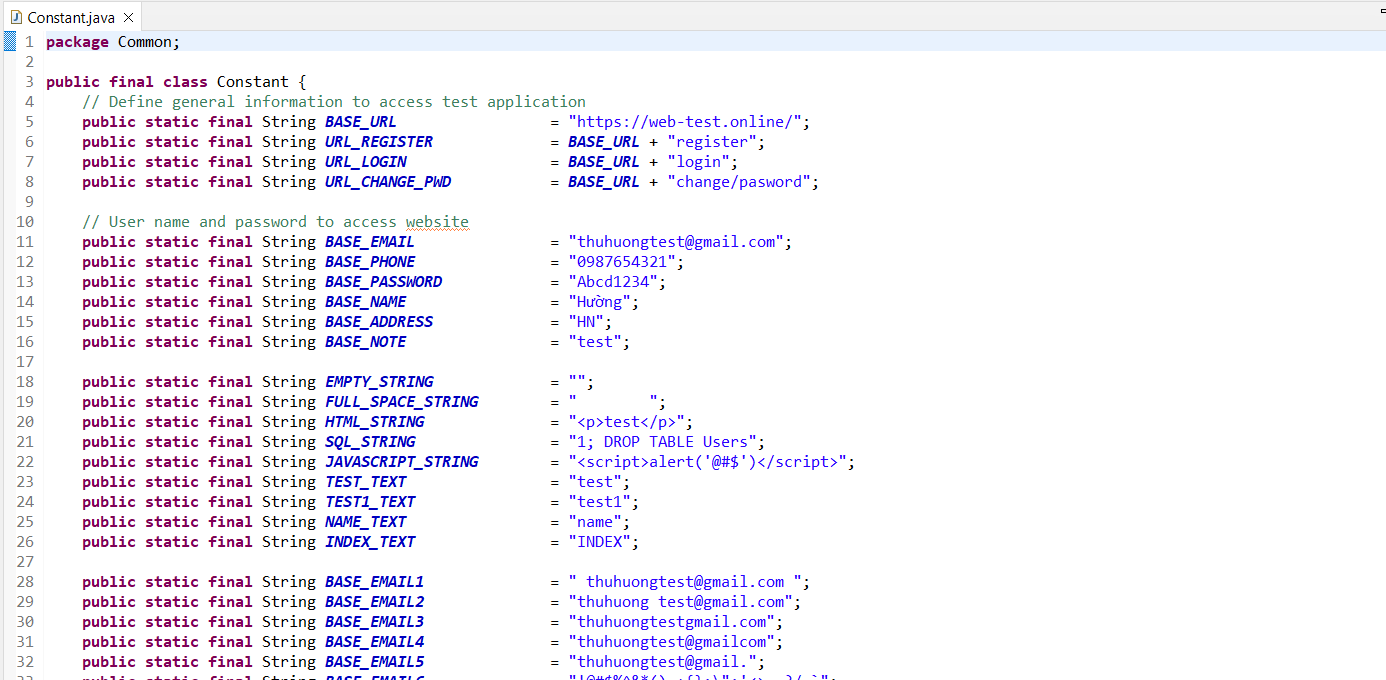
## **3.3 Kịch bản kiểm thử cho từng module →**

### ***3.3.1 Common folder***

Đây là nơi khai báo các biến và hàm chung

#### *3.3.1.1 Constant.java*

File chứa các môi trường test, các hằng số (là giá trị không thể thay đổi sau khi đã gán) sử dụng chung



Cú pháp khai báo hằng như sau:

