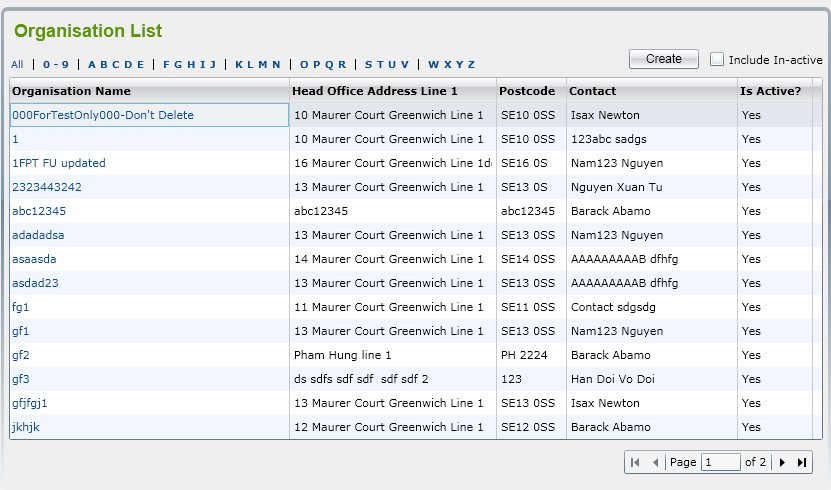
1. (6.2.1) List Organizations



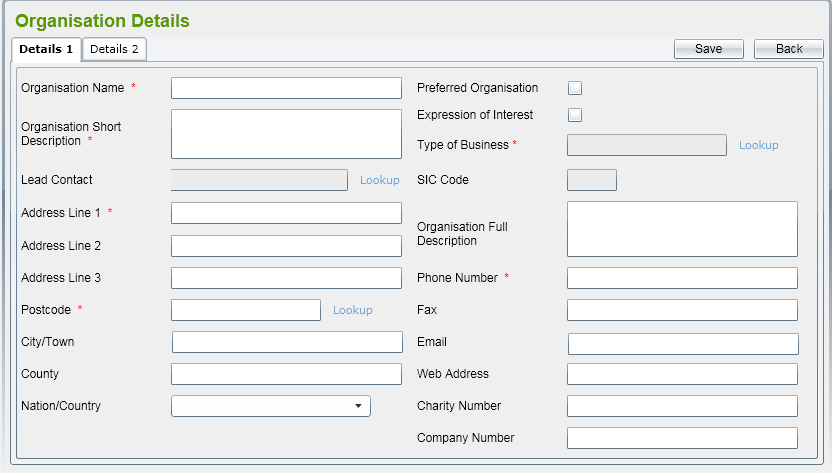
**Unit test**

* kiểm tra lấy dữ liệu ra từ cơ sở dữ liệu
* các biến mapping với bảng đúng với cơ sở dữ liệu
* các biến có kiểu dữ liệu ‘Date’ phải được convert sang dạng dd/MM/yyyy
* lấy dữ liệu ra xong phải đóng luồng

**Integration Test**

* Kiểm tra khi nhấp vào ô 'Include In-active' thì tất cả Organisations đều được hiển thị
* kiểm tra xem khi chọn vào Organisations hộp thoại thông báo có hiển thị
* Kiểm tra dữ liệu hiển thị tương ứng với các cột
* Kiểm tra khi nhấp vào nút OK dữ liệu của Organisations có được thay đổi
* Kiểm tra khi nhấp vào 'All' '0-9' hoặc các chữ từ 'a-z' thì các Organisations sẽ hiển thị theo yêu cầu lọc
* Đẩy lên git thì phải ‘pull’ source mới nhất tránh mất dữ liệu
* Các commit khi đẩy lên phải ghi rõ chức năng và tên người đẩy
* Tạo nhánh riêng không được code vào nhánh main

