

TRƯỜNG ĐẠI HỌC CẦN THƠ
TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG



NIÊN LUẬN CƠ SỞ
NGÀNH MẠNG MÁY TÍNH & TRUYỀN THÔNG DỮ LIỆU

Đề tài

QUẢN LÝ CỬA HÀNG BÁN XE

Người hướng dẫn

TS/Ths Ngô Bá Hùng

Sinh viên thực hiện

Mai Tân Võ

Mã số: B2013574

Khóa: K46

Cần Thơ, 10/2023

LỜI MỞ ĐẦU

Hiện nay, lĩnh vực công nghệ thông tin ngày càng phát triển và lớn mạnh. Cùng với đó là nhu cầu sử dụng các dịch vụ trên Internet của người dùng trên các thiết bị di động, máy tính để bàn ngày một cao hơn. Đòi hỏi các lập trình viên phải có kỹ năng học tập và làm việc hiệu quả hơn, cũng như hiểu và sử dụng được công nghệ mới để xây dựng và phát triển các phần mềm có giao diện thân thiện với người dùng và hiệu năng được tối ưu. Niên luận cơ sở ngành Mạng máy tính và truyền thông dữ liệu, dưới sự giúp đỡ và hướng dẫn nhiệt tình của Thầy Ngô Bá Hùng đã giúp em hiểu biết thêm về công nghệ mới, để có một cái nhìn tổng quát hơn về lĩnh vực phát triển phần mềm, tổng hợp và vận dụng được những kiến thức đã học để xây dựng một ứng dụng hoàn thiện.

Xin chân thành cảm ơn Thầy.

MỤC LỤC

I.	Phần Giới thiệu	4
1.	ĐẶT VÂN ĐÈ	4
2.	MỤC TIÊU ĐỀ TÀI.....	4
II.	PHẦN NỘI DUNG	4
	CHƯƠNG 1. ĐẶT TẢ YÊU CẦU.....	4
1.	Sơ đồ use case	4
2.	Mô tả use case trong ứng dụng	6
3.	Các thành phần trong JavaFX	11
	CHƯƠNG 2. THIẾT KẾ GIẢI PHÁP	12
1.	Sơ đồ lớp của ứng dụng	12
2.	Sơ đồ tuần tự của ứng dụng.....	13
3.	Cách thức lưu trữ dữ liệu	17
	CHƯƠNG 3. CÀI ĐẶT GIẢI PHÁP.....	18
1.	Cài đặt chức năng “Đăng ký tài khoản”.....	18
2.	Cài đặt chức năng “Đăng nhập tài khoản”	22
3.	Cài đặt chức năng “Thống kê”	24
4.	Cài đặt chức năng “Thêm người dùng”.....	29
5.	Cài đặt chức năng “Thêm xe bán”	39
6.	Cài đặt chức năng “Thanh toán”	46
	CHƯƠNG 4. HƯỚNG DẪN CÀI ĐẶT, ĐÁNH GIÁ, KIỂM THỦ	53
1.	Hướng dẫn cài đặt	53
2.	Đánh giá, kiểm thử	53
III.	PHẦN KẾT LUẬN	54
1.	KẾT QUẢ ĐẠT ĐƯỢC.....	54
2.	HẠN CHÉ	54
3.	HƯỚNG PHÁT TRIỂN	54
IV.	TÀI LIỆU THAM THẢO	54

I. Phần Giới thiệu

1. ĐẶT VẤN ĐỀ

Sự thiếu sót trong quản lý của các cửa hàng bán xe, đặc biệt là khi so sánh với sự phát triển của công nghệ thông tin. Hiện nay, nhiều cửa hàng vẫn áp dụng cách thức quản lý truyền thống thông qua giấy tờ, gây ra những hạn chế về tính hiệu quả và tốc độ xử lý thông tin.

Vấn đề này làm tăng cường nhu cầu xây dựng một ứng dụng quản lý thông tin mua bán xe, nhằm cải thiện quá trình quản lý, lưu trữ thông tin và tối ưu hóa hiệu suất của cửa hàng. Điều này không chỉ giúp giảm sự thô sơ trong xử lý thông tin mà còn đề xuất một giải pháp hiện đại hóa để đáp ứng nhu cầu ngày càng cao trong thời đại công nghệ.

Nhằm giúp vấn đề trên được giải quyết thì việc xây dựng một ứng dụng giúp “quản lý thông tin mua bán xe” là sự cần thiết đối với những cửa xe có số lượng xe lớn để quản lý các thông tin khách hàng, xe dễ dàng, tránh rò rỉ thông tin quan trọng và cập nhật thông tin nhanh chóng, chính xác hơn.

2. MỤC TIÊU ĐỀ TÀI

Xây dựng giao diện đồ họa JavaFX nhằm hiểu được cách thức hoạt động của các thành phần ứng dụng, biết cách thiết kế bố cục giao diện hợp lý

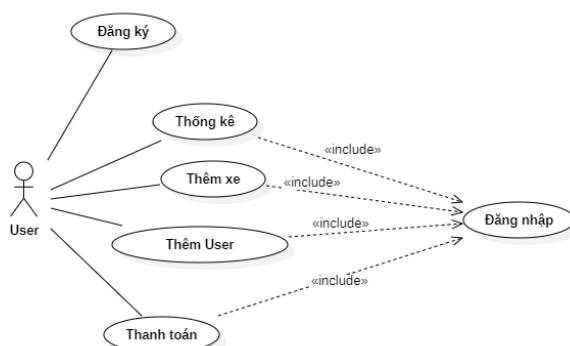
Nghiên cứu cách tạo một ứng dụng quản lý bán xe có tích hợp vào giao diện JavaFX, sử dụng cơ sở dữ liệu MySQL để lưu trữ dữ liệu

II. PHẦN NỘI DUNG

CHƯƠNG 1. ĐẶT TẨY YÊU CẦU

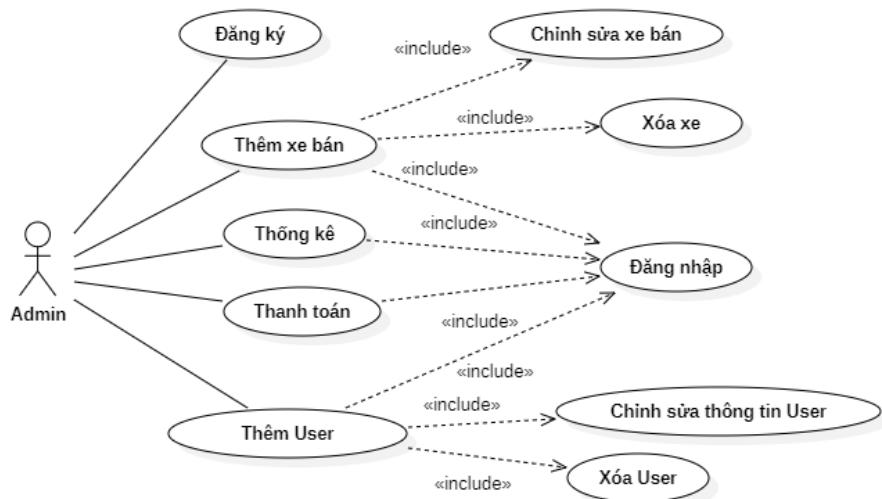
1. Sơ đồ use case

1.1. Sơ đồ use case tổng quát của ứng dụng quản lý bán xe



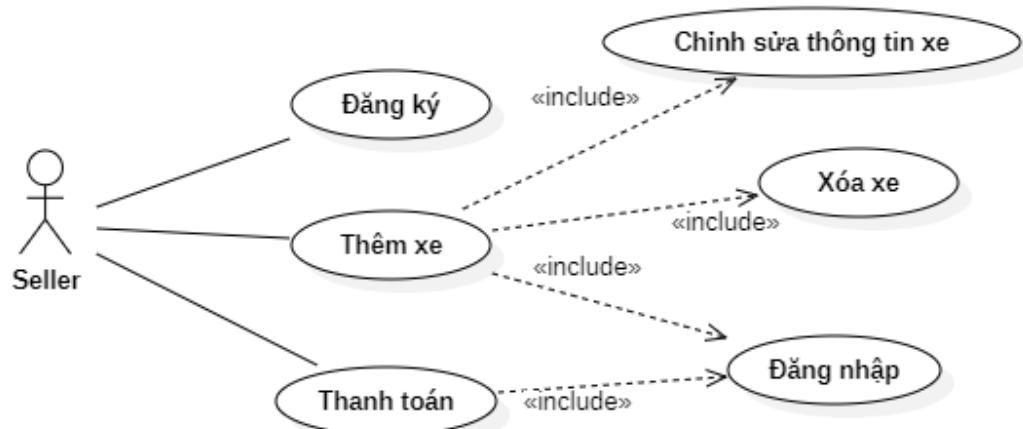
Hình 1.Sơ đồ use case tổng quát của hệ thống

1.2. Sơ đồ use case cho Admin



Hình 2. Sơ đồ use case cho Admin.

1.3. Sơ đồ use case cho seller



Hình 3. Sơ đồ use case cho seller.

2. Mô tả use case trong ứng dụng

2.1. Use case “Đăng ký tài khoản”

Chức năng “Đăng ký tài khoản” là một trong những chức năng của actor Admin và Seller. Nó cho phép người dùng đăng ký tài khoản Admin hay Seller. Các thông tin cụ thể về chức năng này bao gồm các kịch bản sử dụng được mô tả cụ thể ở trong bảng 1 phía bên dưới.

Ở chức năng đăng ký, khi ở giao diện đăng nhập người chọn chức năng để đăng ký khi chọn chức năng xong người dùng buộc phải nhập đầy đủ 3 trường email, tên người dùng và mật khẩu. Nếu thiếu hệ thống sẽ thông báo yêu cầu người dùng nhập đủ. Nếu người dùng nhấn nút đăng nhập, hệ thống sẽ quay lại cảnh đăng nhập. Nếu tên người dùng đã tồn tại thì hệ thống sẽ thông báo cho người dùng, nếu tên người dùng chưa tồn tại và hợp lệ, hệ thống sẽ thông báo người dùng phải đợi xác nhận của admin rồi mới có thể đăng nhập, admin phê duyệt người bán, nếu đã phê duyệt người bán có thể đăng nhập.

Bảng 1. Mô tả use case “Đăng ký tài khoản”.

Tên use case	Đăng ký tài khoản
Tóm tắt	Cho phép người dùng đăng ký tài khoản Admin hay Seller
Actor	Admin, Seller
Kịch bản thường	<ol style="list-style-type: none">Người dùng vào giao diện đăng nhậpChọn chức năng người bán đăng ký tài khoản chức năng đóHệ thống hiển thị giao diện đăng ký cho SellerNgười dùng nhập thông tin để đăng ký tài khoảnNgười nhấn nút đăng kýHệ thống thông báo đăng ký thành công và đợi Admin phê duyệtĐợi Admin phê duyệt và đăng nhập
Kịch bản thay thế	A1 – Người dùng nhập mật khẩu nhỏ hơn 8 Chuỗi A1 bắt đầu từ bước 4 của kịch bản thường. 5. Hệ thống thông báo mật khẩu nhỏ hơn 8 yêu cầu nhập lại. 6. Quay lại bước 2 trong kịch bản thường A2 – Tài khoản người dùng đăng ký đã được tạo Chuỗi A2 bắt đầu từ bước 4 của kịch bản thường 5. Hệ thống thông báo người dùng đã được tạo yêu cầu nhập lại. 6. Quay lại bước 2 của kịch bản thường A3 – Người dùng chọn chức năng Admin Chuỗi A3 bắt đầu từ bước 1 của kịch bản thường 1. Chọn chức năng Admin đăng ký tài khoản Admin

	2. Hệ thống hiển thị giao diện đăng ký cho Admin 3. Người dùng nhập thông tin vào 4. Nhấn nút đăng 5. Hệ thống thông báo đăng ký thành công 6. Người dùng nhấn nút đăng nhập
Kết quả	Người dùng đăng ký tài khoản thành công

2.2. Use case “Đăng nhập tài khoản”

Chức năng “Đăng nhập tài khoản” là một trong những chức năng của actor Admin và Seller. Nó cho phép người dùng đăng nhập tài khoản. Các thông tin cụ thể về chức năng này bao gồm các kịch bản sử dụng được mô tả cụ thể ở trong bảng 2 phía bên dưới.

Ở chức năng đăng nhập, người dùng nhập vào username và password. Nếu nhấn nút login, hệ thống sẽ kiểm tra nếu username là hợp lệ và đúng password trong cơ sở dữ liệu, hệ thống sẽ chuyển người dùng đến màn hình chính. Nếu người dùng nhập sai thông tin username hoặc password, hệ thống sẽ thông báo yêu cầu người dùng nhập lại thông tin. Nếu người dùng chưa có tài khoản thì nhấn vào nút Register, hệ thống sẽ chuyển người dùng đến giao diện đăng ký.

Bảng 2. Mô tả use case “Đăng nhập tài khoản”. Use case “Thống kê”

Tên use case	Đăng nhập tài khoản
Tóm tắt	Cho phép Admin hay Seller đăng nhập tài khoản
Actor	Admin, Seller
Kịch bản thường	1. Người dùng vào giao diện đăng nhập 2. Người dùng nhập thông tin username và password 3. Người dùng nhấn nút đăng nhập 4. Hệ kiểm tra cơ sở dữ liệu hợp lệ và tài khoản thuộc chức năng Seller 5. Hệ thống đưa người dùng vào giao diện của Seller
Kịch bản thay thế	A1 – Người dùng nhập sai thông tin Chuỗi A1 bắt đầu ở bước 3 của kịch bản thường 4. Hệ thống kiểm tra không hợp lệ 5. Hệ thống thông báo thông tin sai yêu cầu nhập lại Quay lại bước 2 của kịch bản thường A2 – Người dùng đăng nhập tài khoản Admin Chuỗi A2 bắt đầu ở bước 3 của kịch bản thường

	4. Hệ thống kiểm tra cơ sở dữ liệu hợp lệ và tài khoản thuộc chức năng Admin 5. Hệ thống đưa người dùng vào giao diện của Admin
Kết quả	Người dùng đăng nhập tài khoản thành công

2.3. Use case “Thêm xe bán”

Chức năng “Thêm xe bán” là một trong những chức năng của Actor Admin và Seller. Nó cho phép người dùng thêm xe vào kho dữ liệu để quản lý. Các thông tin cụ thể về chức năng này bao gồm các kịch bản sử dụng được mô tả cụ thể ở trong bảng 3 ở phía dưới.

Ở chức năng thêm xe bán, hệ thống hiển thị danh sách xe đã được thêm vào cơ sở dữ liệu. Người dùng có thể thêm thông tin xe mới vào cơ sở dữ liệu hay có thể chỉnh sửa thông tin hay xóa xe ra khỏi cơ sở dữ liệu.