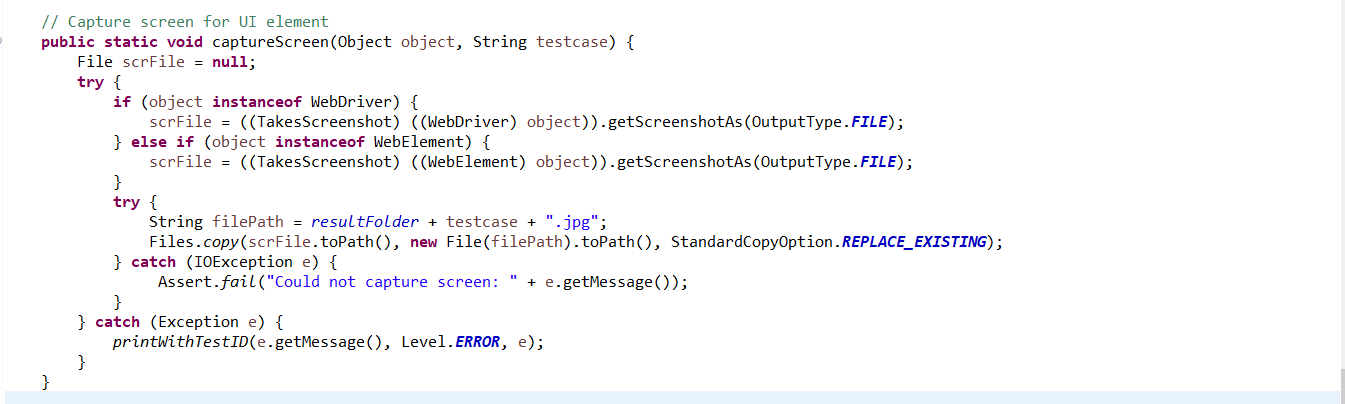
**static** **final** datatype identifier\_name=value;

#### *3.3.1.2 Utilities.java*

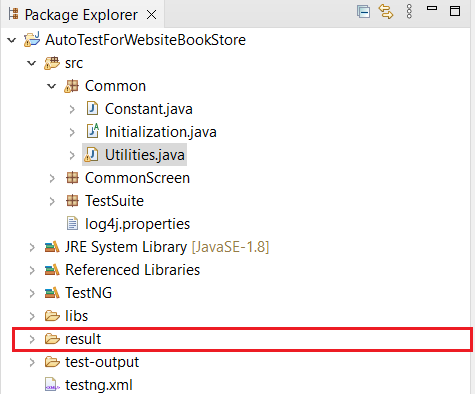
File code các module sau:

* CaptureScreen: Ảnh chụp màn hình khi test

Vì chụp ảnh là 1 phần tách riêng, nó không liên quan gì đến quá trình test cũng những là logic của app nên ta sẽ đặt nó ở trong package **utilities.**



Khi thực hiện 1 case , sẽ có ảnh chụp màn hình case đó, được lưu vào folder result



Trong đó: String filePath = *resultFolder* + testcase + ".jpg" : Đặt tên cho file ảnh, gồm có tên testcase+ extension

* **Tên testcase** chính là tên của method Test
* **Extention** là file có đuôi là .jpg

Có thể trigger việc chụp ảnh ở bất cứ nơi nào, nhưng để tránh việc phải viết quá nhiều, mình chỉ chụp ảnh sau khi run xong test, ở chỗ @AfterMethod. Ngoài ra thì method test của mình cũng nhiều nên mình chỉ muốn chụp ảnh cho các Test fail, mình sẽ không chụp cho method pass nữa.



Trong đó:

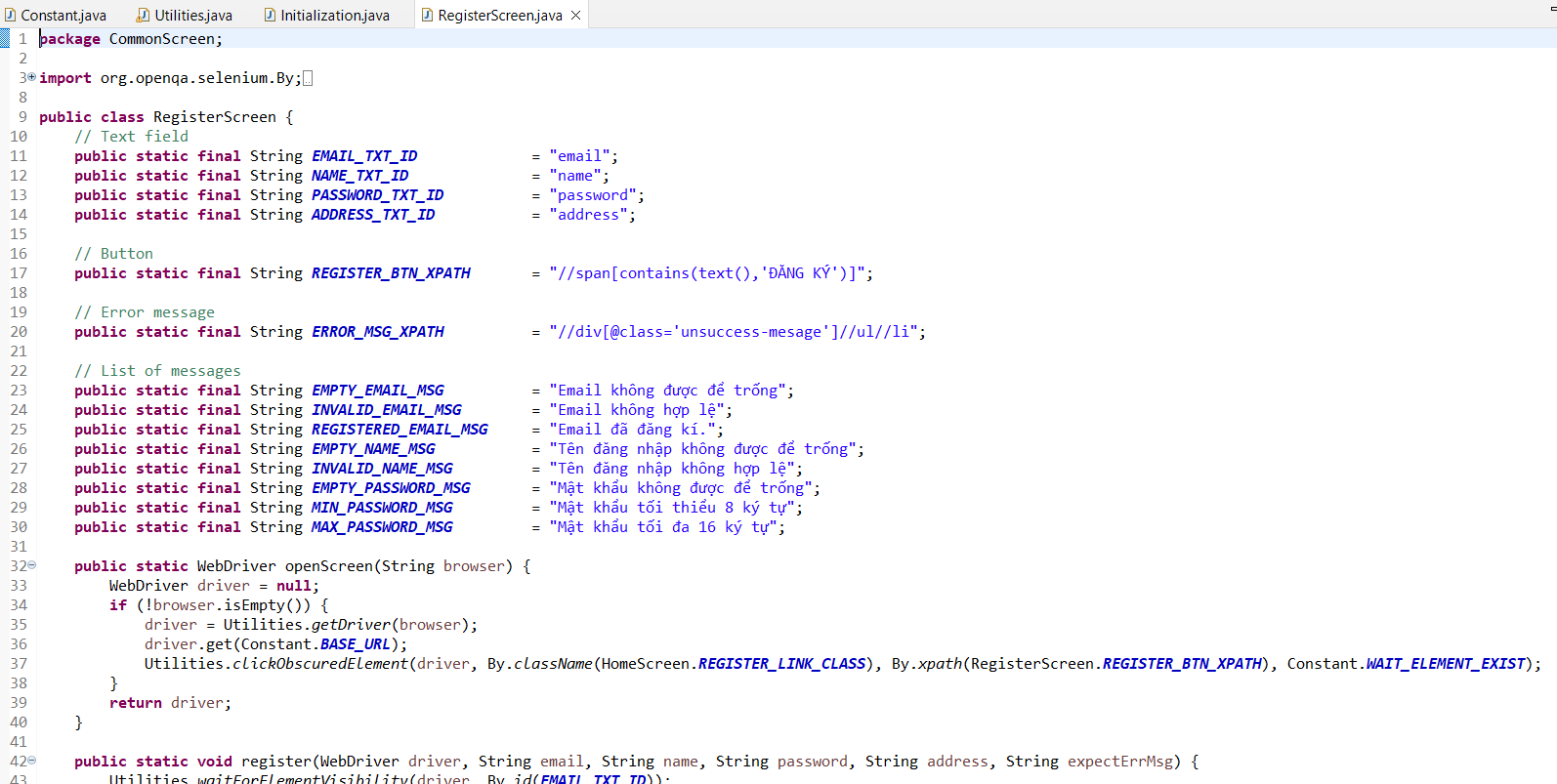
* Mình sử dụng Interface **ITestResult** của TestNG để có thể dễ dàng lấy được tên của method Test, cũng như check được xem kết quả của Test là Pass hay Fail.
* Chỉ khi nào fail thì mình sẽ gọi có function chụp screenshot với tham số đầu vào chính là tên của Method Test.

### ***3.3.2 CommonScreen folder***

Common screen là một khái niệm trong phát triển phần mềm, thường được sử dụng để chỉ những giao diện người dùng được chia sẻ hoặc tái sử dụng trong nhiều thành phần của hệ thống (Change password, Contact, home, login, register, update account). Điều này giúp tiết kiệm thời gian và công sức cho việc thiết kế, phát triển và bảo trì phần mềm, vì các thành phần có thể sử dụng lại các màn hình, nút bấm, biểu tượng và các thành phần giao diện khác mà không cần phải thiết kế lại từ đầu.

Common screen thường được lưu trữ trong các thư viện hoặc các gói (packages) trong mã nguồn của phần mềm. Các gói này có thể được import và sử dụng trong nhiều thành phần khác nhau của hệ thống. Chúng giúp đơn giản hóa quá trình phát triển và bảo trì phần mềm, đồng thời đảm bảo tính nhất quán và đồng bộ cho giao diện người dùng trên toàn hệ thống.