1. (6.2.2) Add Organisations



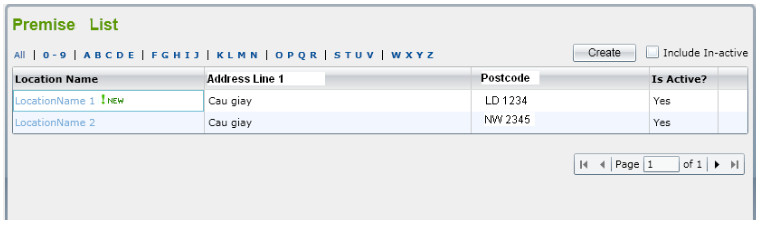
**test unit**

* Tên Class chứa đối tượng phải mang theo database
* Kiểu dữ liệu của biến phải phù hợp với dữ liệu tương thích trong DB
* Method của ADD phải có kiểu trả về
* Kiểm tra điều kiện của dữ liệu trước khi thay đổi để đảm bảo toàn vẹn dữ liệu
* Màn không được chứa hàm main chỉ được chạy một main duy nhất Applications

**Integration Test**

* kiểm tra trường hợp bắt buộc của input Organization Name
* kiểm tra tính duy nhất của input Organization Name
* kiểm tra trường bắt buộc Organization Short Description
* kiểm tra trường bắt buộc Type of Business
* kiểm tra trường bắt buộc Address Line 1
* kiểm tra trường bắt buộc Postcode
* kiểm tra trường bắt buộc Phone Number
* kiểm tra nút Save góc trên phải màn
* Kiểm tra nút Back góc trên phải màn
* kiểm tra Combobox Nation/Country
* ‘Pull’ mới sources trước khi code đảm bảo dữ liệu luôn mới nhất
* Commit ghi rõ các chức năng update và tên người comnit

1. test unit 6.5.1 List Premises



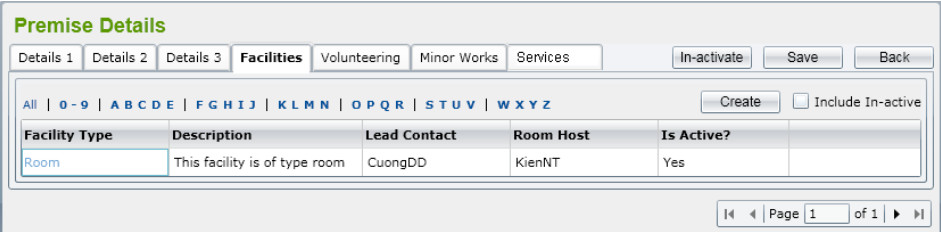
**test unit**

* Các biến mapping với bảng đúng với cơ sở dữ liệu
* Kiểm tra dữ liệu trong database
* Hàm CreatePremise có tham số đầu vào không
* Sự kiện Onclick trên table có gọi đến hàm onclick
* Kiểm tra dữ liệu trong cơ sở dữ liệu có khớp với dữ liệu trả ra của hàm getAllPremise.

**Integration test**

* Kiểm tra chức năng sắp xếp Premise bằng cách click đến tên cột.
* Kiểm tra chức năng click checkbox “Include In-active”, tất cả “Premise” hoạt động hay không hoạt động sẽ hiển thị trên list.
* Kiểm tra các dữ liệu load lên table đã được chuyền đúng cột chưa
* Kiểm tra chức năng chuyển trang khi click 

1. test unit 6.5.5 Facility Maintenance



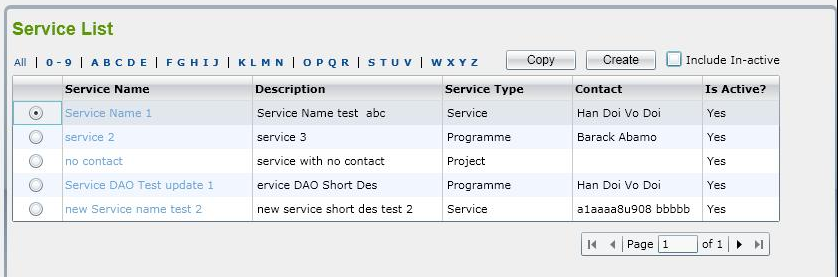
**test unit**

* kiểm tra dữ liệu lấy ra từ cơ sở dữ liệu có đúng theo yêu cầu
* Kiểu dữ liệu của biến phải phù hợp với dữ liệu tương thích trong database
* lấy dữ liệu ra xong phải đóng luồng
* Đẩy lên git thì phải ‘pull’ source mới nhất tránh mất dữ liệu