Bảng 3. Mô tả use case “Thêm xe bán”

Tên use case	Thêm xe bán
Tóm tắt	Cho phép người dùng thêm xe vào cơ sở dữ liệu
Actor	Admin, Seller
Kịch bản thường	1. Hiển thị giao diện thêm xe bán 2. Người dùng nhập thông tin xe bán 3. Người dùng nhấn nút thêm 4. Hệ thống thông báo thêm thành công 5. Hiển thị danh sách những xe đã được thêm
Kịch bản thay thế	A1 – Người dùng chỉnh sửa thông tin Chuỗi A1 bắt đầu ở bước 1 của kịch bản thường 2. Người dùng chọn xe cần chỉnh sửa trên danh sách 3. Nhập thông tin cần chỉnh sửa 4. Nhấn nút cập nhật 5. Hệ thống thông báo cập nhật thành công Quay lại bước 5 của kịch bản thường A2 – Người dùng xóa xe ra khỏi kho Chuỗi A2 bắt đầu ở bước 1 của kịch bản thường 2. Chọn xe cần xóa trên danh sách 3. Nhấn nút xóa 4. Hệ thống thông báo xóa thành công

	Quay lại bước 5 của kịch bản thường
Kết quả	Người dùng thêm, chỉnh sửa, xóa xe trong kho dữ liệu thành công

2.4. Use case “Thêm người dùng”

Tên use case	Thêm người dùng
Tóm tắt	Cho phép thêm người dùng và phê duyệt cho seller
Actor	Admin
Kịch bản thường	<ol style="list-style-type: none"> 1. Hệ thống hiển thị giao diện thêm người dùng 2. Hiển thị danh sách người dùng 3. Nhấn nút thêm người dùng 4. Hiển thị giao diện thêm 5. Nhập thông tin người dùng 6. Nhấn nút thêm 7. Hệ thống thông báo thêm thành công 8. Nhấn nút thoát 9. Hiển thị danh sách người dùng
Kịch bản thay thế	<p>A1 – Người dùng cập nhật thông tin tài khoản Chuỗi A1 bắt đầu ở bước 2 của kịch bản thường</p> <ol style="list-style-type: none"> 3. Chọn người dùng cần chỉnh sửa thông tin 4. Nhấn nút cập nhật 5. Hệ thống hiển thị giao diện cập nhật 6. Chính sửa thông tin 7. Nhấn nút cập nhật 8. Hệ thống thông báo cập nhật thành công <p>Quay lại bước 8 của kịch bản thường</p> <p>A2 – Người dùng xóa người dùng khác Chuỗi A2 bắt đầu ở bước 2 của kịch bản thường</p> <ol style="list-style-type: none"> 3. Chọn người dùng cần xóa 4. Nhấn nút xóa 5. Hệ thống thông báo xóa thành công
Kết quả	Người dùng thêm, sửa, xóa người dùng thành công

2.5. Use case “Thanh toán”

Tên use case	Thanh toán
Tóm tắt	Cho phép lưu thông tin khách hàng đã mua xe
Actor	Admin, Seller
Kịch bản thường	<ol style="list-style-type: none"> 1. Hiện thị giao diện thanh toán 2. Nhập thông tin khách hàng 3. Nhấn nút thêm 4. Hệ thống thông báo thêm thành công
Kịch bản thay thế	<p>A1 – Người dùng liệt kê thông tin của khách hàng đã mua xe Chuỗi A1 bắt đầu ở bước 2 của kịch bản thường</p> <ol style="list-style-type: none"> 3. Nhấn nút list 4. Hệ thống hiển thị danh sách khách hàng <p>A2 – Người dùng xóa thông tin của khách hàng Chuỗi A2 bắt đầu ở bước 4 của kịch bản thay thế A1</p> <ol style="list-style-type: none"> 5. Chọn thông tin cần xóa 6. Nhấn nút xóa 7. Hệ thống thông báo xóa thành công 8. Hiển thị danh sách khách hàng
Kết quả	Người dùng có thể tìm kiếm, thêm, xóa thông tin khách hàng

2.6. Use case “Thống kê”

Tên use case	Thống kê
Tóm tắt	Cho phép hiển thị doanh thu của cửa hàng
Actor	Admin
Kịch bản thường	<ol style="list-style-type: none"> 1. Hiện thị giao diện chính 2. Hệ thống hiển thị những doanh thu của cửa hàng
Kịch bản thay thế	Không có kịch bản thay thế
Kết quả	Người dùng nhận được doanh thu của cửa hàng

3. Các thành phần trong JavaFX

3.1. Stage (Sân khấu).

Giai đoạn (một cửa sổ) chứa tất cả các đối tượng của ứng dụng JavaFX. Nó được đại diện bởi lớp Stage của gói javafx.stage. Stage chính được tạo bởi chính nền tảng của nó. Đối tượng stage đã tạo sẽ được truyền như một đối số cho phương thức start() của lớp Application.

Mỗi sân khấu có hai tham số xác định vị trí của nó là Width và Height. Nó được chia thành Content Area (Khu vực nội dung) và Decoration (Trang trí).

Có 5 loại sân khấu có sẵn: trang trí, chưa trang trí, trong suốt, thống nhất, tiện ích.

Phải gọi phương thức show() để hiển thị nội dung của một vùng.

3.2. Scene (Bối cảnh).

Một bối cảnh tượng trưng cho nội dung vật lý của một ứng dụng JavaFX. Nó chứa tất cả các nội dung của một biểu đồ cảnh. Lớp Scene của gói javafx.scene đại diện cho đối tượng scene. Tại một thời điểm, đối tượng scene chỉ được thêm vào một giai đoạn.

Tạo một cảnh bằng cách khởi tạo lớp Scene, có thể chọn cách thước của cảnh bằng cách truyền kích thước của nó (chiều cao và chiều rộng) cùng với nút gốc tới hàm tạo của nó.

3.3. Scene Graph (Đồ thị cảnh).

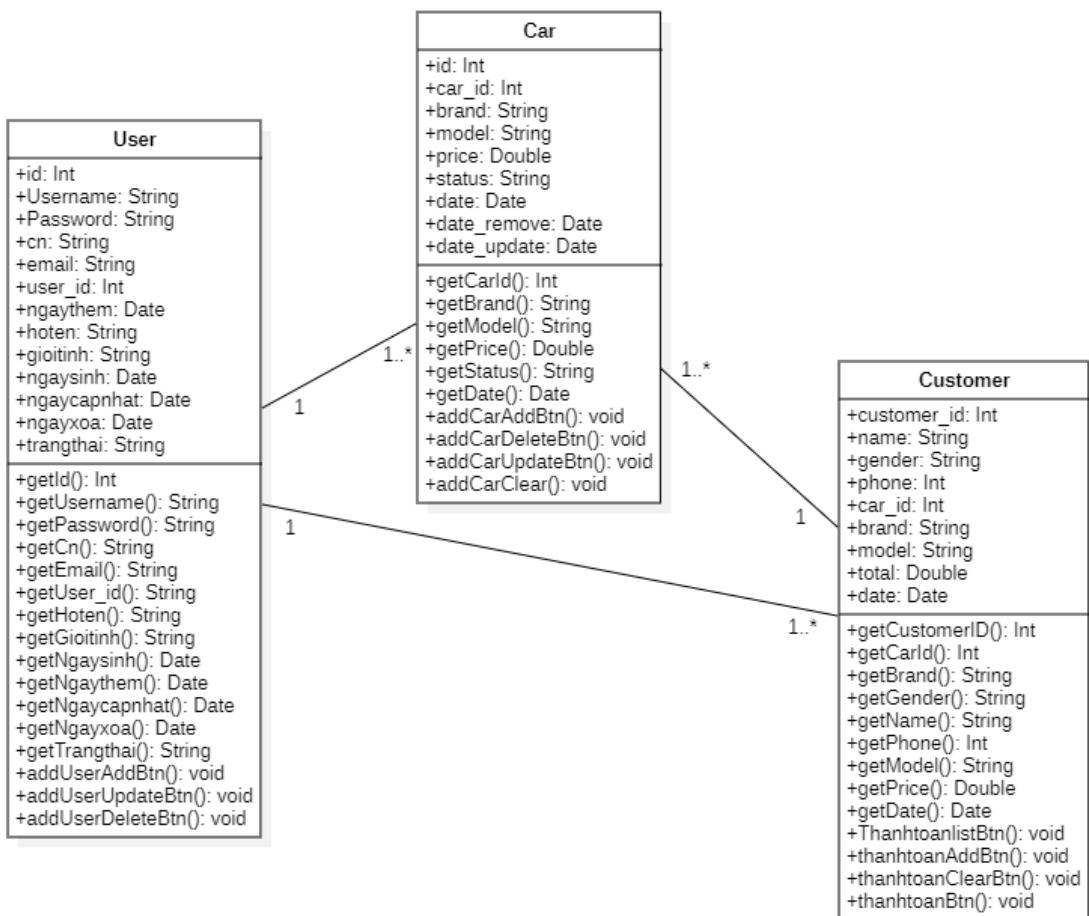
Là cơ sở của giao diện người dùng trong JavaFX, được sử dụng để xây dựng và tổ chức các phần tử UI (User Interface). Đồ thị cảnh sử dụng các nút (nodes) và mối quan hệ cha-con để hiển thị giao diện người dùng.

3.4. Phần tử UI (UI Controls).

JavaFX cung cấp nhiều phần tử điều khiển giao diện người dùng như nút (Button), ô văn bản (TextField), hộp chọn (ComboBox), danh sách (ListView), bảng (TableView), và nhiều phần tử khác để tương tác với người dùng.

CHƯƠNG 2. THIẾT KẾ GIẢI PHÁP

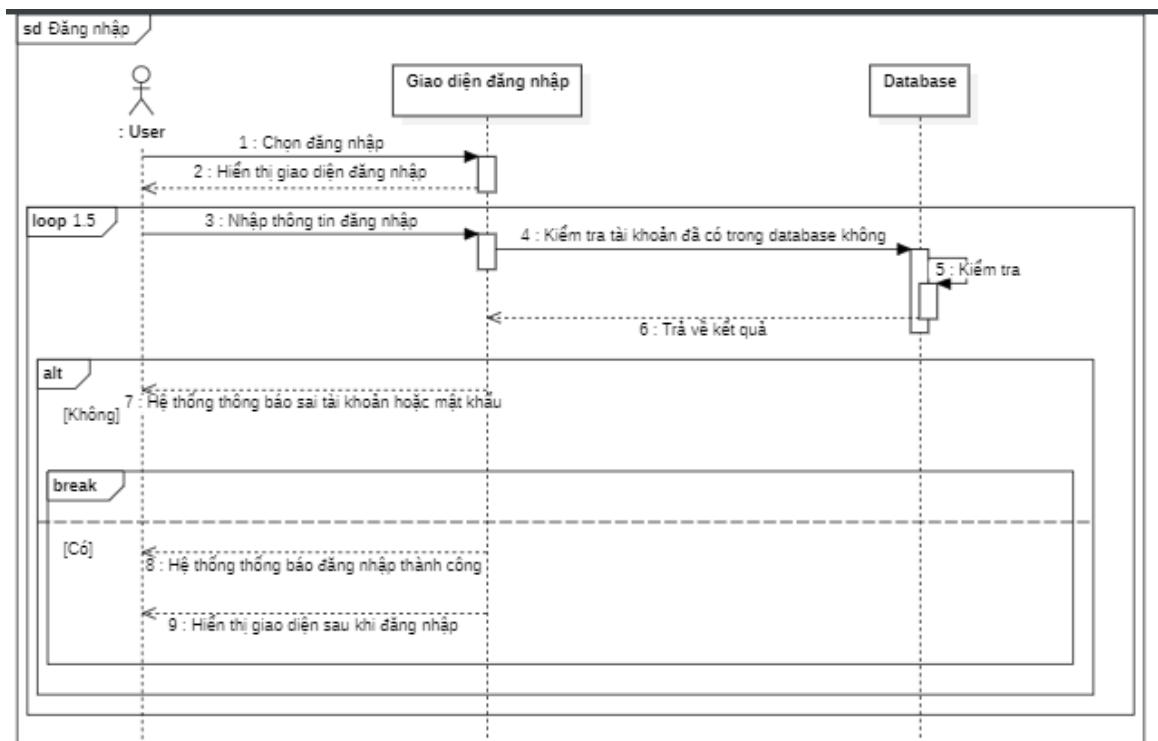
1. Sơ đồ lớp của ứng dụng



Hình 4. Sơ đồ lớp của ứng dụng.

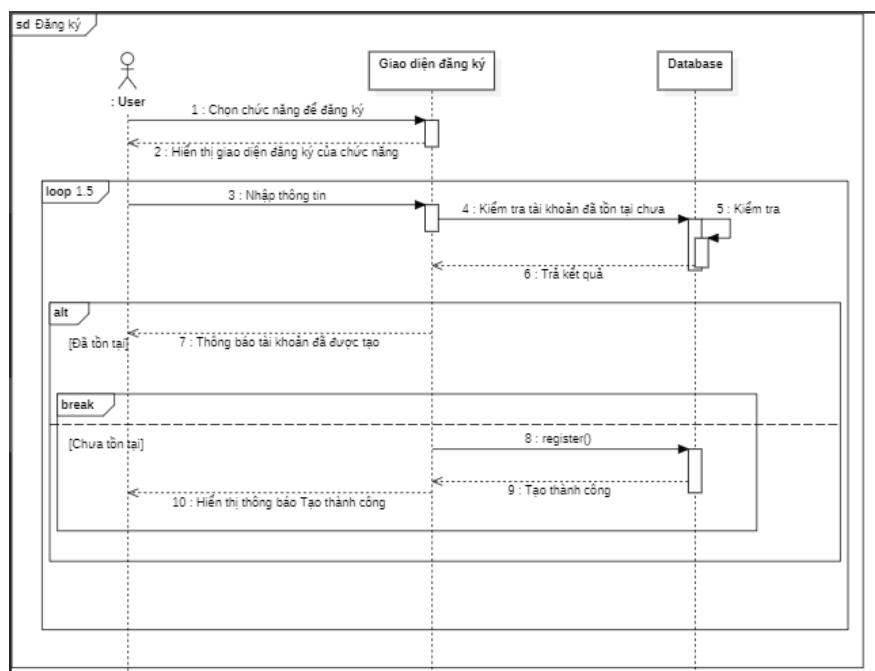
2. Sơ đồ tuần tự của ứng dụng

2.1. Sơ đồ tuần tự “Đăng nhập”



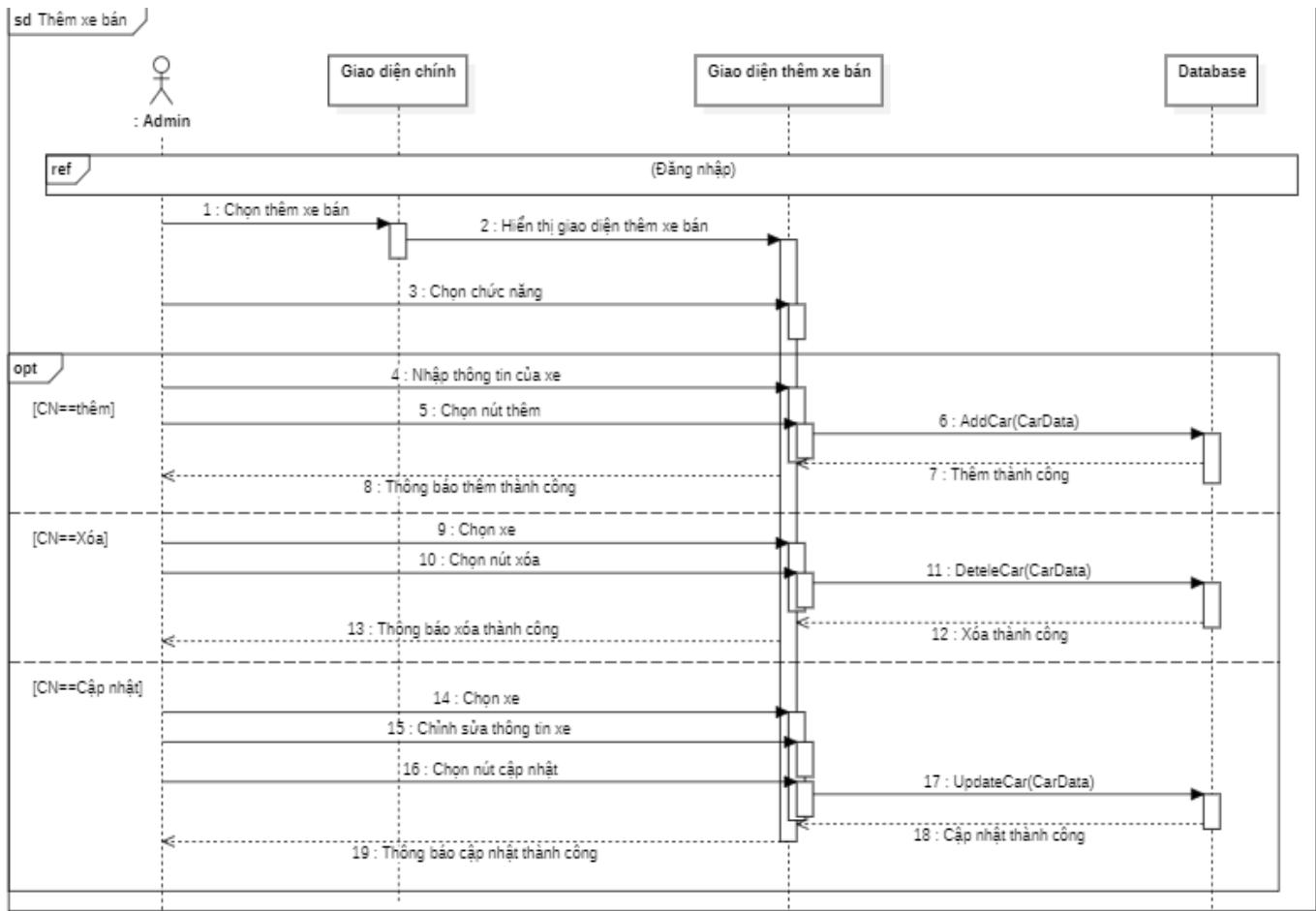
Hình 5. Sơ đồ tuần tự đăng nhập.

