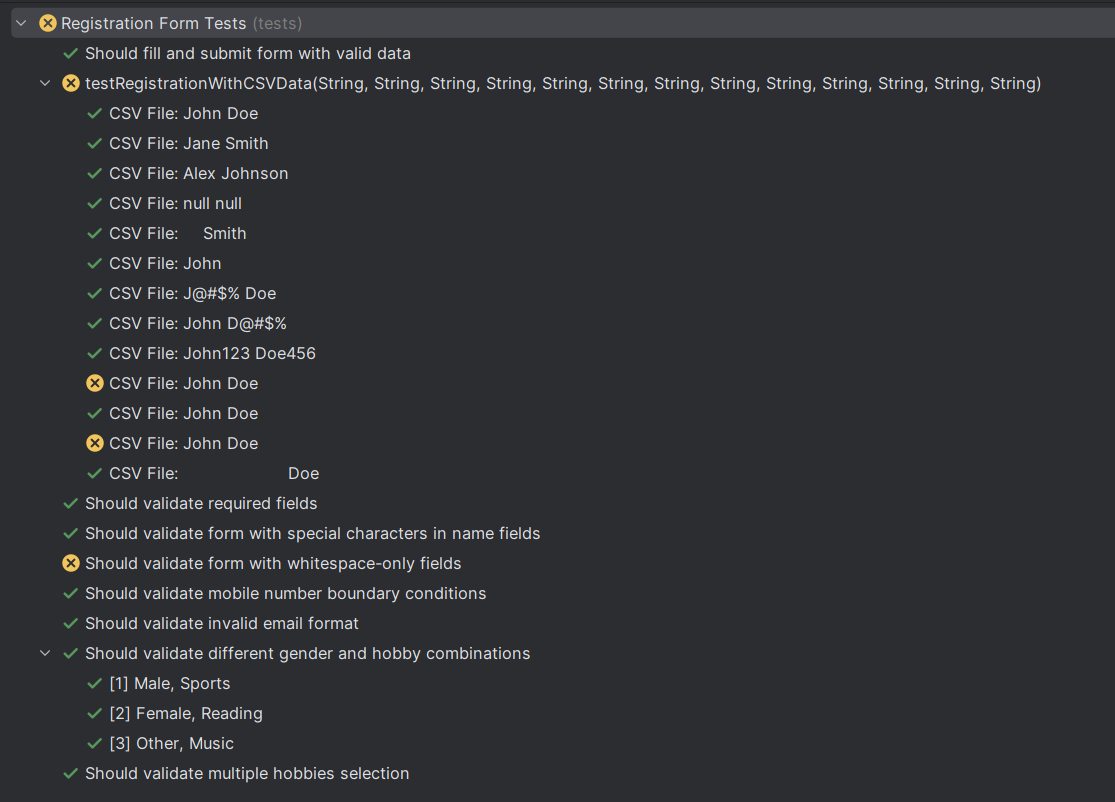
}

**Explanation**: RegistrationTest contains a comprehensive set of test cases covering basic functionality and various edge cases.

## **Step 8: Handle challenges and edge cases**

* Use JavaScript Executor to interact with elements blocked by advertisements
* Add exception handling to ensure tests don't stop when encountering errors
* Implement validation methods to check form validation messages
* Add specific tests for boundary conditions, invalid formats, and special characters

## **Result :**



### **Detailed Analysis**

CSV Data Test Results

Most CSV data tests passed successfully, including tests with:

* Standard valid data (John Doe, Jane Smith)
* Empty fields (null null)
* Whitespace in first name (" Smith")
* Special characters in names (J@#$% Doe, John D@#$%)
* Numeric characters in names (John123 Doe456)
* Long whitespace fields (" Doe")

The failures in two "John Doe" test cases might be due to:

1. Duplicate data in the CSV file causing conflicts
2. Specific issues with the form handling these particular combinations
3. Timing issues during test execution
4. Problems with advertisements or other elements blocking interaction

Edge Case Test Results

* Special characters test passed: The form accepts special characters in name fields, which is good for international names.
* Whitespace-only fields test failed: This suggests the form might be accepting whitespace-only input when it should reject it, which could be a potential bug in the application.
* Mobile number boundary tests passed: The form correctly rejects too short or too long mobile numbers.
* Invalid email format test passed: The form correctly validates email format.