**Integration test**

* Đẩy lên git thì phải ‘pull’ source mới nhất tránh mất dữ liệu
* Kiểm tra Premise click checkbox “Include In-active”, tất cả “Premise” hoạt động hay không hoạt động sẽ hiển thị trên list.
* kiểm tra nút Back góc trên phải màn
* Kiểm tra chức năng chuyển trang khi click 
* kiểm tra nút Save góc trên phải màn
* Kiểm tra chức năng của nút In-activate

**5.**  6.3.1 List Service



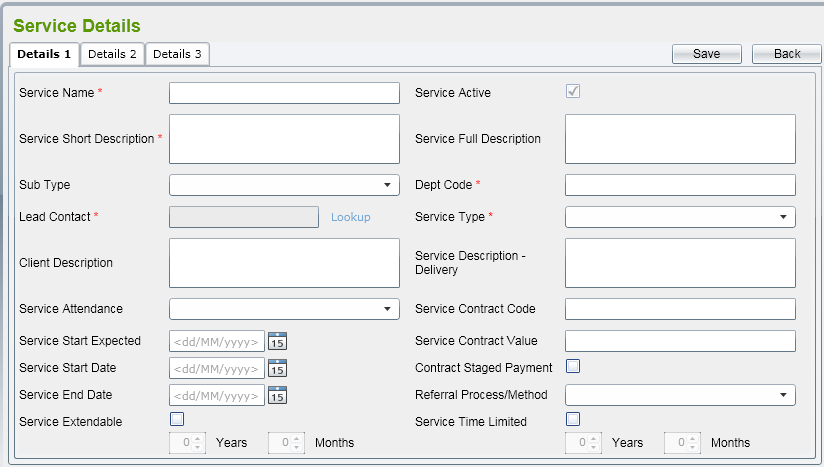
**Test Unit**

* Kiểm tra câu query lấy dữ liệu ra từ hàm getAllService đúng chưa.
* Các biến mapping với bảng đúng với cơ sở dữ liệu
* Trước khi dùng các hàm có tác động đến database phải mở luồng
* Lấy dữ liệu ra xong phải đóng luồng
* Button “Copy” có gọi đến hàm copyServices
* Button “Create” có gọi đến hàm createServices

**Integration Test**

* Kiểm tra người dùng click checkbox “Include In-active”, tất cả “Services” hoạt động hay không hoạt động sẽ hiển thị trên list.
* Kiểm tra người dùng có thể sắp xếp Service bằng cách click đến tên cột.
* Kiểm tra người dùng chọn [Services In-active] trong list, hệ thống sẽ hiển thị message “Do you want to make this Service active?”
* Đẩy lên git thì phải ‘pull’ source mới nhất tránh mất dữ liệu
* Các commit khi đẩy lên phải ghi rõ chức năng và tên người đẩy
* Tạo nhánh riêng không được code vào nhánh main

**6.** 6.3.2 Add Service



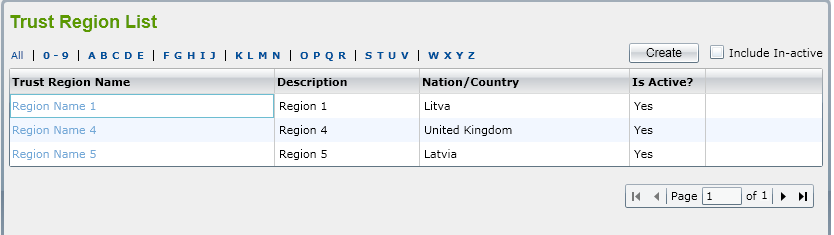
**Test Unit**

* Tên Class chứa đối tượng phải mang theo database
* Kiểu dữ liệu của biến phải phù hợp với dữ liệu tương thích trong database
* Method của save phải có kiểu trả về
* Kiểm tra điều kiện của dữ liệu trước khi thay đổi để đảm bảo toàn vẹn dữ liệu
* Màn không được chứa hàm main chỉ được chạy một main duy nhất Applications
* các biến có kiểu dữ liệu ‘Date’ phải được convert sang dạng dd/MM/yyyy