2.2. Sơ đồ tuần tự “Đăng ký”

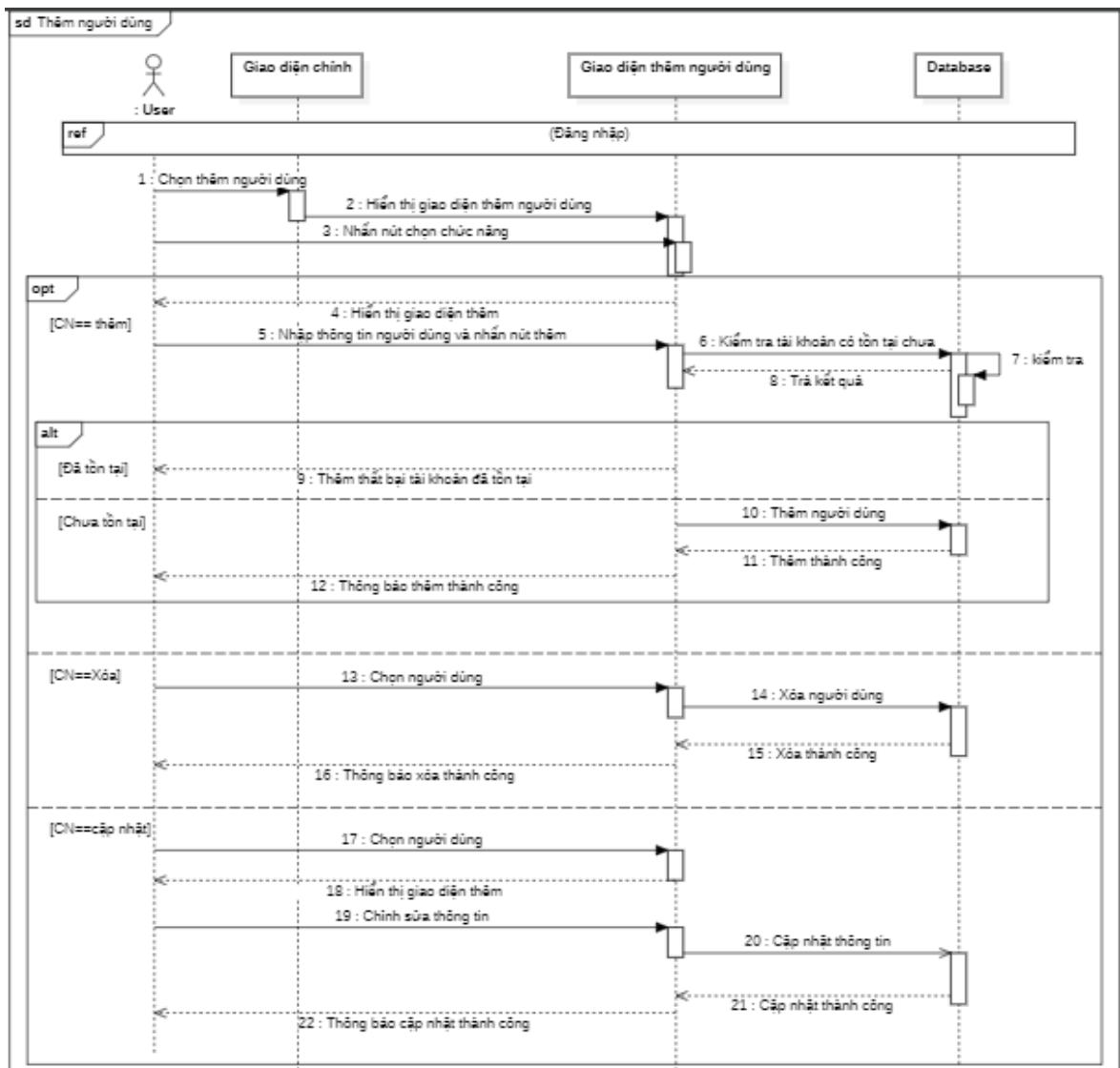


Hình 6. Sơ đồ tuần tự đăng ký

2.3. Sơ đồ tuần tự “Thêm xe bán”

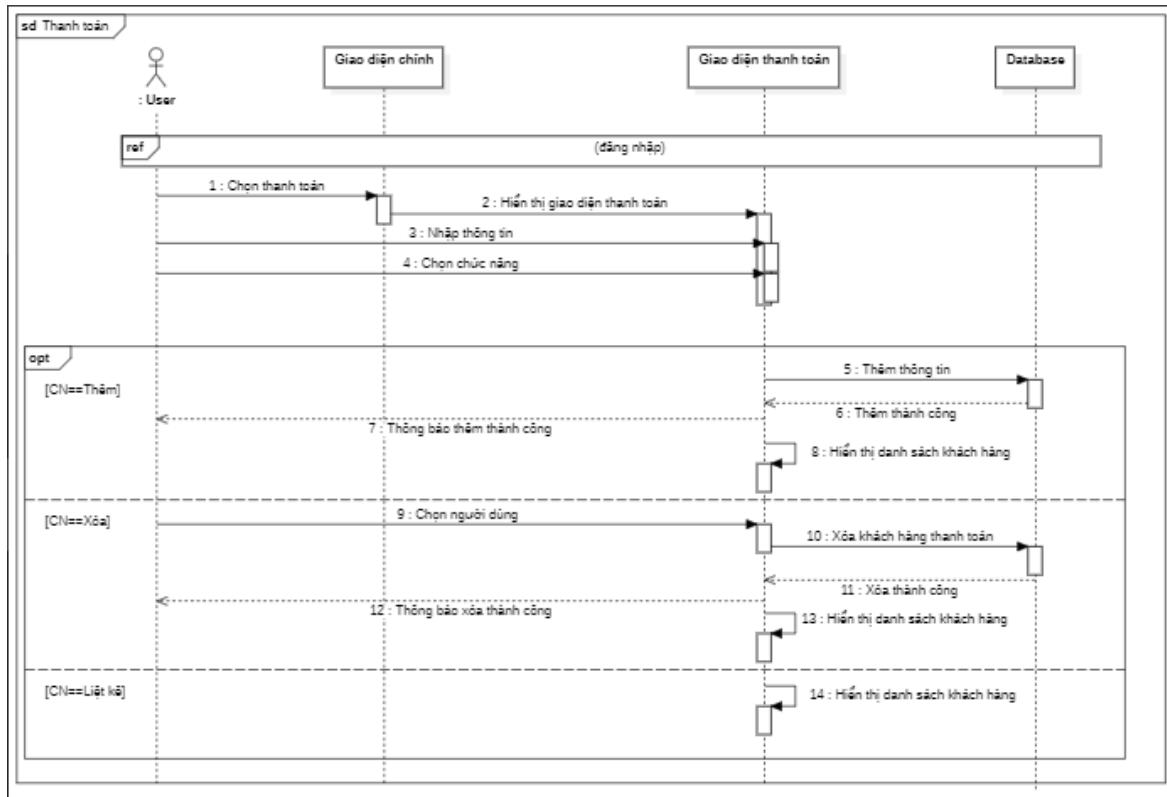


2.4. Sơ đồ tuần tự “Thêm người dùng”



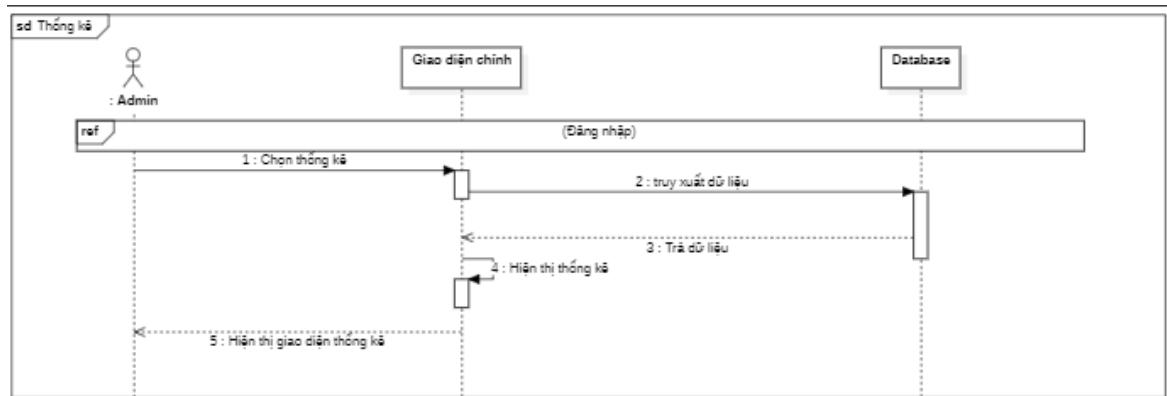
Hình 8. Sơ đồ tuần tự thêm người dùng

2.5. Sơ đồ tuần tự “Thanh toán”



Hình 9. Sơ đồ tuần tự thanh toán

2.6. Sơ đồ tuần tự “Thống kê”



Hình 10. Sơ đồ tuần tự thống kê

3. Cách thức lưu trữ dữ liệu

3.1. Hệ quản trị cơ sở dữ liệu MySQL

MySQL là hệ quản trị cơ sở dữ liệu tự do nguồn mở phổ biến nhất thế giới và được các nhà phát triển rất ưa chuộng trong quá trình phát triển ứng dụng. Vì MySQL là hệ quản trị cơ sở dữ liệu tốc độ cao, ổn định và dễ sử dụng, có tính khả chuyển, hoạt động trên nhiều hệ điều hành cung cấp một hệ thống lớn các hàm tiện ích rất mạnh. Với tốc độ và tính bảo mật cao, MySQL rất thích hợp cho các ứng dụng có truy cập CSDL trên Internet.

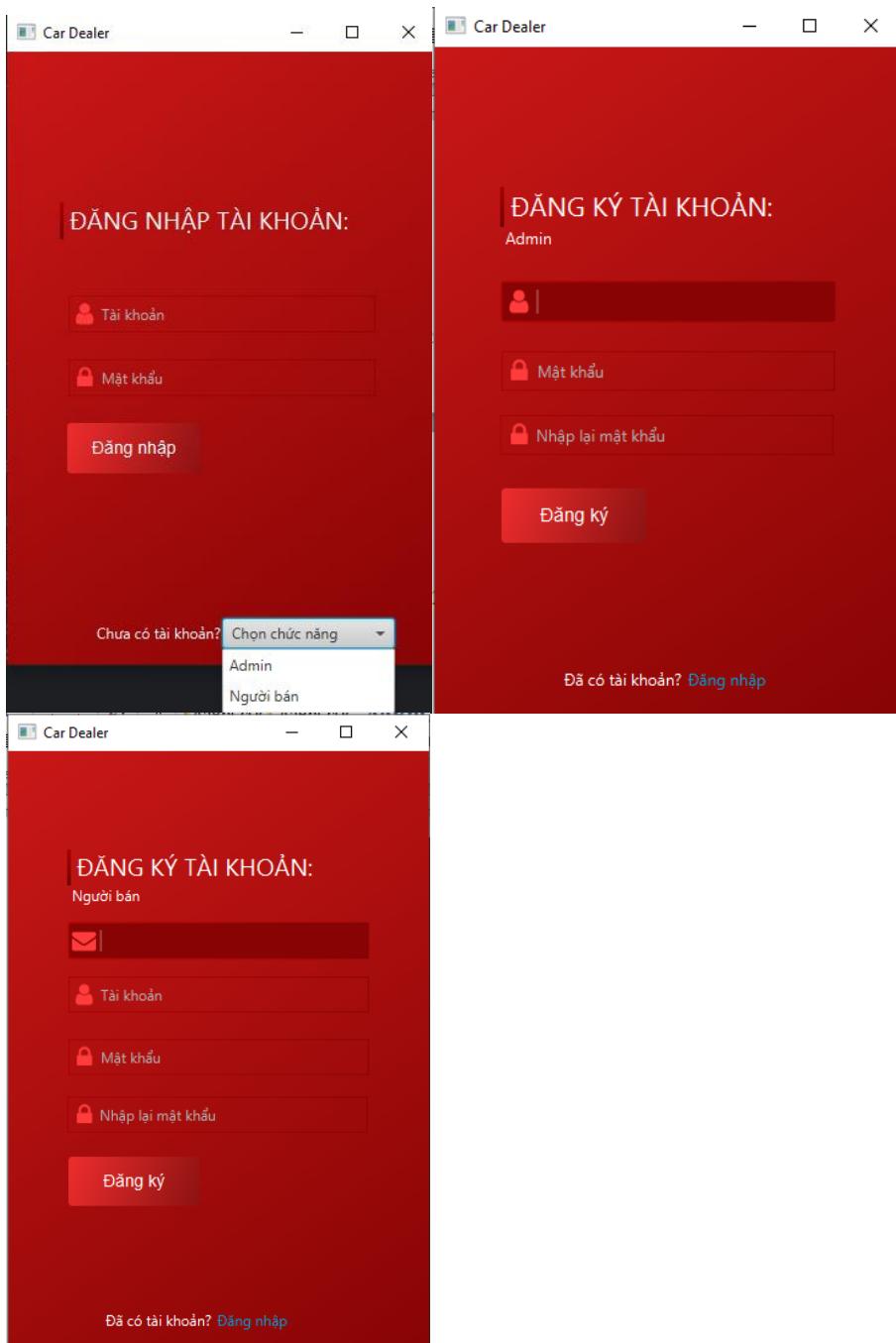
Sử dụng JDBC để truy vấn dữ liệu từ cơ sở dữ liệu

Java JDBC là một Java API được sử dụng để kết nối và thực hiện truy vấn với cơ sở dữ liệu. JDBC API sử dụng trình điều khiển jdbc để kết nối với cơ sở dữ liệu

