**TRƯỜNG CAO ĐẲNG THỰC HÀNH**

**FPT-POLYTECHNIC**

**KIỂM THỬ NÂNG CAO**

**KIỂM THỬ DỰ ÁN BÁN ĐỒ THỂ THAO**

**Người thực hiện :**

**LÊ BÁ TRÌNH**

**VÕ THỊ THÀNH**

**NGÔ VĂN LẬP**

**Lớp : IT18302**

**Giáo viên hướng dẫn: Hải Nam**

***Tp Đà Nẵng , ngày 22 tháng 3 năm 2024.***

**MENU**

REQUIREMENT

[I. NỘI DUNG CHÍNH CỦA DỰ ÁN](#_2p2csry)

[1. Giới thiệu dự án](#_147n2zr)

[2. Cấu trúc dự án :](#_3o7alnk)

[3. Sơ đồ quan hệ thực thể](#_23ckvvd)

4.Các yêu cầu chức năng

[5. Các công cụ mã nguồn mở .](#_vx1227)

[5.1 Giới thiệu về JUnit](#_3fwokq0)

[II. TEST DESIGN](#_1v1yuxt)

[1. Tạo tài khoản và đăng nhập](#_4f1mdlm)

[2. Thêm sửa tìm kiếm sản phẩm:](#_2u6wntf)

[3. Xóa sửa User](#_19c6y18)

[4. Bảng phân chia công việc](#_3tbugp1)

[III. TEST CASE – TEST TRÊN TEST CASE](#_28h4qwu)

1. Test Login
2. Đăng ký
3. Chức năng update user của admin
4. Chức năng delete user của admin
5. Kết quả

[1 Login](#_1egqt2p)

[2 Signup](#_3ygebqi)

[3 Update](#_2dlolyb)

4 Delete

Project and database trên google drive :

1. **NỘI DUNG CHÍNH CỦA DỰ ÁN**
2. **GIỚI THIỆU DỰ ÁN**

Nhu cầu quản lí phòng gym ...

1. **Cấu trúc dự án :**

Home-Trang chủ (chọn video để xem)

Page sản phẩm – quản lí các sản phẩm , loại sản phẩm

Page đơn hàng – quản lí các đơn hàng đã mua và in ra

Page quản lí user .

1. **Sơ đồ quan hệ thực thể :**

A diagram of a computer

Description automatically generated

1. **Các yêu cầu chức năng :**

### **Login và Logout:**

Người dùng nhập Username và Password để đăng nhập vào hệ thống

Nếu người dùng quên mật khẩu, họ có thể nhấp vào liên kết 'Quên mật khẩu' trên trang Đăng nhập. Màn hình sẽ hiển thị cho phép người dùng nhập Tên đăng nhập và Email để lấy lại mật khẩu. Nếu Tên đăng nhập và Email không khớp, thông báo lỗi sẽ hiển thị 'Tên người dùng và Email không khớp'. Nếu không, hệ thống sẽ gửi mật khẩu cho đầu vào email.

1. **Chức năng quản lý dành admin.**

- Chức năng thêm video mới .

- Chức năng thay đổi thông tin video đã có .

- Chức năng xóa bỏ các video không còn hoạt động .

- Chức năng xem toàn bộ video hiện có trên database của admin.

- Chức năng tìm kiếm video

- Chức năng xem thống kê lượt xem lượt thích của từng video.

1. **Chức năng dành cho người dùng.**

- Xem các video hiển thị trên trang chủ và xem chi tiết từng video.

- Chức năng xem danh sách video đã thích hoặc đã share.

- Chức năng xem thông tin tài khoản đã đăng kí

- Chức năng đổi mật khẩu.

1. **Các công cụ mã nguồn mở .**
   1. **Giới thiệu về JUnit**
      1. **JUnit là gì ?**

Trong Java, để thực hiện viết code cho Unit Test chúng ta có thể sử dụng một trong hai Framework: [JUnit](https://junit.org/) và [TestNG](https://testng.org/).