**Integration Test**

* kiểm tra trường hợp bắt buộc của input Service Name
* kiểm tra tính duy nhất của input Service Name
* kiểm tra trường bắt buộc Service Short Description
* kiểm tra trường bắt buộc Service Type
* kiểm tra trường bắt buộc Lead Contact
* kiểm tra trường bắt buộc Dept Code
* kiểm tra nút Save góc trên phải màn
* Kiểm tra nút Back góc trên phải màn
* Kiểm tra nút Save góc trên phải màn
* ‘Pull’ mới sources trước khi code đảm bảo dữ liệu luôn mới nhất
* Commit ghi rõ các chức năng update và tên người comnit

7. 6.4.1 List Trust Region



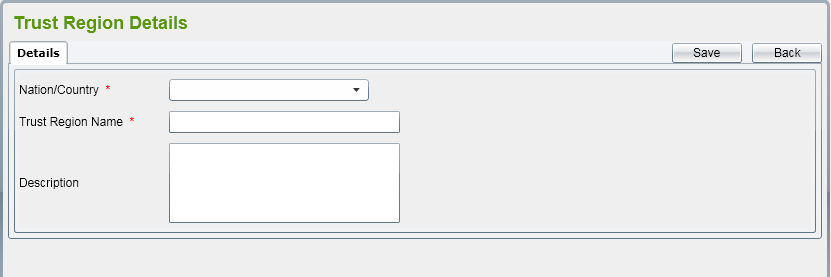
**unit test**

* kiểm tra lấy dữ liệu ra từ cơ sở dữ liệu
* Các biến mapping với bảng đúng với cơ sở dữ liệu
* Trước khi sử dụng dữ liệu trong database phải mở luồng
* Lấy dữ liệu ra xong phải đóng luồng
* Tên database trong pakage Utililties phải giống tên database dự án
* Hàm getAllTrustRegion phải là hàm trả về

**Integration Test**

* Kiểm tra user click checkbox “Include In-active”, tất cả “Trust Region” hoạt động hay không hoạt động sẽ hiển thị trên list.
* Kiểm tra người dùng có thể sắp xếp Trust Region bằng cách click đến tên cột.
* Kiểm tra user chọn [Trust Region In-active] trong list, hệ thống sẽ hiển thị message “Do you want to make this Trust Region active?”
* Đẩy lên git thì phải ‘pull’ source mới nhất tránh mất dữ liệu
* Các commit khi đẩy lên phải ghi rõ chức năng và tên người đẩy
* Tạo nhánh riêng không được code vào nhánh main