CHƯƠNG 3. CÀI ĐẶT GIẢI PHÁP

1. Cài đặt chức năng “Đăng ký tài khoản”.

1.1. Giao diện đăng ký.



Hình 11. Giao diện đăng ký chọn chức năng

Hình 12. Giao diện đăng ký tài khoản Admin

Hình 13. Giao diện đăng ký tài khoản Người bán

1.2. Xử lý các sự kiện khi nhấn nút Register.

1.2.1. Các sự kiện xử lý đăng ký tài khoản admin.

```
public void registerAdmin(){
    if(admin_username.getText().isEmpty() || admin_password.getText().isEmpty() || admin_cpassword.getText().isEmpty()){
        alert.thatbai(tinhan: "Vui lòng điền đầy đủ thông tin không để trống");
    }else {
        connect = Database.connectDB();

        String selectData = "SELECT * FROM Users WHERE username = '" + admin_username.getText() + "'";

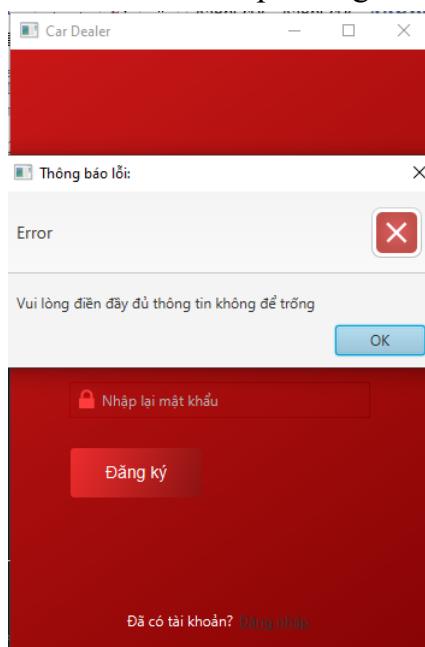
        try {
            statement = connect.createStatement();
            result = statement.executeQuery(selectData);
            if (result.next()) {
                alert.thatbai(tinhan: "Đã tồn tại");
            } else if (!admin_password.getText().equals(admin_cpassword.getText())) {
                alert.thatbai(tinhan: "Mật khẩu không trùng khớp vui lòng nhập lại");
            } else if (admin_password.getText().length() < 8) {
                alert.thatbai(tinhan: "Mật khẩu quá ngắn ít nhất phải hơn 8 kí tự");
            } else {
                String insertData = "INSERT INTO Users (username, password, cn, date)" + "VALUES(?, ?, ?, ?)";
                Date date = new Date();
                java.sql.Date sqlDate = new java.sql.Date(date.getTime());
                prepare = connect.prepareStatement(insertData);
                prepare.setString(parameterIndex: 1, admin_username.getText());
                prepare.setString(parameterIndex: 2, admin_password.getText());
                prepare.setString(parameterIndex: 3, x: "Admin");
                prepare.setString(parameterIndex: 4, String.valueOf(sqlDate));

                prepare.executeUpdate();
                alert.thanhcong(tinhan: "Đăng ký thành công!");

                login_form.setVisible(true);
                admin_form.setVisible(false);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

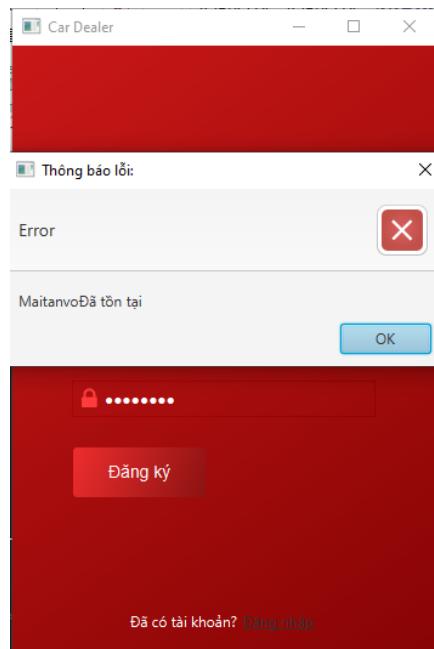
Hình 14. Sự kiện xử lý đăng ký tài khoản Admin

- Giao diện khi nhập không đủ thông tin



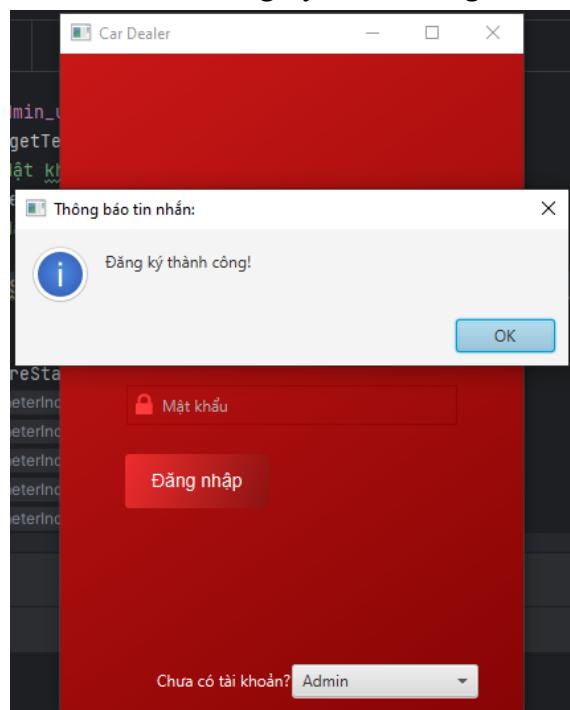
Hình 15. Giao diện thông báo khi nhập không đủ thông tin

- Giao diện khi tạo tài khoản đã tồn tại



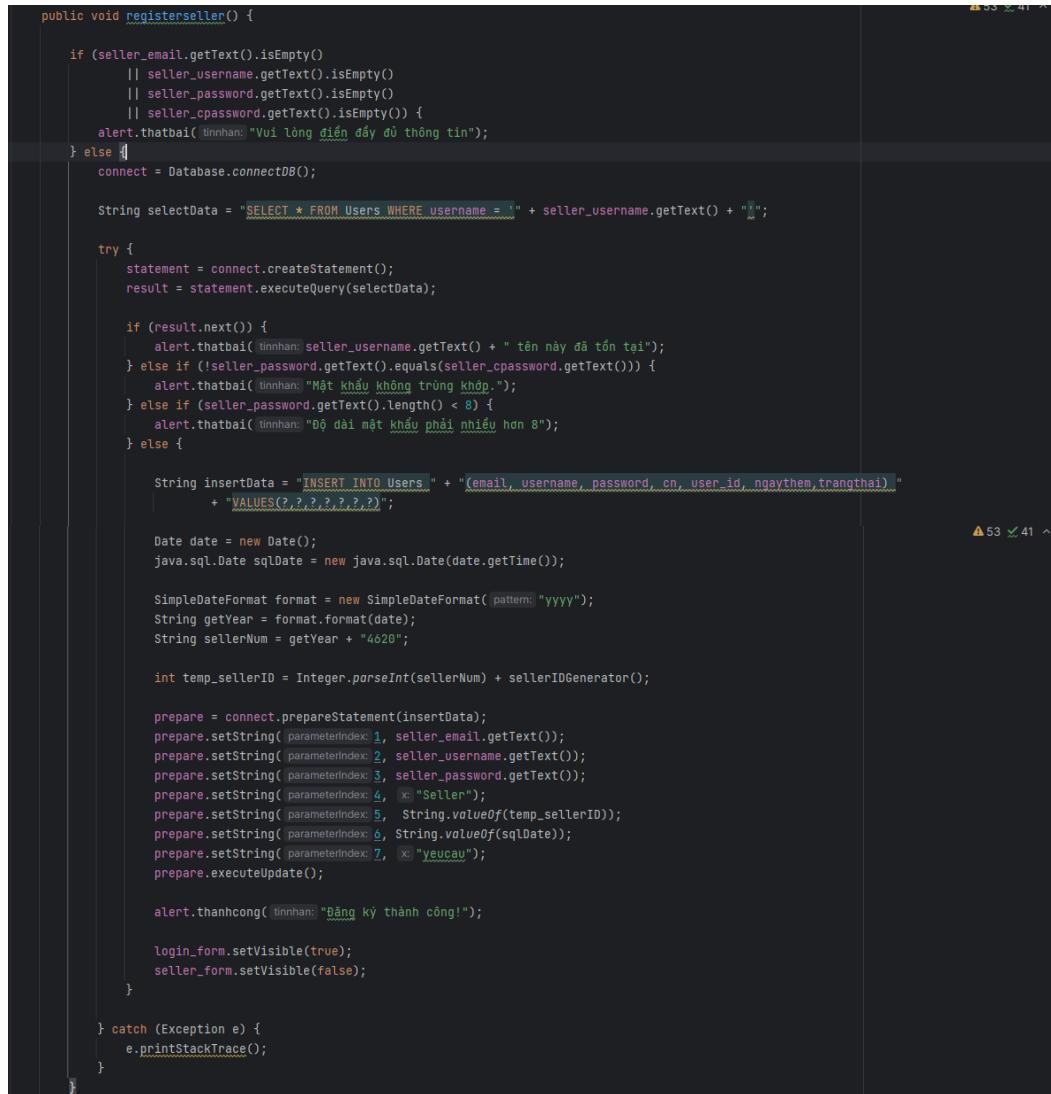
Hình 16. Giao diện khi đã có tài khoản tồn tại

- Giao diện khi đăng ký thành công



Hình 17. Giao diện Đăng ký thành công

1.2.2. Sự kiện đăng ký người dùng seller



```
public void registerseller() {
    if (seller_email.getText().isEmpty()
        || seller_username.getText().isEmpty()
        || seller_password.getText().isEmpty()
        || seller_cpassword.getText().isEmpty()) {
        alert.thatbai(tinhan: "Vui lòng điền đầy đủ thông tin");
    } else {
        connect = Database.connectDB();

        String selectData = "SELECT * FROM Users WHERE username = '" + seller_username.getText() + "'";

        try {
            statement = connect.createStatement();
            result = statement.executeQuery(selectData);

            if (result.next()) {
                alert.thatbai(tinhan: seller_username.getText() + " tên này đã tồn tại");
            } else if (!seller_password.getText().equals(seller_cpassword.getText())) {
                alert.thatbai(tinhan: "Mật khẩu không trùng khớp.");
            } else if (seller_password.getText().length() < 8) {
                alert.thatbai(tinhan: "Độ dài mật khẩu phải nhiều hơn 8");
            } else {

                String insertData = "INSERT INTO Users" + "(email, username, password, cn, user_id, ngaythem, trangthai)" +
                    "VALUES (?, ?, ?, ?, ?, ?)";

                Date date = new Date();
                java.sql.Date sqlDate = new java.sql.Date(date.getTime());

                SimpleDateFormat format = new SimpleDateFormat(pattern: "yyyy");
                String getYear = format.format(date);
                String sellerNum = getYear + "4620";

                int temp_sellerID = Integer.parseInt(sellerNum) + sellerIDGenerator();

                prepare = connect.prepareStatement(insertData);
                prepare.setString(parameterIndex: 1, seller_email.getText());
                prepare.setString(parameterIndex: 2, seller_username.getText());
                prepare.setString(parameterIndex: 3, seller_password.getText());
                prepare.setString(parameterIndex: 4, "Seller");
                prepare.setString(parameterIndex: 5, String.valueOf(temp_sellerID));
                prepare.setString(parameterIndex: 6, String.valueOf(sqlDate));
                prepare.setString(parameterIndex: 7, "yeucau");
                prepare.executeUpdate();

                alert.thanhcong(tinhan: "Đăng ký thành công!");

                login_form.setVisible(true);
                seller_form.setVisible(false);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Hình 18. Sự kiện xử lý đăng ký tài khoản người bán

1.2.3. Sự kiện tạo ngẫu nhiên ID seller



```
1 usage
public int sellerIDGenerator() {

    String sql = "SELECT MAX(id) FROM Users WHERE cn = 'Seller'";

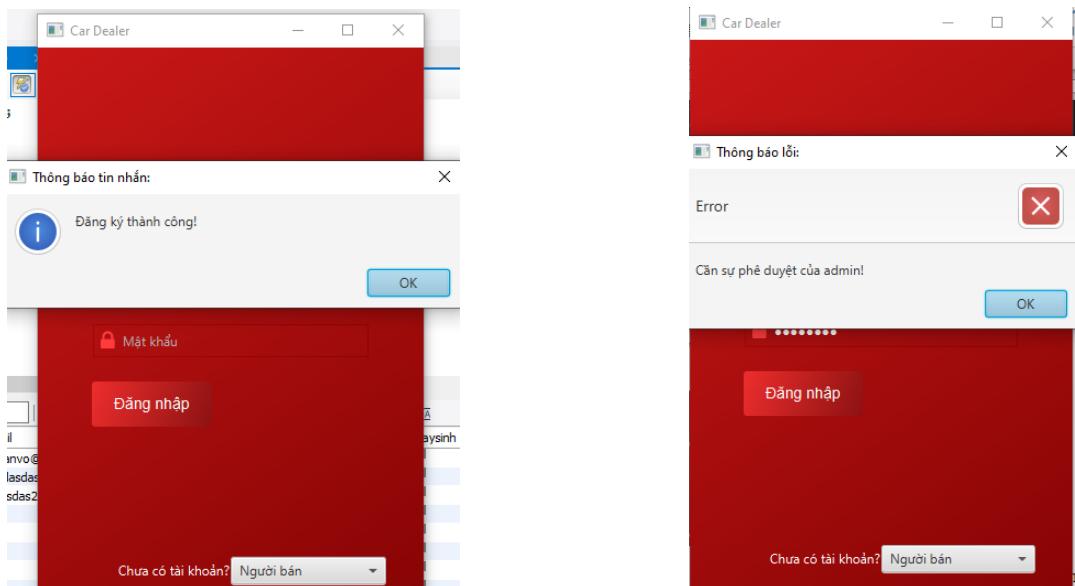
    connect = Database.connectDB();
    int temp_sellerID = 0;
    try {
        prepare = connect.prepareStatement(sql);
        result = prepare.executeQuery();

        if (result.next()) {
            temp_sellerID = result.getInt(columnLabel: "MAX(id)");
        }

        if (temp_sellerID == 0) {
            sellerID = 1;
        } else {
            sellerID = temp_sellerID + 1;
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return sellerID;
}
```