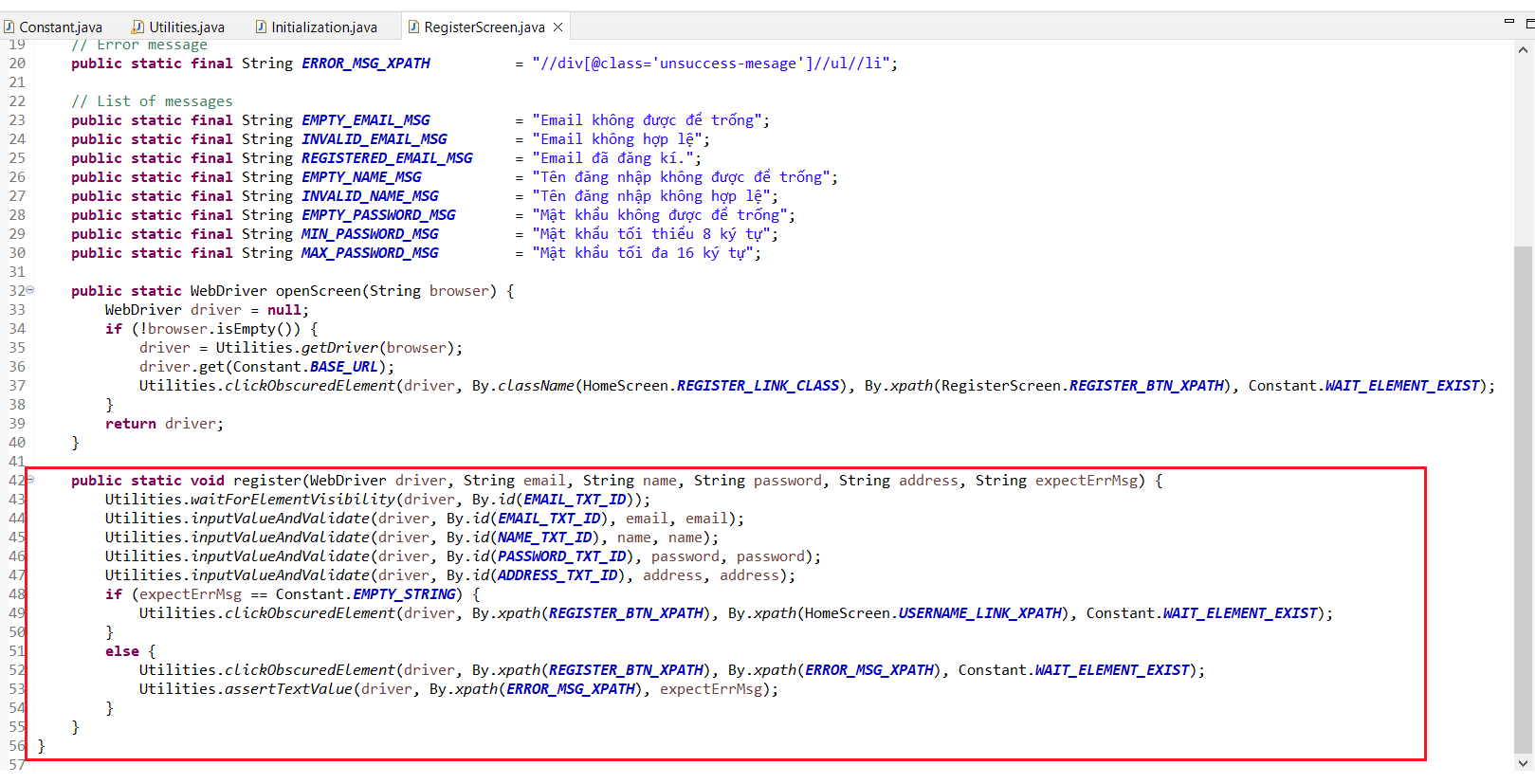
#### *3.3.2.1 Register screen*



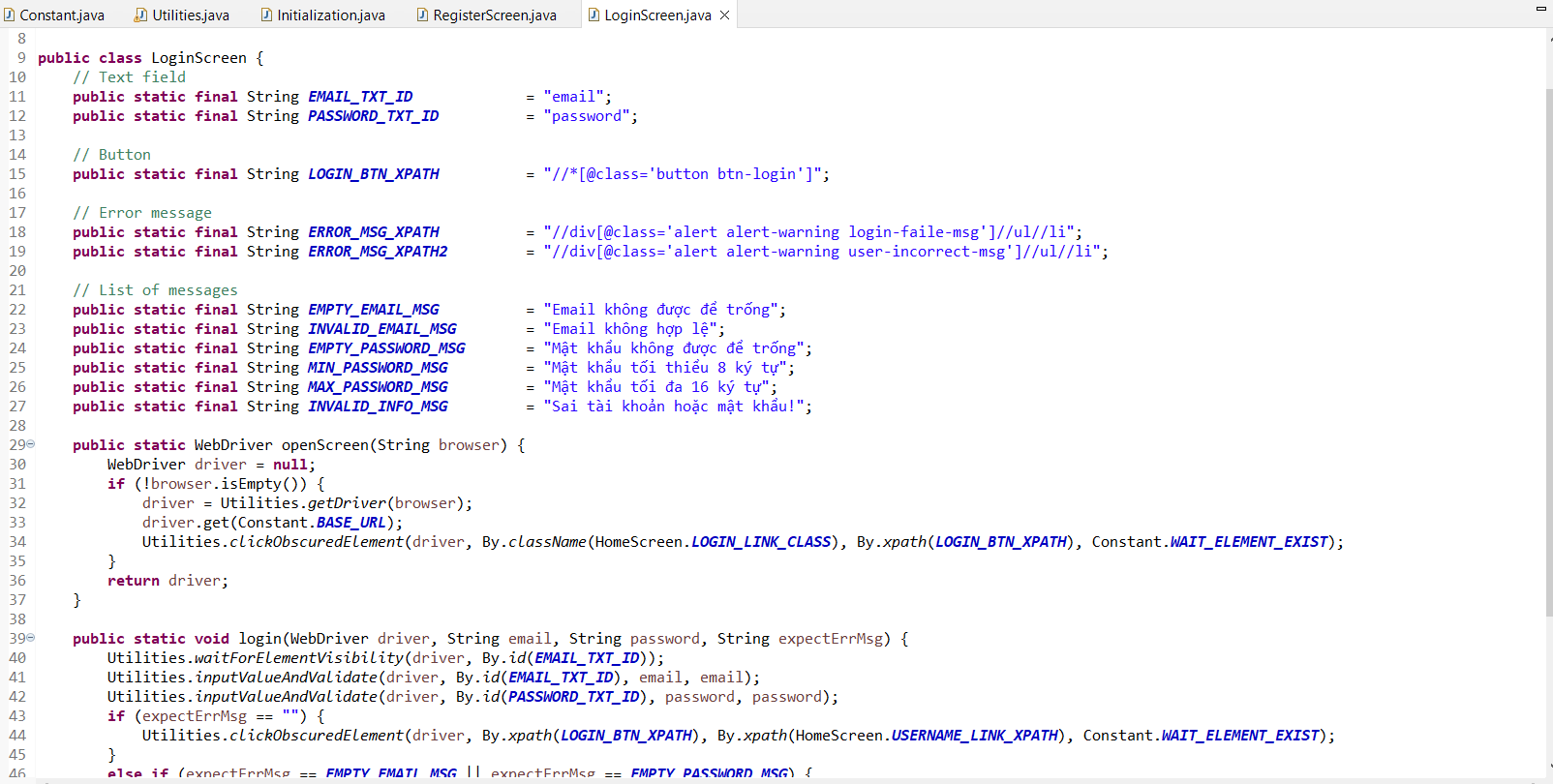
Class RegisterScreen chứa các phần tử web và các phương thức để đăng ký người dùng trên ứng dụng web. Lớp có một phương thức có tên là openScreen nhận vào kiểu trình duyệt làm tham số và trả về một đối tượng WebDriver. Phương thức register nhận vào một đối tượng WebDriver, email, tên đăng nhập, mật khẩu, địa chỉ và một thông báo lỗi dự kiến ​​như tham số.