8.6.4.2 Add Trust Region



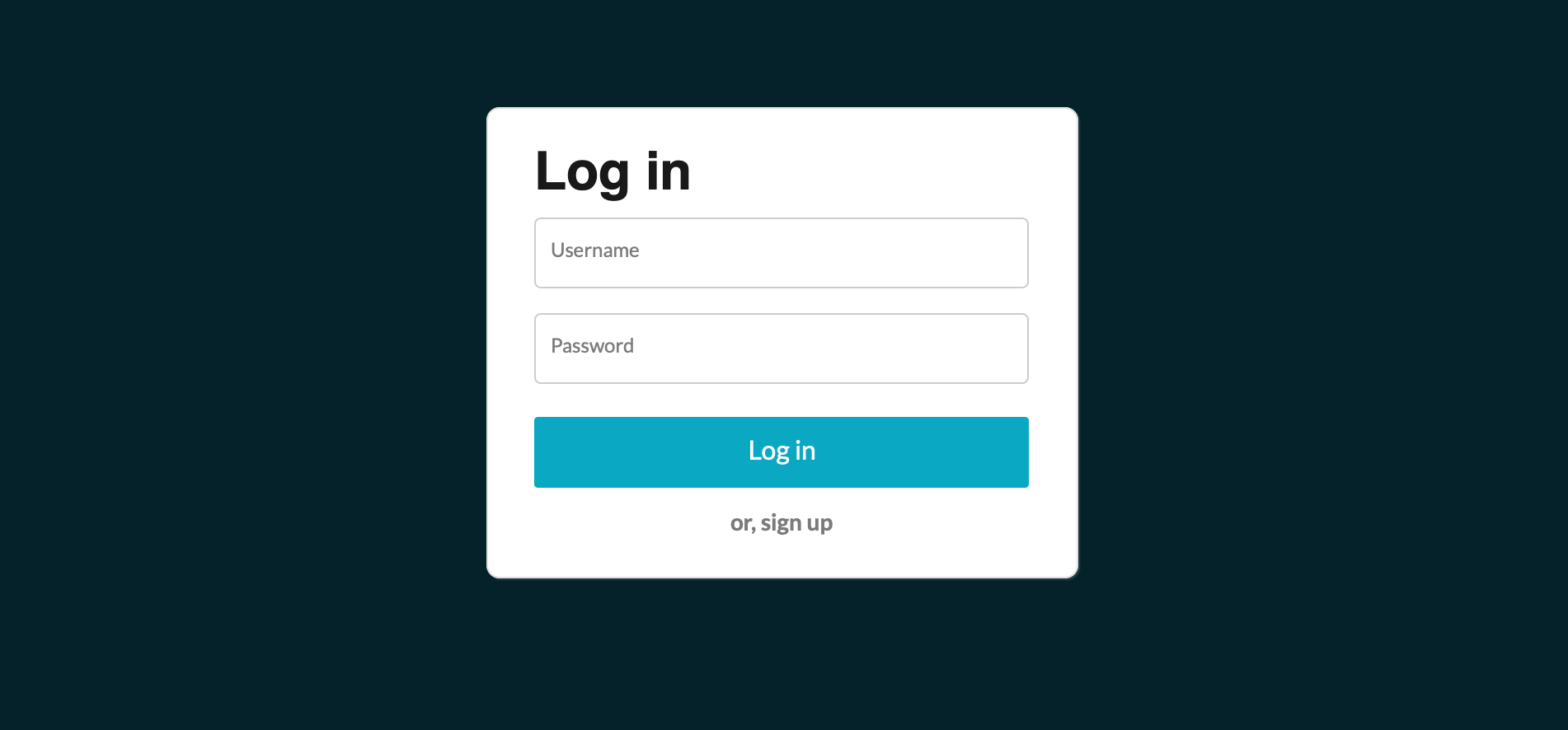
**Test Unit**

* Tên Class chứa đối tượng phải mang theo database
* Kiểu dữ liệu của biến phải phù hợp với dữ liệu tương thích trong database
* Method của Save phải có kiểu trả về
* Kiểm tra điều kiện của dữ liệu trước khi thay đổi để đảm bảo toàn vẹn dữ liệu
* Màn không được chứa hàm main chỉ được chạy một main duy nhất Applications

**Integration Test**

* kiểm tra trường hợp bắt buộc của combobox Nation/Country
* kiểm tra tính duy nhất của input Trust Region Name
* kiểm tra trường bắt buộc Description
* kiểm tra nút Back
* kiểm tra nút Save
* ‘Pull’ mới sources trước khi code đảm bảo dữ liệu luôn mới nhất
* Commit ghi rõ các chức năng update và tên người comnit

9.Chức năng Logon (mục 6.1)