**JUnit** là một framework mã nguồn mở, miễn phí, đơn giản dùng để unit test cho ngôn ngữ lập trình Java. Trong Java, chúng ta thường sẽ sử dụng method để làm unit test.

Chúng ta có thể sử dụng JUnit để viết code test cho cả unit testing và integration testing.

## **6.1.2 Các tính năng của JUnit**

* JUnit là một framework mã nguồn mở, được sử dụng để viết và chạy kiểm thử.
* Cung cấp các annotation để định nghĩa các phương thức kiểm thử.
* Cung cấp các Assertion để kiểm tra kết quả mong đợi.
* Cung cấp các test runner để thực thi các test script.
* Test case JUnit có thể được chạy tự động.
* Test case JUnit có thể được tổ chức thành các test suite.
* JUnit cho thấy kết quả test một cách trực quan: pass (không có lỗi) là màu xanh và fail (có lỗi) là màu đỏ.

1. **TEST DESIGN**
2. **Tạo tài khoản và đăng nhập**

Website chỉ cho người dùng like hoặc share video bằng việc tạo tài khoản người dùng trước khi muốn like hoặc share một video nào đó trên trang web .

- **Tạo tài khoản và đăng nhập trước khi like, share video**  : kiểm tra rằng video đã like và share được kết nối với tài khoản chính xác. Ngoài ra, bạn không nên bị yêu cầu phải login lại nếu bạn đã login rồi, hoặc chưa login mà bạn muốn like hoặc share thì sẽ được đưa về login .

**- Đăng nhập và Đăng xuất** : khi bạn đã đăng nhập và sau đó đăng xuất, hãy chắc chắn rằng bạn không thể truy cập vào trang tài khoản.

1. **Thêm sửa tìm kiếm video :**

**Video thích hợp** – Kiểm tra các video được hiển thị tương ứng với điều kiện tìm kiếm..

**Thông tin video**  : các video hiển thị kèm theo hình ảnh, tên.

**Thêm video của admin**: kiểm tra xem video khi được thêm vào có xuất hiện trên trang views các video của website hay không. Và video có được lưu trữ lại trong dữ liệu website hay không.

**Update video của admin** : kiểm tra sự thay đổi thông tin video có được hiện thị trên các chức năng tương ứng khác .

1. **Xóa Sửa User:**

* **Xóa:** Admin có quyền xóa bỏ tài khoảng của các user.
* **Sửa:** Admin có quyền sửabỏ tài khoảng của các user.

1. **Bảng Phân chia công việc**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table Name** | **Duration** | **Start** | **Finish** | **Predecessers** | **Resource**  **Name** |
| Thời gian phát triển dự án | 20days | 7/2/2022 | 27/3/2022 |  | Cả nhóm |
| Test login | 17days | 10/2/2022 | 27/3/2022 |  | Hoàng Duy |
| Test Sign-Up | 17days | 10/2/2022 | 27/3/2022 |  | Huy Phi |
| Test Update | 17days | 10/2/2022 | 27/3/2022 |  | Đức Duy |
| Test Delete | 17days | 10/2/2022 | 27/3/2022 |  | Danh Lợi |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

1. **TEST CASE**

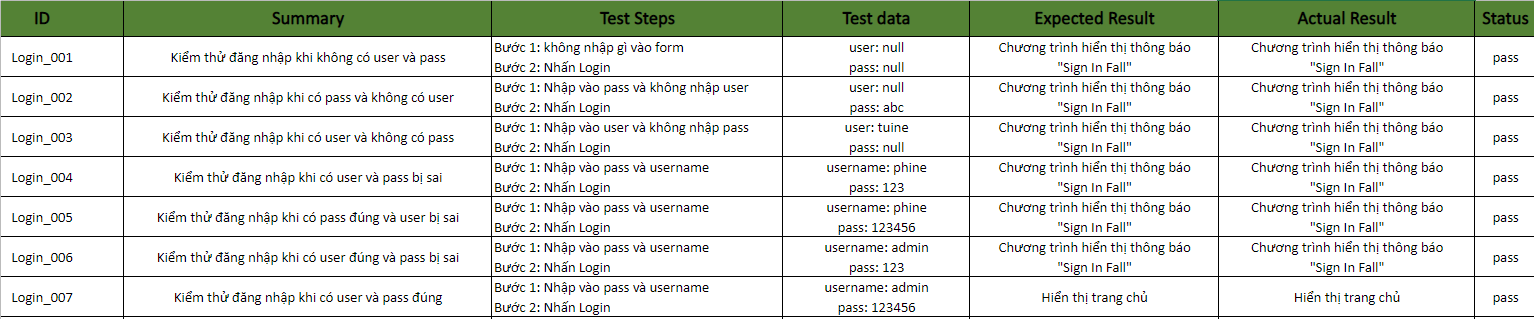
Trong Test Packages tạo ra các class test. Đề test lớp Login, Sign-Up, Update, Delete ta tạo các lớp chứa các phương thức test Test Login,Test Sign-Up,Test Update, Test Delete. Chi tiết như sau :