Hình 19. Sự kiện xử lý tạo ngẫu nhiên ID người bán

- Giao diện khi đăng ký seller thành công



Hình 20. Giao diện thông báo đăng ký thành công

Hình 21. Giao diện thông báo cần phê duyệt

2. Cài đặt chức năng “Đăng nhập tài khoản”

2.1. Giao diện đăng nhập



Hình 22. Giao diện Đăng nhập

2.2. Xử lý sự kiện sau khi nhấn nút đăng nhập

```
public void loginAccount() {
    if (Login_username.getText().isEmpty() || Login_password.getText().isEmpty()) {
        alert.thatbai(tinhan: "Vui lòng điền đầy đủ thông tin");
    } else {
        String selectData = "SELECT * FROM Users WHERE username = ? AND password = ?";
        connect = Database.connectDB();
        String role = "";
        String greadname = "";
        ResultSet loginResult = null; // Sử dụng biến khác để lưu kết quả truy vấn

        try {
            prepare = connect.prepareStatement(selectData);
            prepare.setString(parameterIndex: 1, Login_username.getText());
            prepare.setString(parameterIndex: 2, Login_password.getText());
            loginResult = prepare.executeQuery(); // Lưu kết quả vào biến mới

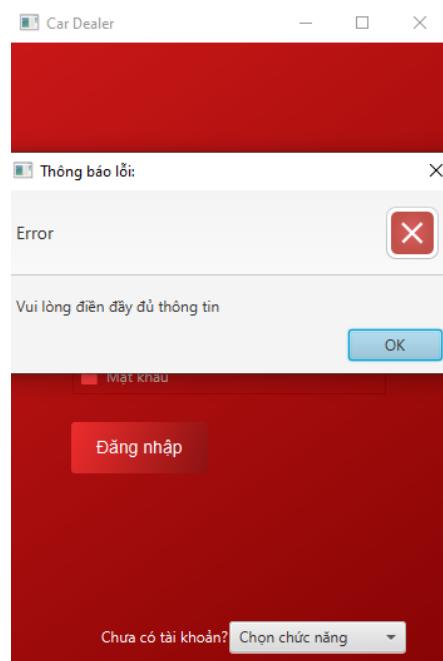
            if (loginResult.next()) {
                if (loginResult.getString(columnLabel: "ngayxoa") == null) {
                    role = loginResult.getString(columnLabel: "cn");
                    greadname = loginResult.getString(columnLabel: "Username");
                    listdata.temp_greadname = greadname;

                    Thread.sleep(millis: 1000);

                    if (role.equals("Admin")) {
                        listdata.admin_username = Login_username.getText();
                        Parent root = FXMLLoader.load(getClass().getResource(name: "IU/Admin.fxml"));
                        Stage stage = new Stage();
                        stage.setTitle("Car dealer System | Admin Portal");
                        stage.setScene(new Scene(root));
                        stage.show();
                        Login_btn.getScene().getWindow().hide();
                    } else if (role.equals("Seller")) {
                        String tempsellerID = loginResult.getString(columnLabel: "user_id");
                        String checkData = "SELECT * FROM Users WHERE user_id = '" + tempsellerID + "'";
                        statement = connect.createStatement();
                        ResultSet sellerResult = statement.executeQuery(checkData); // Kết quả truy vấn mới
                        if (sellerResult.next()) {
                            String trangthai = sellerResult.getString(columnLabel: "trangthai");
                            if (trangthai.equals("yeucau")) {
                                alert.thatbai(tinhan: "Cần sự phê duyệt của admin!");
                            } else if (trangthai.equals("nguoidung")) {
                                alert.thatbai(tinhan: "ERROR!!!!");
                            } else {
                                listdata.seller_username = Login_username.getText();
                                Parent root = FXMLLoader.load(getClass().getResource(name: "IU/seller.fxml"));
                                Stage stage = new Stage();
                                stage.setTitle("Car dealer System | Seller Portal");
                                stage.setScene(new Scene(root));
                                stage.show();
                                Login_btn.getScene().getWindow().hide();
                            }
                        }
                    }
                } else {
                    alert.thatbai(tinhan: "Hiện tài khoản đã bị xóa");
                }
            } else {
                alert.thatbai(tinhan: "Sai Tài khoản/Mật khẩu");
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            // Đóng ResultSet nếu cần thiết
            try {
                if (loginResult != null) {
                    loginResult.close();
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Hình 23. Sự kiện xử lý chức năng đăng nhập

- Giao diện khi nhập thiếu thông tin



Hình 24. Giao diện thông báo nhập không đủ thông tin

3. Cài đặt chức năng “Thông kê”

3.1. Giao diện thống kê



Hình 25. Giao diện thống kê

3.2. Xử lý sự kiện sao khi nhấn nút quản lý

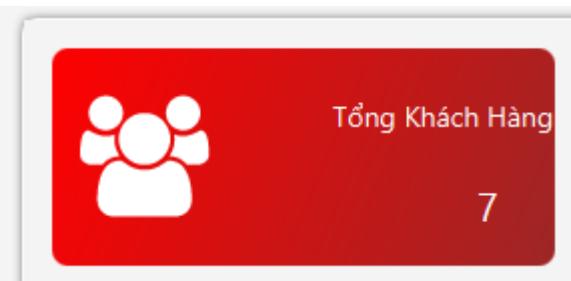
3.2.1. Sự kiện hiển thị tổng khách hàng

```
public void dashboardDisplayTS() {
    String sql = "SELECT COUNT(customer_id) FROM customer";
    connect = Database.connectDB();
    int tempTS = 0;
    try {
        prepare = connect.prepareStatement(sql);
        result = prepare.executeQuery();

        if (result.next()) {
            tempTS = result.getInt(columnLabel: "COUNT(customer_id)");
        }

        if (dashboard_TS != null) {
            dashboard_TS.setText("" + tempTS);
        } else {
            System.out.println("dashboard_TS is null. Please initialize it properly.");
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Hình 26. Sự kiện xử lý hiển thị tổng khách hàng



Hình 27. Giao diện hiển thị tổng khách hàng

3.2.2. Sự kiện hiển thị tổng xe bán

```
public void dashboardDisplayTT() {
    String sql = "SELECT COUNT(customer_id) FROM customer";
    connect = Database.connectDB();
    int tempTT = 0;
    try {
        prepare = connect.prepareStatement(sql);
        result = prepare.executeQuery();

        if (result.next()) {
            tempTT = result.getInt(columnLabel: "COUNT(customer_id)");
        }

        if (dashboard_TT != null) {
            dashboard_TT.setText(String.valueOf(tempTT));
        } else {
            System.out.println("dashboard_TT is null. Please initialize it properly.");
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Hình 28. Sự kiện xử lý hiển thị tổng xe bán



Hình 29. Giao diện hiển thị tổng xe bán

3.2.3. Sự kiện hiển thị tổng khách hàng ngày

```
public void dashboardDisplaySRT() {
    Date date = new Date();
    java.sql.Date sqlDate = new java.sql.Date(date.getTime());
    String sql = "SELECT COUNT(customer_id) FROM customer WHERE date = " + sqlDate + " ";
    connect = Database.connectDB();
    int tempSRT = 0;
    try {
        prepare = connect.prepareStatement(sql);
        result = prepare.executeQuery();

        if (result.next()) {
            tempSRT = result.getInt(columnLabel: "COUNT(customer_id)");
        }
        if (dashboard_SRT != null) {
            dashboard_SRT.setText(String.valueOf(tempSRT));
        } else {
            System.out.println("dashboard_SRT is null. Please initialize it properly.");
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Hình 30. Sự kiện xử lý hiển thị khách hàng theo ngày



Hình 31. Giao diện hiển thị tổng khách hàng theo ngày

3.2.4. Sự kiện hiển thị tổng tiền theo ngày

```
public void dashboardDisplayTI() {
    Date date = new Date();
    java.sql.Date sqlDate = new java.sql.Date(date.getTime());
    String sql = "SELECT SUM(total) FROM customer WHERE date = '" + sqlDate + "'";
    connect = Database.connectDB();
    double tempTI = 0;
    try {
        prepare = connect.prepareStatement(sql);
        result = prepare.executeQuery();

        if (result.next()) {
            tempTI = result.getDouble(columnLabel: "SUM(total)");
        }

        if (dashboard_TI != null) {
            dashboard_TI.setText("$" + tempTI);
        } else {
            System.out.println("dashboard_TI is null. Please initialize it properly.");
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Hình 32. Sự kiện xử lý hiển thị tổng tiền theo ngày



Hình 33. Giao diện hiển thị tổng tiền theo ngày

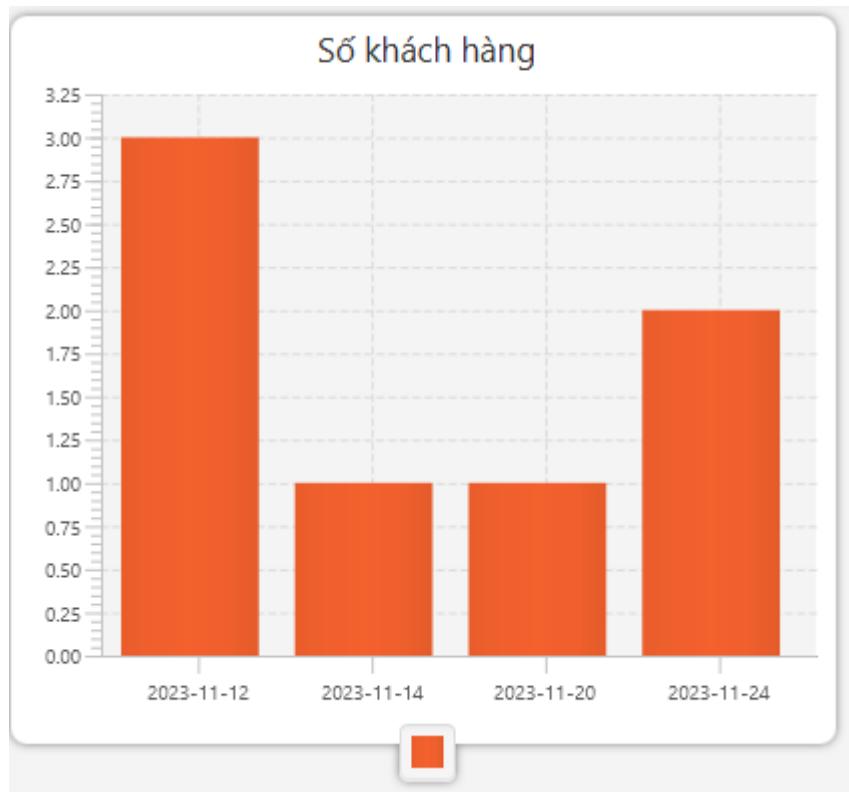
3.2.5. Sự kiện hiển thị biểu đồ cột theo khách hàng theo ngày

```
public void dashboardDTChart() {
    try {
        if (dashboard_chart_DT != null) {
            dashboard_chart_DT.getData().clear();
            String sql = "SELECT DATE(date) AS date, COUNT(customer_id) AS id_count FROM customer" +
                " GROUP BY DATE(date) ORDER BY DATE(date) ASC LIMIT 5";
            connect = Database.connectDB();
            XYChart.Series chart = new XYChart.Series<>();

            prepare = connect.prepareStatement(sql);
            result = prepare.executeQuery();

            while (result.next()) {
                chart.getData().add(new XYChart.Data<>(result.getString(columnIndex: 1), result.getInt(columnIndex: 2)));
            }
            dashboard_chart_DT.getData().add(chart);
        } else {
            System.out.println("dashboard_chart_DT is null. Please initialize it properly.");
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Hình 34. Sự kiện xử lý hiển thị biểu đồ cột số khách theo ngày