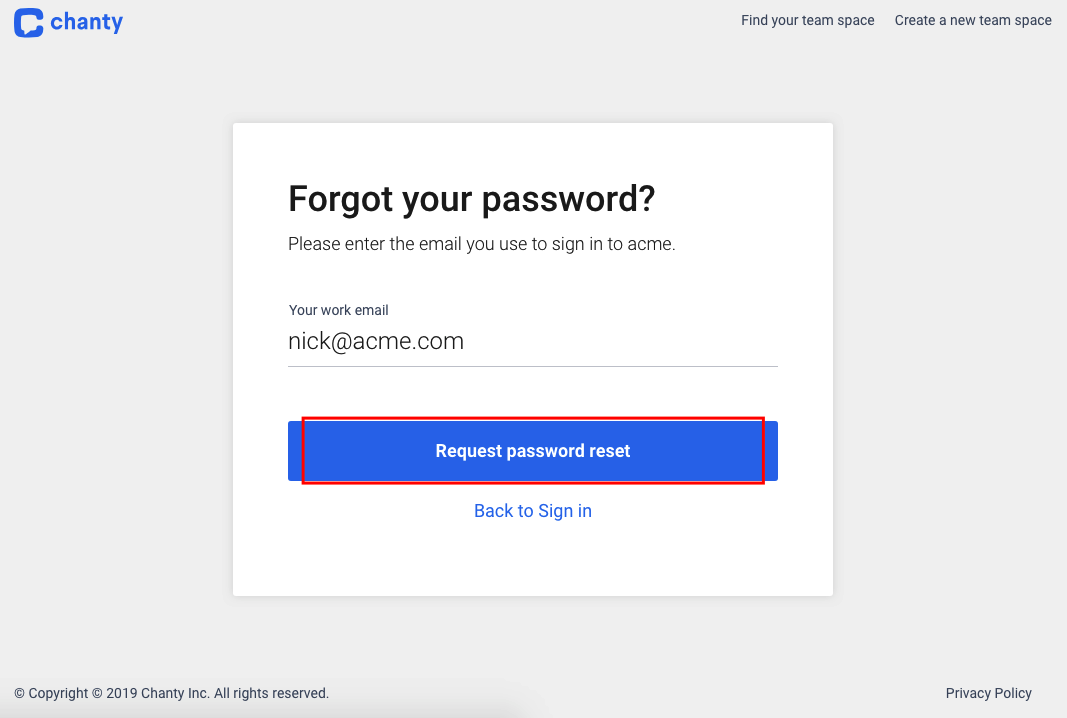
**Unit test**

* Kiểm tra tên biến đặt đúng chuẩn java.
* Hàm logonServices, ValidateUser, ValidatePass phải là hàm có trả về
* Trước khi sử dụng các hàm có tác động đến database phải mở luồng
* Sau khi lấy dữ liệu từ database xong phải đóng luồng
* Các biến phải khai báo trên cùng
* Các biến phải access modifier là Private để tăng tính bảo mật
* Button “Log in” phải gọi đến hàm logonServices
* Import class logonServices
* Import class ValidateUser và ValidatePass của package services

**Integration Test**

* Kiểm tra xem có vadile các trường user va pass
* Kiểm tra xem hàm logonServices có trả về giá trị true khi cung cấp thông tin đăng nhập hợp lệ.
* Kiểm tra xem hàm logonServices có trả về giá trị false khi cung cấp thông tin đăng nhập không hợp lệ.
* Kiểu dữ liệu của các trường “Username” và “Password” có đứng với kiểu dữ liệu trong database.
* Kiểm tra xem hàm logonServices có trả ra giá trị false khi truyền vào thông tin đăng nhập trống.
* ‘Pull’ mới sources trước khi code đảm bảo dữ liệu luôn mới nhất
* Commit ghi rõ các chức năng update và tên người comnit

10. Chức năng Forgot Password



**Unit test**

* Kiểm tra các biến có được khai báo theo đúng chuẩn java.
* Kiểm tra hàm message đã được import.
* Biến email có kiểu dữ liệu giống trong database.
* Trước khi sử dụng các hàm có tác động đến database phải mở luồng
* Sau khi lấy dữ liệu từ database xong phải đóng luồng
* Import class forgotPassword.
* Import class forgotPasswordServices trong package Services

**Integration Test**

* Kiểm tra chức năng forgotPassword (Hệ thống có gửi lại pass mới hay không)
* Button “Request password reset” có gọi hàm forgotPassword.
* Class forgotPassword phải là class trả về
* Class forgotPassword gọi đến class forgotPasswordServices
* ‘Pull’ mới sources trước khi code đảm bảo dữ liệu luôn mới nhất
* Commit ghi rõ các chức năng update và tên người comnit