1. **Test Login :**

Mục đích kiểm tra: kiểm tra việc đăng nhập vào hệ thống của user thành công thì sẽ hiển thị như thế nào và không thành công thì sẽ thông báo lỗi ra sao ( admin or user)

Chức năng kiểm tra: login()

Giả định : có 2 trường là username và password. Username : **anh**; password

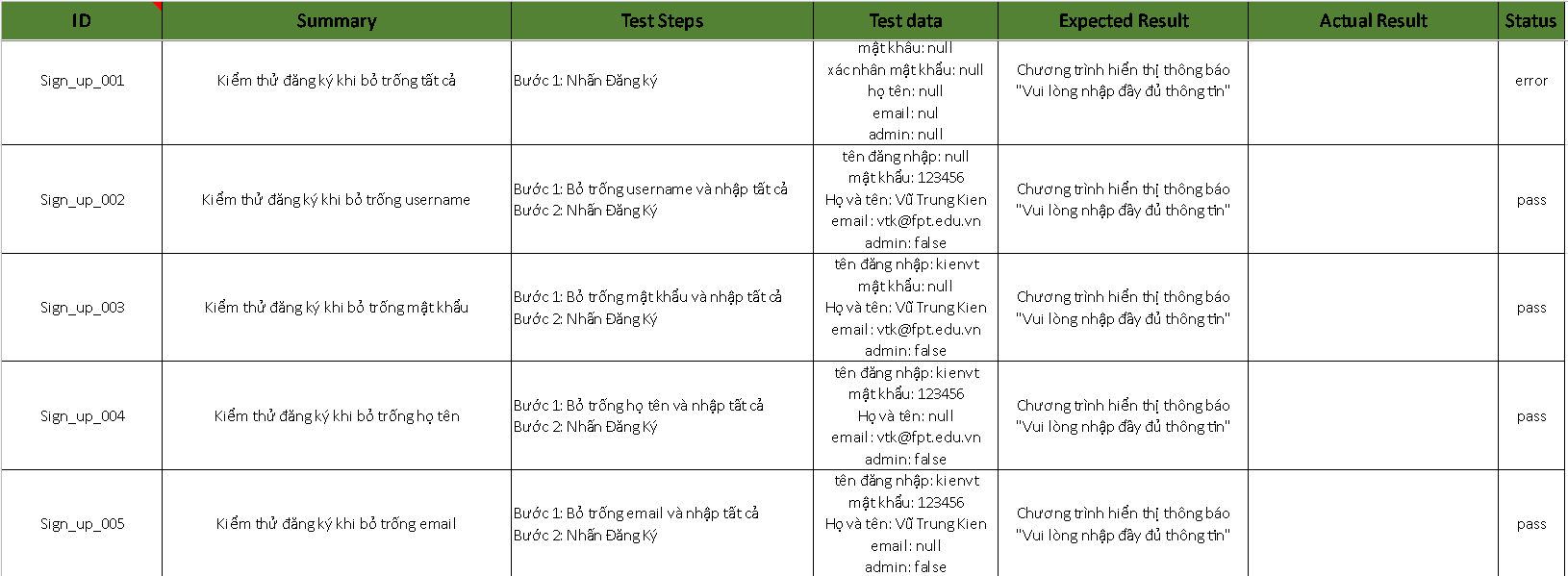


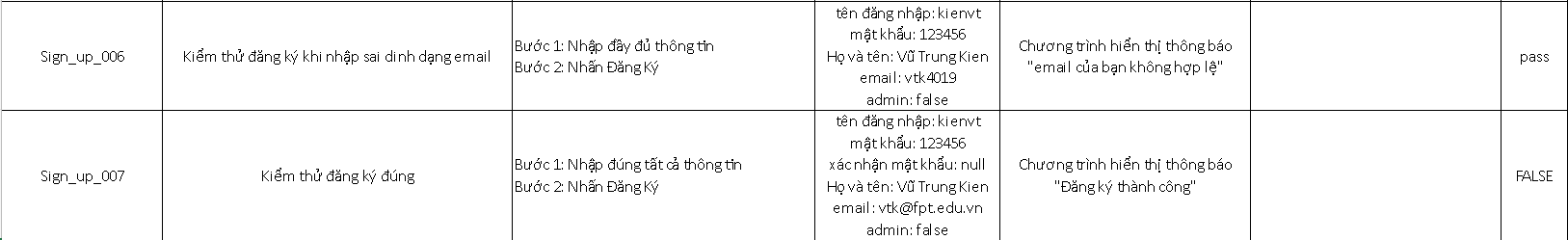
|  |
| --- |
| package testdao;  import static org.junit.Assert.\*;  import org.junit.After;  import org.junit.AfterClass;  import org.junit.Before;  import org.junit.BeforeClass;  import org.junit.Test;  import poly.dao.UserDAO;  import poly.model.User;  public class TestUpdate {  UserDAO udao = new UserDAO();  @BeforeClass  public static void OpenConnect() {  }  @Before  public void beforeUser() {  }  @Test  public void testUpdateNullusername() {  boolean t;  User uip = new User();  try {  uip.setId("");  uip.setPassword("123456");  uip.setFullname("Vũ Trung Kien");  uip.setEmail("vtk@fpt.edu.vn");  uip.setAdmin(false);  udao.update(uip);  } catch (Exception e) {  e.printStackTrace();  }  User ubf = udao.findById(uip.getId());  if (ubf != null) {  boolean temp = uip.getId().equals(ubf.getId()) && uip.getPassword().equals(ubf.getPassword())  && uip.getFullname().equals(ubf.getFullname()) && uip.getEmail().equals(ubf.getEmail())  && uip.getAdmin() == ubf.getAdmin();  t = temp;  } else {  t = false;  }  assertFalse(t);  }  @Test  public void testUpdateNullpass() {  boolean t;  User uip = new User();  try {  uip.setId("kienvt");  uip.setPassword("");  uip.setFullname("Vũ Trung Kien");  uip.setEmail("vtk@fpt.edu.vn");  uip.setAdmin(false);  udao.update(uip);  } catch (Exception e) {  e.printStackTrace();  }  User ubf = udao.findById(uip.getId());  if (ubf != null) {  boolean temp = uip.getId().equals(ubf.getId()) && uip.getPassword().equals(ubf.getPassword())  && uip.getFullname().equals(ubf.getFullname()) && uip.getEmail().equals(ubf.getEmail())  && uip.getAdmin() == ubf.getAdmin();  t = temp;  } else {  t = false;  }  assertFalse(t);  }  @Test  public void testUpdateNullname() {  boolean t;  User uip = new User();  try {  uip.setId("kienvt");  uip.setPassword("123456");  uip.setFullname("");  uip.setEmail("vtk@fpt.edu.vn");  uip.setAdmin(false);  udao.update(uip);  } catch (Exception e) {  e.printStackTrace();  }  User ubf = udao.findById(uip.getId());  if (ubf != null) {  boolean temp = uip.getId().equals(ubf.getId()) && uip.getPassword().equals(ubf.getPassword())  && uip.getFullname().equals(ubf.getFullname()) && uip.getEmail().equals(ubf.getEmail())  && uip.getAdmin() == ubf.getAdmin();  t = temp;  } else {  t = false;  }  assertFalse(t);  }  @Test  public void testUpdateNullemail() {  boolean t;  User uip = new User();  try {  uip.setId("kienvt");  uip.setPassword("123456");  uip.setFullname("Vũ Trung Kiên");  uip.setEmail("");  uip.setAdmin(false);  udao.update(uip);  } catch (Exception e) {  e.printStackTrace();  }  User ubf = udao.findById(uip.getId());  if (ubf != null) {  boolean temp = uip.getId().equals(ubf.getId()) && uip.getPassword().equals(ubf.getPassword())  && uip.getFullname().equals(ubf.getFullname()) && uip.getEmail().equals(ubf.getEmail())  && uip.getAdmin() == ubf.getAdmin();  t = temp;  } else {  t = false;  }  assertFalse(t);  }  @Test  public void testUpdateEmailError() {  boolean t;  User uip = new User();  try {  uip.setId("kienvt");  uip.setPassword("123456");  uip.setFullname("Vũ Trung Kiên");  uip.setEmail("vtk4019");  uip.setAdmin(false);  udao.update(uip);  } catch (Exception e) {  e.printStackTrace();  }  User ubf = udao.findById(uip.getId());  if (ubf != null) {  boolean temp = uip.getId().equals(ubf.getId()) && uip.getPassword().equals(ubf.getPassword())  && uip.getFullname().equals(ubf.getFullname()) && uip.getEmail().equals(ubf.getEmail())  && uip.getAdmin() == ubf.getAdmin();  t = temp;  } else {  t = false;  }  assertFalse(t);  }  @After  public void afterUser() {  }  @AfterClass  public static void CloseConnect() {  }  } |