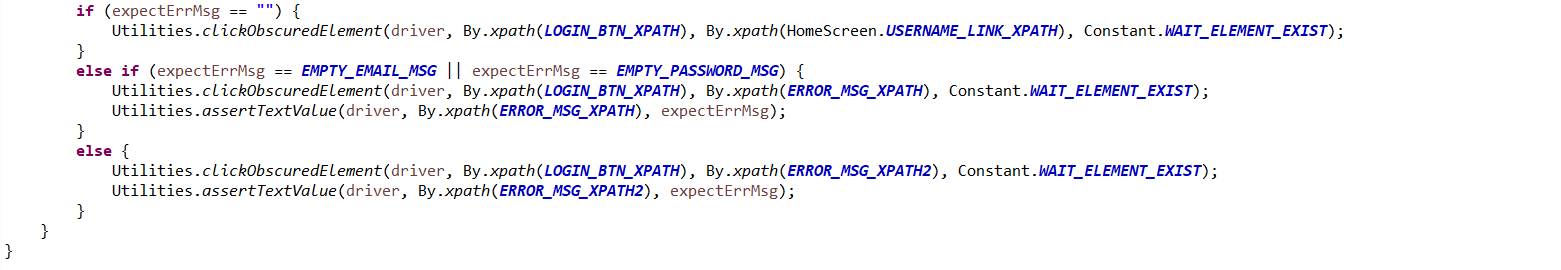
- Phương thức openScreen khởi chạy một trình duyệt và điều hướng đến URL cơ sở của ứng dụng. Sau đó, nó nhấp vào liên kết đăng ký để mở màn hình đăng ký.

- Phương thức register: được sử dụng để thực hiện việc nhập dữ liệu vào các trường nhập liệu trên trang đăng ký, sau đó kiểm tra và xử lý các thông báo lỗi (nếu có) dựa trên điều kiện trong biến expectErrMsg.



#### 3.3.2.2 Login screen

**

**

Phương thức openScreen nhận một chuỗi trình duyệt làm đầu vào, khởi tạo một thể hiện WebDriver bằng cách sử dụng phương thức getDriver từ lớp Utilities, và điều hướng đến một URL cơ bản. Sau đó, nó nhấn vào một liên kết đăng nhập bằng cách sử dụng phương thức clickObscuredElement từ cùng lớp Utilities.

Phương thức login nhận một thể hiện WebDriver,email, Tên đăng nhập và thông báo lỗi mong đợi làm đầu vào. Nó sử dụng phương thức waitForElementVisibility từ Utilities để chờ đợi trường văn bản điện thoại được hiển thị, nhập giá trị cho cả hai trường điện thoại và mật khẩu bằng cách sử dụng phương thức inputValueAndValidate từ cùng lớp Utilities. Tùy thuộc vào giá trị của expectErrMsg, nó sẽ nhấp vào nút đăng nhập hoặc xác nhận rằng thông báo lỗi cụ thể đã xuất hiện trên màn hình. Phương thức assertTextValue từ lớp Utilities được sử dụng để kiểm tra xem thông báo lỗi mong đợi có khớp với thông báo lỗi thực tế hiển thị trên màn hình hay không.

#### *3.3.2.3 Home screen*





- Phương thức openScreen nhận một chuỗi trình duyệt làm đầu vào, khởi tạo một thể hiện WebDriver bằng cách sử dụng phương thức getDriver từ lớp Utilities, truy cập đến URL cơ bản và đăng nhập vào ứng dụng bằng phương thức login của lớp LoginScreen.

- Phương thức search nhận một thể hiện WebDriver, từ khóa và một cờ giá trị Boolean để chỉ định kết quả mong đợi. Nó sử dụng phương thức inputValueAndValidate để nhập từ khóa và clickObscuredElement để nhấp vào biểu tượng tìm kiếm. Sau đó, nó kiểm tra số lượng kết quả tìm kiếm có đúng với mong đợi hay không.

- Phương thức register nhận một thể hiện WebDriver, một địa chỉ email và một thông báo lỗi mong đợi. Nó sử dụng phương thức inputValueAndValidate để nhập địa chỉ email và clickObscuredElement để nhấp vào nút đăng ký. Sau đó, nó sử dụng phương thức assertTextValueVisible để xác nhận rằng thông báo lỗi mong đợi xuất hiện trên màn hình.

- Phương thức logout nhận một thể hiện WebDriver và sử dụng phương thức clickObscuredElement để đăng xuất khỏi ứng dụng.

#### *3.3.2.4 Change password screen*