Hình 35. Giao diện hiển thị biểu đồ cột số khách hàng theo ngày

3.2.6. Sự kiện hiển thị biểu đồ đường theo tổng tiền theo ngày

```
public void dashboardDIChart() {
    try {
        if ( dashboard_chart_DI != null) {
            dashboard_chart_DI.getData().clear();

            String sql = "SELECT DATE(date) AS date, SUM(total) AS total_price FROM customer " +
                         "GROUP BY DATE(date) ORDER BY DATE(date) ASC LIMIT 5";

            connect = Database.connectDB();

            XYChart.Series chart = new XYChart.Series<>();

            prepare = connect.prepareStatement(sql);
            result = prepare.executeQuery();

            while (result.next()) {
                chart.getData().add(new XYChart.Data<>(result.getString( columnIndex: 1), result.getInt( columnIndex: 2)));
            }
            dashboard_chart_DI.getData().add(chart);
        } else {
            System.out.println("dashboard_chart_DI is null. Please initialize it properly.");
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Hình 36. Sự kiện xử lý hiển thị biểu đồ đường Doanh thu theo ngày



Hình 37. Giao diện hiển thị biểu đồ đường doanh thu theo ngày

4. Cài đặt chức năng “Thêm người dùng”

4.1. Giao diện thêm người dùng

ID	Username	Password	Role	Email	Họ Tên	Giới Tính	Ngày Sinh	Ngày Xóa	Ngày Cập Nhật	Ngày T... Trang Thái
20230...	maitanvo	maitanvo	Seller	maitanvo...						2023-10... nguoban
20230...	maitanvo1	maitanvo	Seller	asmadasda...						2023-10... nguoban
20234...	maitanvo2	maitanvo	Seller	asdasdas...				2023-11-25		2023-10... nguoban
0	admin	maitanvo	Admin					2023-10-06		
0	ckynet	maitanvo	Seller					2023-10-06		nguoban
20230...	ckynet	maitanvo	Seller					2023-10-06		nguoban
20230...	ckynet	maitanvo	Seller					2023-10-06		nguoban
123123	ckyentvp	maitanvo	Admin							
12312...	asdadsa	asdasd	Admin						2023-11...	
0	VoAdmin	maitanvo	Admin						2023-11...	
0	VoAdmin1	maitanvo	Admin						2023-11...	
20234...	test	maitanvo	Seller	sieunhanx...				2023-11-25	2023-11...	Nguoiban
20234...	testseller	maitanvo	Seller	sieunhanx...					2023-11...	yeucau

Hình 38. Giao diện Thêm người dùng

Thêm người dùng

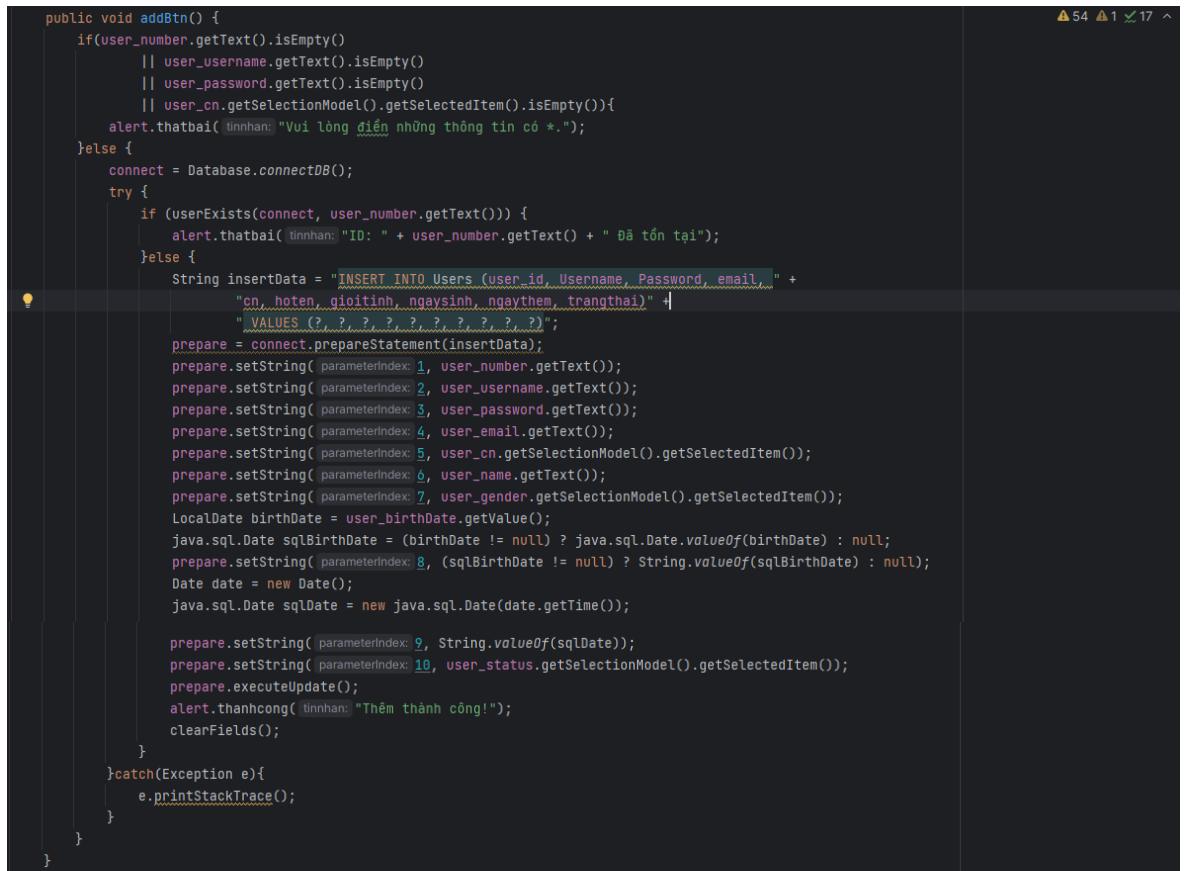
Thông tin người dùng

*ID: <input type="text"/>	*Username: <input type="text"/>
Họ và Tên: <input type="text"/>	*Password: <input type="password"/>
Ngày sinh: <input type="text"/> <input type="button" value="..."/>	Email: <input type="text"/>
Giới tính: <input type="button" value="Choose..."/>	*Chức năng: <input type="button" value="Choose..."/>
Trạng thái: <input type="button" value="Choose..."/>	<input type="button" value="Add"/> <input type="button" value="Update"/>

Hình 39. Giao diện thêm

4.2. Xử lý sự kiện

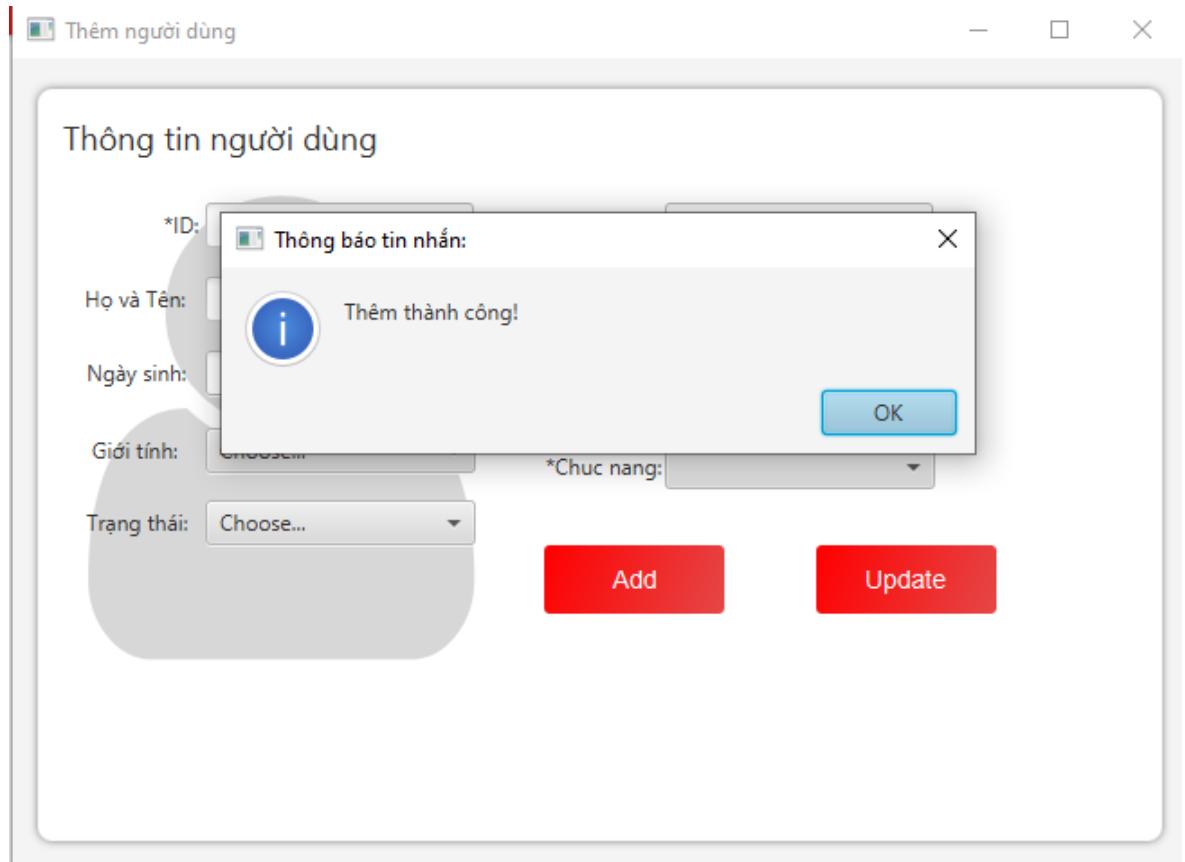
4.2.1. Sự kiện thêm người dùng



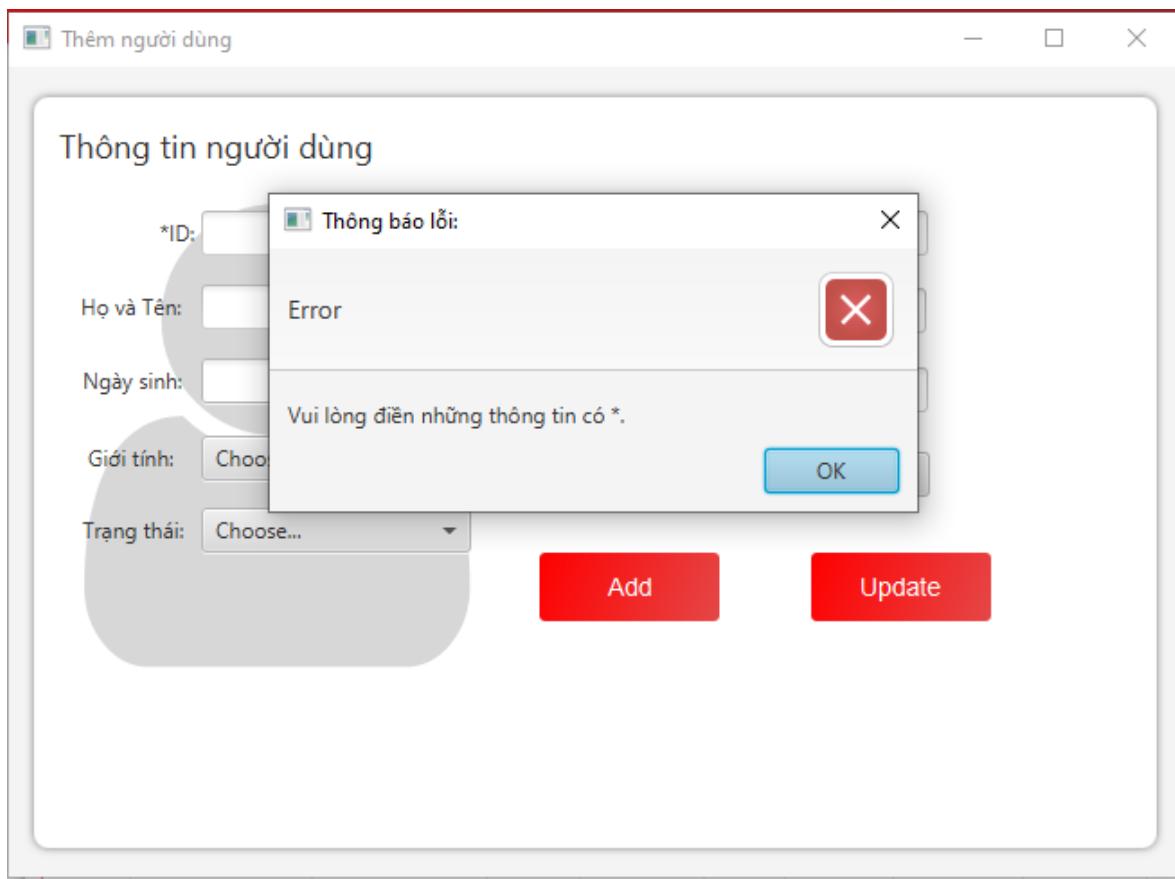
```
public void addBtn() {
    if(user_number.getText().isEmpty()
        || user_username.getText().isEmpty()
        || user_password.getText().isEmpty()
        || user_cn.getSelectionModel().getSelectedItem().isEmpty()){
        alert.thatbai( tinhnhan: "Vui lòng điền những thông tin có *.");
    }else {
        connect = Database.connectDB();
        try {
            if (userExists(connect, user_number.getText())) {
                alert.thatbai( tinhnhan: "ID: " + user_number.getText() + " Đã tồn tại");
            }else {
                String insertData = "INSERT INTO Users (user_id, Username, Password, email, cn, hoten, gioitinh, ngaysinh, ngaythem, trangthai)" +
                    "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
                prepare = connect.prepareStatement(insertData);
                prepare.setString( parameterIndex: 1, user_number.getText());
                prepare.setString( parameterIndex: 2, user_username.getText());
                prepare.setString( parameterIndex: 3, user_password.getText());
                prepare.setString( parameterIndex: 4, user_email.getText());
                prepare.setString( parameterIndex: 5, user_cn.getSelectionModel().getSelectedItem());
                prepare.setString( parameterIndex: 6, user_name.getText());
                prepare.setString( parameterIndex: 7, user_gender.getSelectionModel().getSelectedItem());
                LocalDate birthDate = user_birthDate.getValue();
                java.sql.Date sqlBirthDate = (birthDate != null) ? java.sql.Date.valueOf(birthDate) : null;
                prepare.setString( parameterIndex: 8, (sqlBirthDate != null) ? String.valueOf(sqlBirthDate) : null);
                Date date = new Date();
                java.sql.Date sqlDate = new java.sql.Date(date.getTime());

                prepare.setString( parameterIndex: 9, String.valueOf(sqlDate));
                prepare.setString( parameterIndex: 10, user_status.getSelectionModel().getSelectedItem());
                prepare.executeUpdate();
                alert.thanhcong( tinhnhan: "Thêm thành công!");
                clearFields();
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

Hình 39. Sự kiện xử lý thêm người dùng



Hình 40. Giao diện thông báo thêm thành công

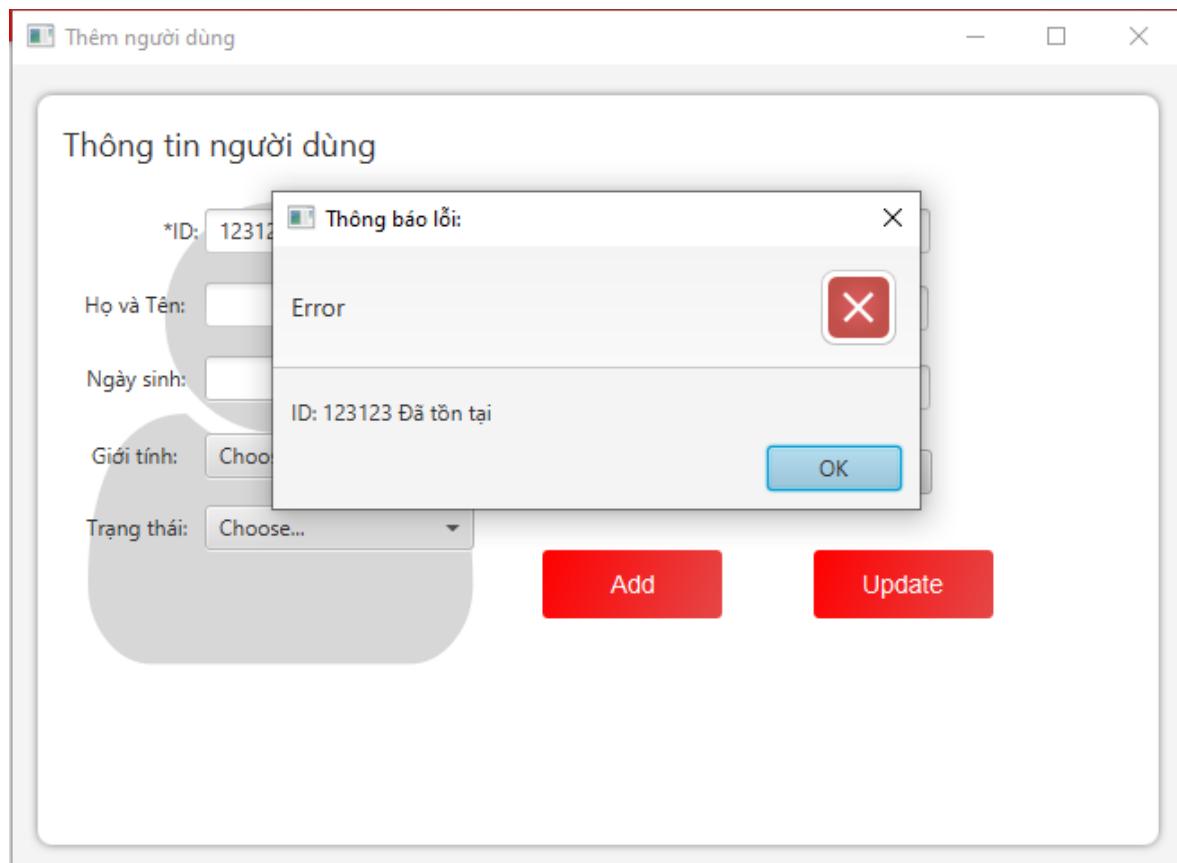


Hình 41. Giao diện thông báo nhập không đủ thông tin

4.2.2. Sự kiện kiểm tra người dùng đã tồn tại chưa

```
private boolean userExists(Connection connection, String userId) throws SQLException {
    // Kiểm tra xem người dùng có tồn tại hay không
    String checkUserNum = "SELECT * FROM Users WHERE user_id = ?";
    try (PreparedStatement preparedStatement = connection.prepareStatement(checkUserNum)) {
        preparedStatement.setString(1, userId);
        return preparedStatement.executeQuery().next();
    }
}
```