1. **Đăng ký**

Mục đích kiểm tra: kiểm tra việc đăng ký vào hệ thống của user thành công thì sẽ hiển thị như thế nào và không thành công thì sẽ thông báo lỗi ra sao ( admin or user)

Chức năng kiểm tra: SignUp()

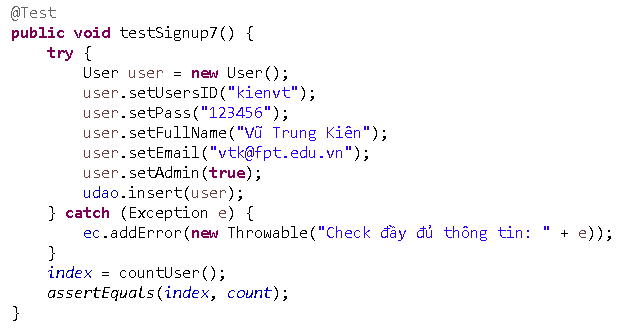




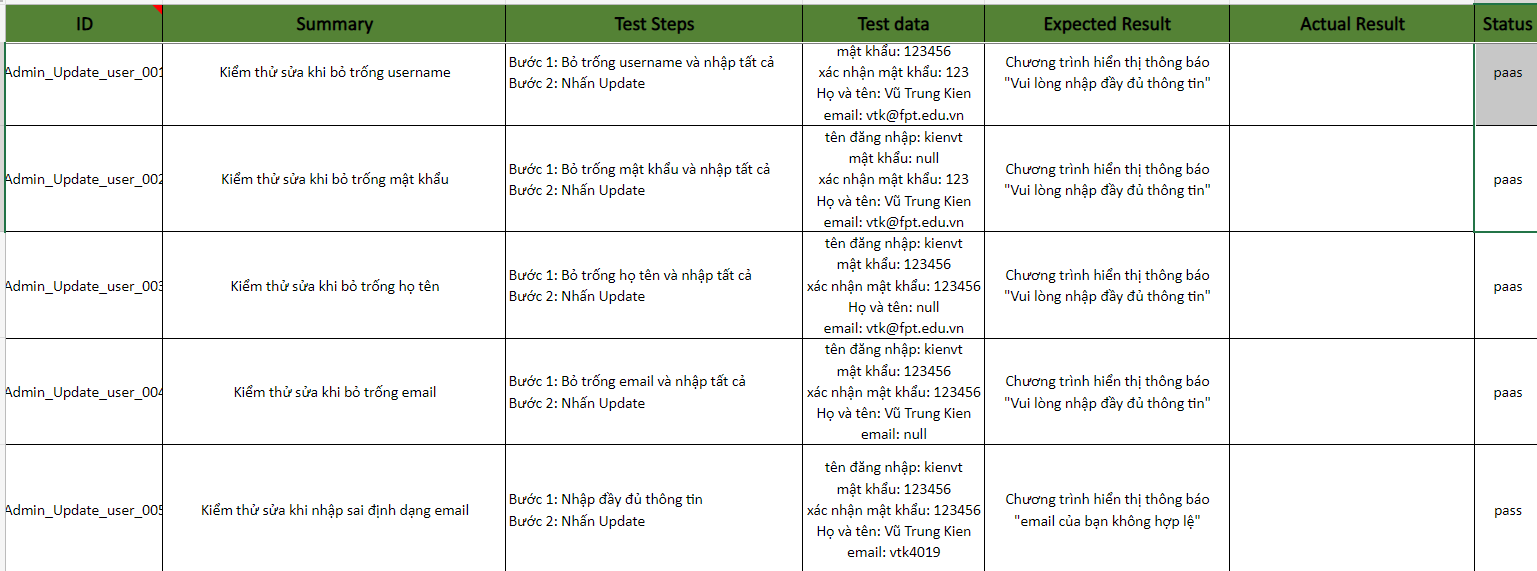








1. **Chức năng update user của admin :**

****

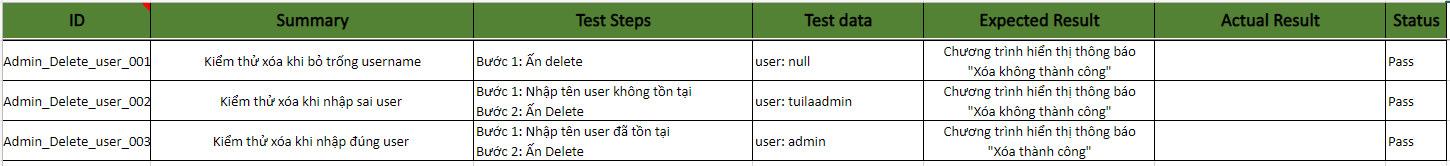
|  |
| --- |
| package testdao;  import static org.junit.Assert.\*;  import org.junit.After;  import org.junit.AfterClass;  import org.junit.Before;  import org.junit.BeforeClass;  import org.junit.Test;  import poly.dao.UserDAO;  import poly.model.User;  public class TestUpdate {  UserDAO udao = new UserDAO();  @BeforeClass  public static void OpenConnect() {  }  @Before  public void beforeUser() {  }  @Test  public void testUpdateNullusername() {  boolean t;  User uip = new User();  try {  uip.setId("");  uip.setPassword("123456");  uip.setFullname("Vũ Trung Kien");  uip.setEmail("vtk@fpt.edu.vn");  uip.setAdmin(false);  udao.update(uip);  } catch (Exception e) {  e.printStackTrace();  }  User ubf = udao.findById(uip.getId());  if (ubf != null) {  boolean temp = uip.getId().equals(ubf.getId()) && uip.getPassword().equals(ubf.getPassword())  && uip.getFullname().equals(ubf.getFullname()) && uip.getEmail().equals(ubf.getEmail())  && uip.getAdmin() == ubf.getAdmin();  t = temp;  } else {  t = false;  }  assertFalse(t);  }  @Test  public void testUpdateNullpass() {  