Hình 42. Sự kiện xử lý kiểm tra người dùng tồn tại chưa

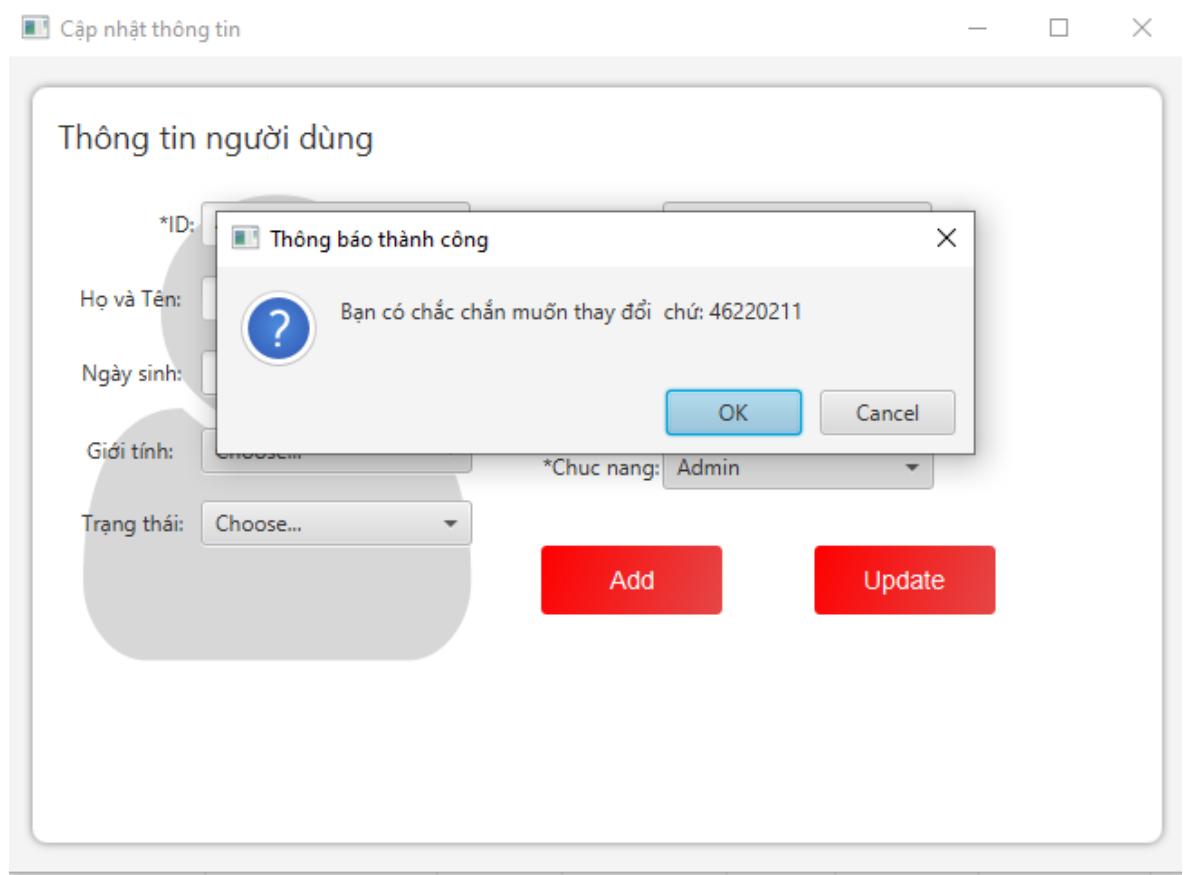


Hình 43. Giao diện thông báo người dùng đã tồn tại

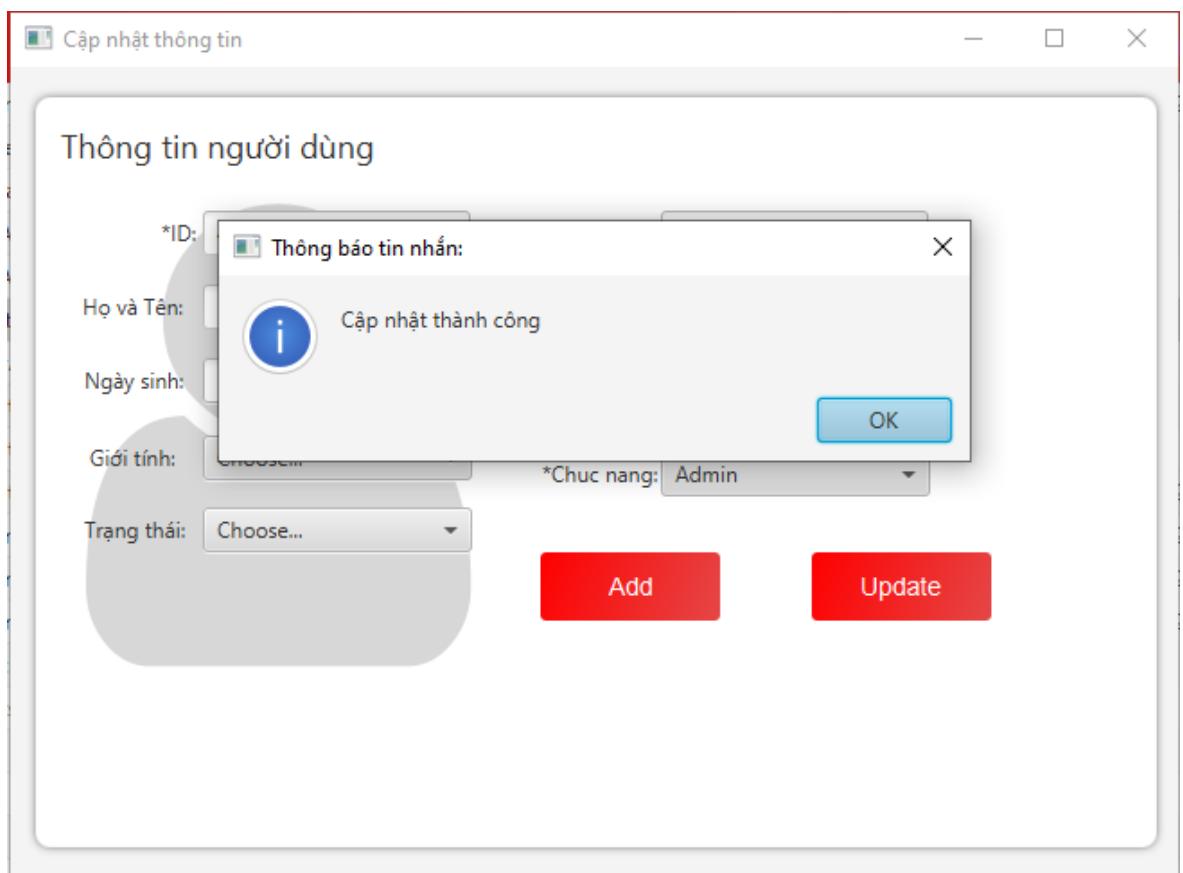
4.2.3. Sự kiện cập nhật thông tin tài khoản

```
public void updateBtn() {
    if(user_number.getText().isEmpty()
        || user_username.getText().isEmpty()
        || user_password.getText().isEmpty()
        || user_cn.getSelectionModel().getSelectedItem().isEmpty()){
        alert.thatbai(tinnhan: "Vui lòng điền những thông tin có *.");
    }else {
        Date date = new Date();
        java.sql.Date sqlDate = new java.sql.Date(date.getTime());
        if (alert.xacnhan(tinnhan: "Bạn có chắc chắn muốn thay đổi chữ: "
            + user_number.getText())){
            String updateData = "UPDATE Usens SET "
                + "hoten = ?, ngaysinh = ?, gioitinh = ?, trangthai = ?, Password = ?, email = ?, ngaycapnhat = ?"
                + " WHERE user_id = ?";
            try{
                prepare = connect.prepareStatement(updateData);
                prepare.setString( parameterIndex: 1, user_name.getText());
                LocalDate birthDate = user_birthDate.getValue();
                if (birthDate != null) {
                    prepare.setDate( parameterIndex: 2, java.sql.Date.valueOf(birthDate));
                } else {
                    prepare.setNull( parameterIndex: 2, java.sql.Types.DATE);
                }
                prepare.setString( parameterIndex: 3, user_gender.getSelectionModel().getSelectedItem());
                prepare.setString( parameterIndex: 4, user_status.getSelectionModel().getSelectedItem());
                prepare.setString( parameterIndex: 5, user_password.getText());
                prepare.setString( parameterIndex: 6, user_email.getText());
                prepare.setString( parameterIndex: 7, String.valueOf(sqlDate));
                prepare.setString( parameterIndex: 8, user_number.getText());
                prepare.executeUpdate();
                alert.thanhcong(tinnhan: "Cập nhật thành công");
            }catch (Exception e){
                e.printStackTrace();
            }
        }else {
            alert.thatbai(tinnhan: "Hủy");
        }
    }
}
```

Hình 44. Sự kiện xử lý cập nhật thông tin người dùng



Hình 45. Giao diện thông báo xác nhận thay đổi



Hình 46. Giao diện thông báo cập nhật thành công

4.2.4. Sự kiện xóa người dùng

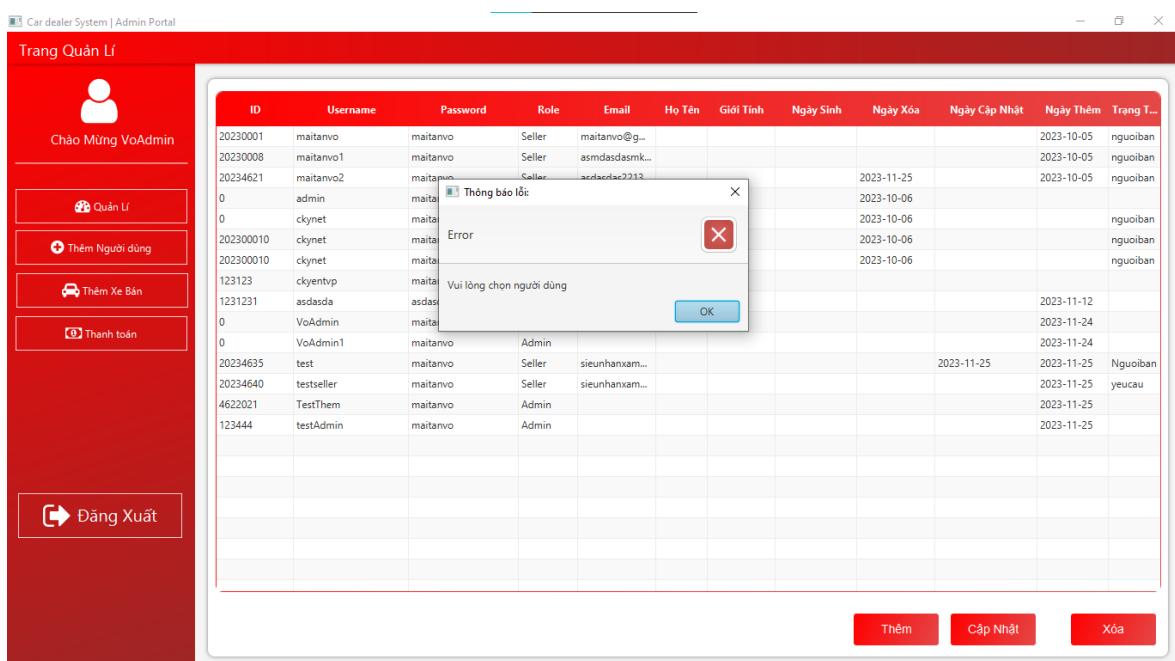
```

public void addUserDeleteBtn(){
    UserData uData = AddUser_tableView.getSelectionModel().getSelectedItem();
    int num = AddUser_tableView.getSelectionModel().getSelectedIndex();
    Date date = new Date();
    java.sql.Date sqlDate = new java.sql.Date(date.getTime());
    if ((num - 1) < -1) {
        alert.thatbai( tinhhan: "Vui lòng chọn người dùng");
        return;
    } else {
        if (alert.xacnhhan( tinhhan: "Bạn có chắc chắn xóa người dùng này: "
            + uData.getUser_id() + "?")) {
            String deleteData = "UPDATE Users SET ngayxoa = ? WHERE user_id = ?";
            connect = Database.connectDB();

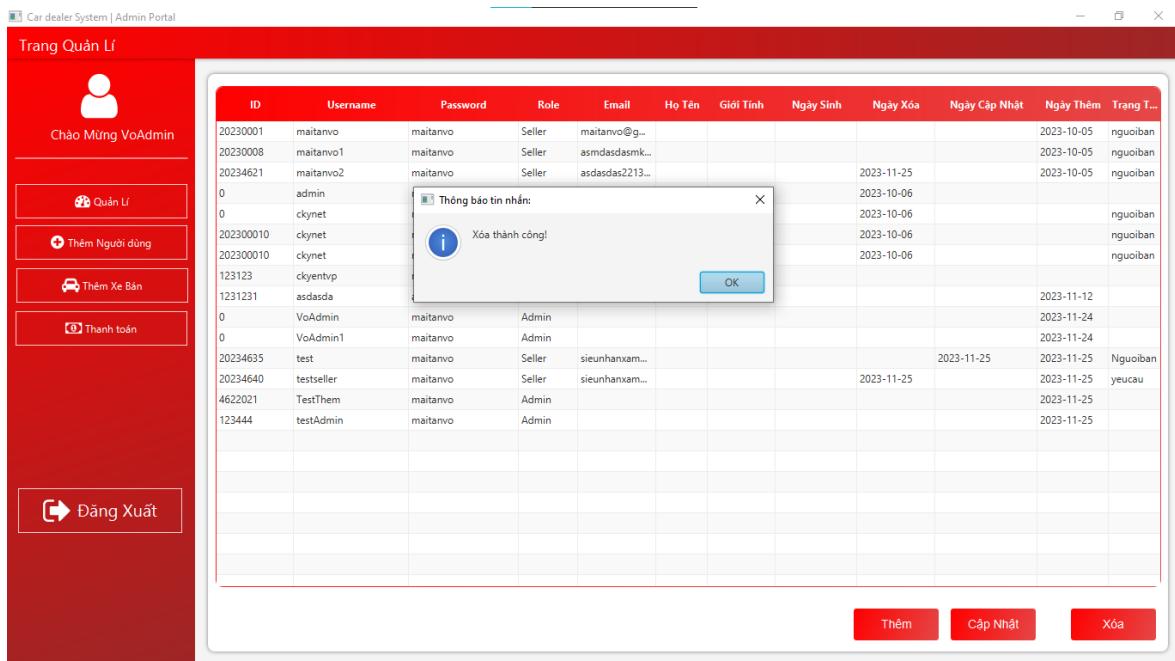
            try {
                prepare = connect.prepareStatement(deleteData);
                prepare.setString( parameterIndex: 1, String.valueOf(sqlDate));
                prepare.setString( parameterIndex: 2, uData.getUser_id());
                prepare.executeUpdate();
                alert.thanhcong( tinhhan: "Xóa thành công!");
            } catch (Exception e) {
                e.printStackTrace();
            }
        } else {
            alert.thatbai( tinhhan: "Bị hủy.");
        }
    }
    addUserDisplayData();
}

```

Hình 47. Sự kiện xử lý xóa người dùng



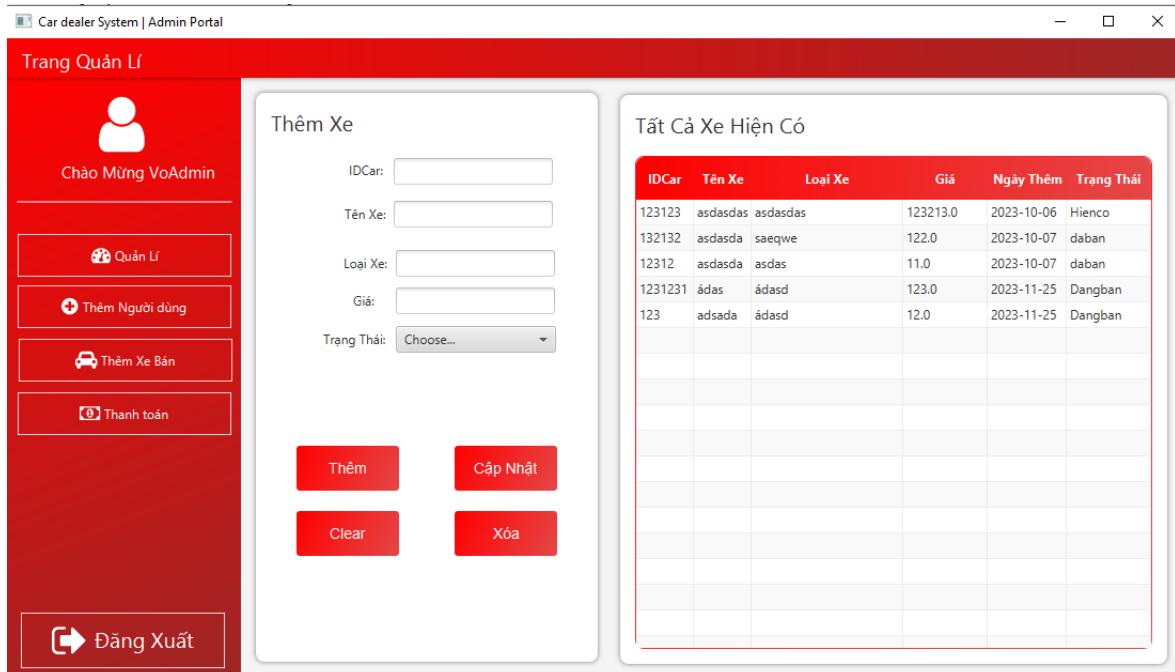
Hình 48. Giao diện thông báo yêu cầu chọn người dùng để xóa