boolean t;  User uip = new User();  try {  uip.setId("kienvt");  uip.setPassword("");  uip.setFullname("Vũ Trung Kien");  uip.setEmail("vtk@fpt.edu.vn");  uip.setAdmin(false);  udao.update(uip);  } catch (Exception e) {  e.printStackTrace();  }  User ubf = udao.findById(uip.getId());  if (ubf != null) {  boolean temp = uip.getId().equals(ubf.getId()) && uip.getPassword().equals(ubf.getPassword())  && uip.getFullname().equals(ubf.getFullname()) && uip.getEmail().equals(ubf.getEmail())  && uip.getAdmin() == ubf.getAdmin();  t = temp;  } else {  t = false;  }  assertFalse(t);  }  @Test  public void testUpdateNullname() {  boolean t;  User uip = new User();  try {  uip.setId("kienvt");  uip.setPassword("123456");  uip.setFullname("");  uip.setEmail("vtk@fpt.edu.vn");  uip.setAdmin(false);  udao.update(uip);  } catch (Exception e) {  e.printStackTrace();  }  User ubf = udao.findById(uip.getId());  if (ubf != null) {  boolean temp = uip.getId().equals(ubf.getId()) && uip.getPassword().equals(ubf.getPassword())  && uip.getFullname().equals(ubf.getFullname()) && uip.getEmail().equals(ubf.getEmail())  && uip.getAdmin() == ubf.getAdmin();  t = temp;  } else {  t = false;  }  assertFalse(t);  }  @Test  public void testUpdateNullemail() {  boolean t;  User uip = new User();  try {  uip.setId("kienvt");  uip.setPassword("123456");  uip.setFullname("Vũ Trung Kiên");  uip.setEmail("");  uip.setAdmin(false);  udao.update(uip);  } catch (Exception e) {  e.printStackTrace();  }  User ubf = udao.findById(uip.getId());  if (ubf != null) {  boolean temp = uip.getId().equals(ubf.getId()) && uip.getPassword().equals(ubf.getPassword())  && uip.getFullname().equals(ubf.getFullname()) && uip.getEmail().equals(ubf.getEmail())  && uip.getAdmin() == ubf.getAdmin();  t = temp;  } else {  t = false;  }  assertFalse(t);  }  @Test  public void testUpdateEmailError() {  boolean t;  User uip = new User();  try {  uip.setId("kienvt");  uip.setPassword("123456");  uip.setFullname("Vũ Trung Kiên");  uip.setEmail("vtk4019");  uip.setAdmin(false);  udao.update(uip);  } catch (Exception e) {  e.printStackTrace();  }  User ubf = udao.findById(uip.getId());  if (ubf != null) {  boolean temp = uip.getId().equals(ubf.getId()) && uip.getPassword().equals(ubf.getPassword())  && uip.getFullname().equals(ubf.getFullname()) && uip.getEmail().equals(ubf.getEmail())  && uip.getAdmin() == ubf.getAdmin();  t = temp;  } else {  t = false;  }  assertFalse(t);  }  @After  public void afterUser() {  }  @AfterClass  public static void CloseConnect() {  }  } |

1. **Chức năng Delete user của admin :**

Mục đích kiểm tra : kiểm tra việc xóa user

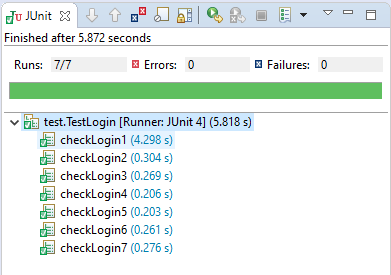
Chức năng : UserDAO.delete()

Giả định : admin đăng nhập vào hệ thống và muốn xóa user để phù hợp hơn cho việc quản lý.

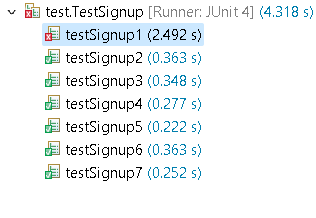
****

****

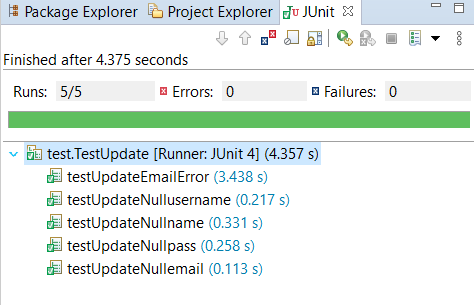
1. **Kết quả**
2. **Login:**

****

1. **Đăng ký:**



1. **Update:**

****