Hình 49. Giao diện thông báo xóa thành công

5. Cài đặt chức năng “Thêm xe bán”

5.1. Giao diện thêm xe bán



Hình 50. Giao diện thêm xe bán

5.2. Xử lý sự kiện

5.2.1. Sự kiện thêm xe bán

```
public void addCarAddBtn() {
    connect = Database.connectDB();
    String checkExistingQuery = "SELECT COUNT(*) FROM car WHERE car_id = ? AND date_remove IS NULL";
    String sql = "INSERT INTO car (car_id, brand, model, price, status, date)" + "VALUES(?, ?, ?, ?, ?, ?)";
    try {
        prepare = connect.prepareStatement(checkExistingQuery);
        prepare.setString( parameterIndex: 1, addCar_carID.getText());
        ResultSet resultSet = prepare.executeQuery();
        resultSet.next();
        int count = resultSet.getInt( columnIndex: 1);
        if (count > 0) {
            alert.thatbai( tinhnhan: "Car ID đã tồn tại");
        } else {
            if(addCar_car.getText().isEmpty()
                || addCar_model.getText().isEmpty()
                || addCar_price.getText().isEmpty()
                || addCar_status.getSelectionModel().getSelectedItem().isEmpty()){
                alert.thatbai( tinhnhan: "Vui lòng điền đủ thông tin.");
            }else {

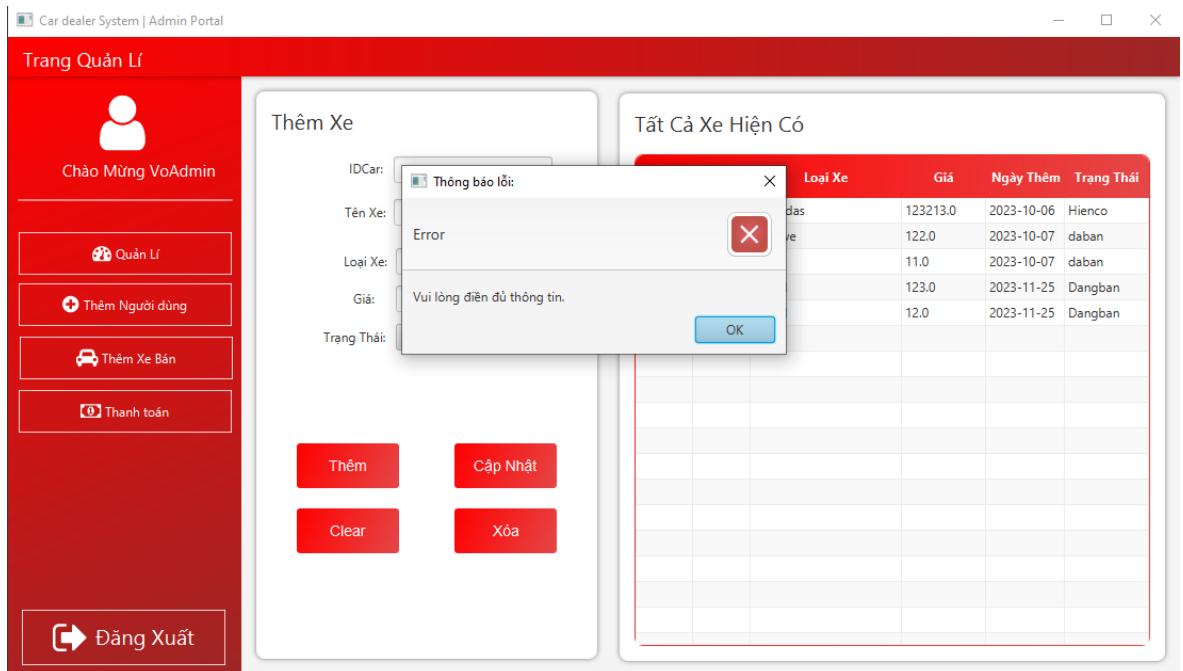
                prepare = connect.prepareStatement(sql);
                prepare.setString( parameterIndex: 1, addCar_carID.getText());
                prepare.setString( parameterIndex: 2, addCar_car.getText());
                prepare.setString( parameterIndex: 3, addCar_model.getText());
                prepare.setString( parameterIndex: 4, addCar_price.getText());
                prepare.setString( parameterIndex: 5, addCar_status.getSelectionModel().getSelectedItem());
                Date date = new Date();
                java.sql.Date sqlDate = new java.sql.Date(date.getTime());

                prepare.setString( parameterIndex: 6, String.valueOf(sqlDate));
                prepare.executeUpdate();
                alert.thanhcong( tinhnhan: "Thêm thành công!");

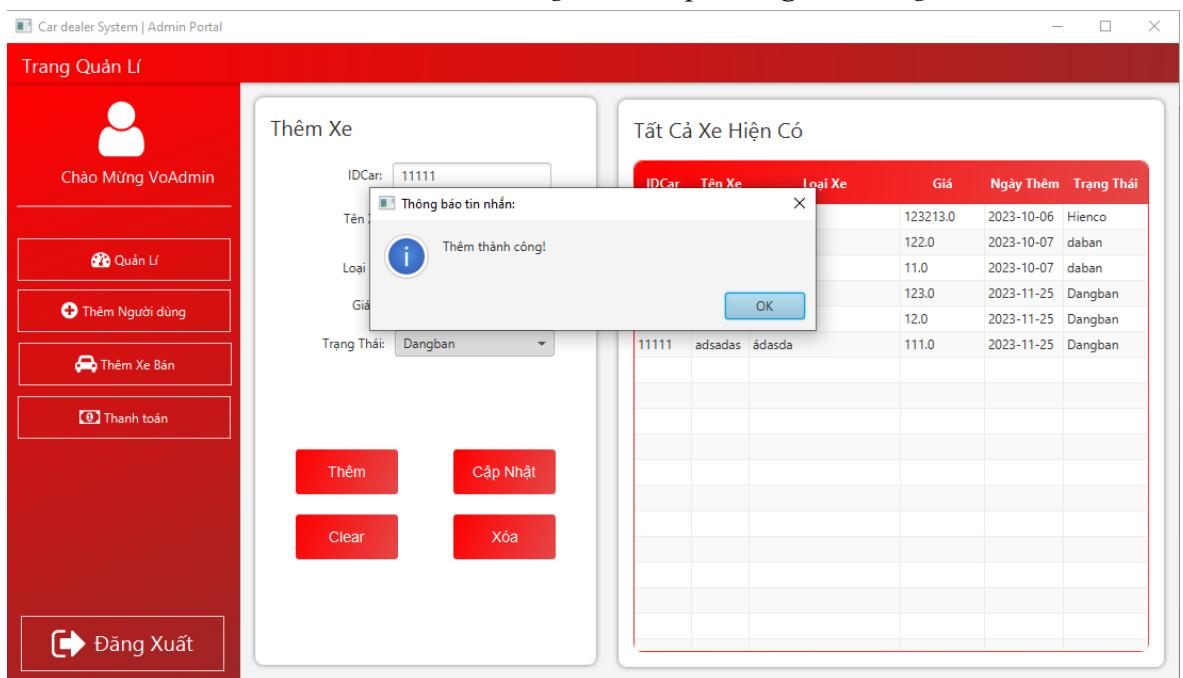
                addCarDisplayData();

            }
        }
    }catch(Exception e){
        e.printStackTrace();
    }
}
```

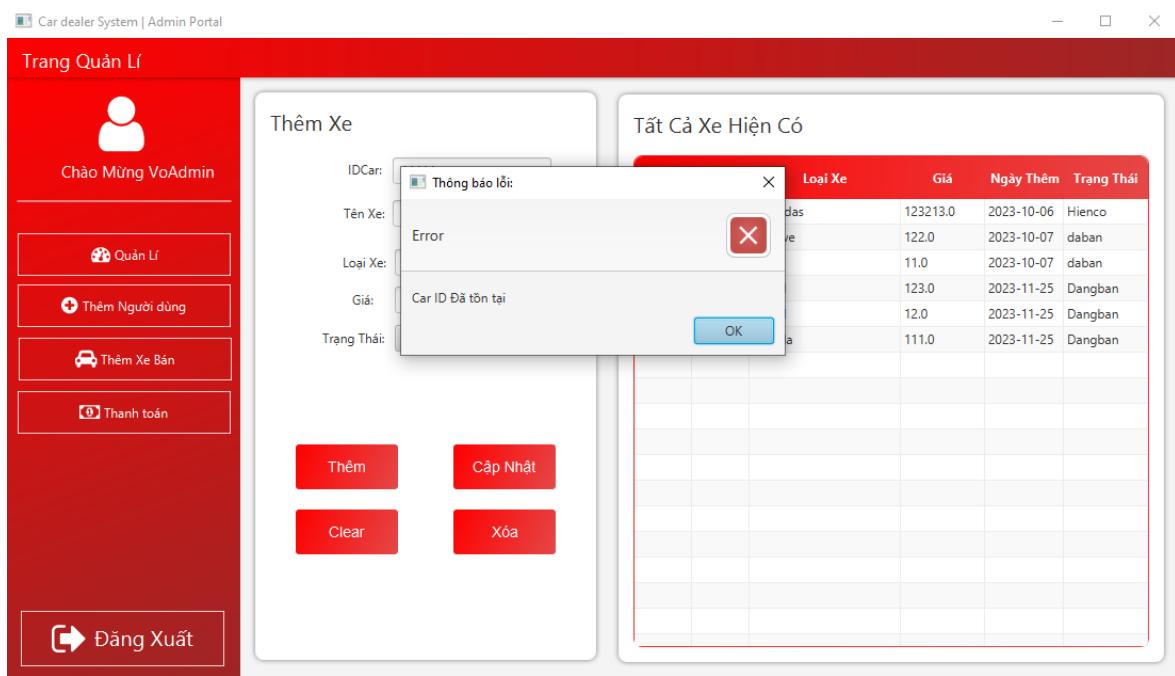
Hình 51. Sự kiện xử lý thêm xe bán



Hình 52. Giao diện thông báo nhập không đủ thông tin



Hình 53. Giao diện thông báo thêm xe thành công



Hình 54. Giao diện thông báo trùng id_car

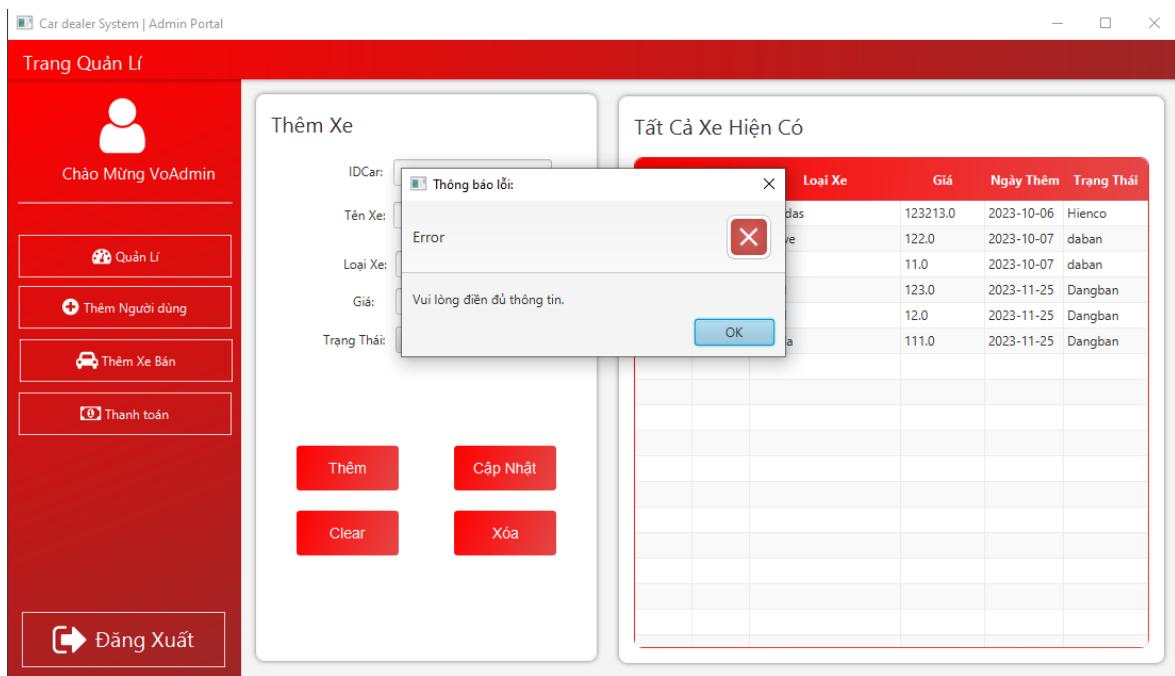
5.2.2. Sự kiện cập nhật thông tin

```

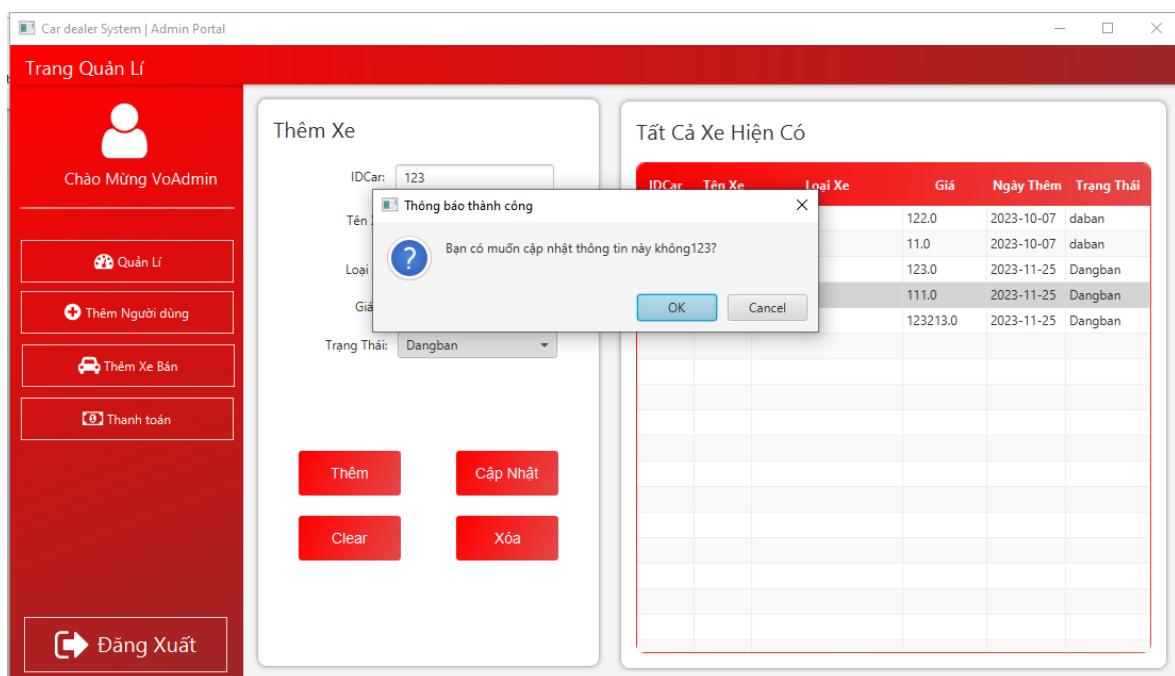
public void addCarUpdateBtn() {
    connect = Database.connectDB();
    if (addCar_car.getText().isEmpty()
        || addCar_model.getText().isEmpty()
        || addCar_price.getText().isEmpty()
        || addCar_status.getSelectionModel().getSelectedItem() == null
        || addCar_status.getSelectionModel().getSelectedItem().isEmpty()) {
        alert.thatbai( tinnhan: "Vui lòng điền đủ thông tin.");
    } else {
        if (alert.xacnhan( tinnhan: "Bạn có muốn cập nhật thông tin này không" + addCar_carID.getText() + "?")) {
            Date date = new Date();
            java.sql.Date sqlDate = new java.sql.Date(date.getTime());
            String updateData = "UPDATE car SET car_id = ?, brand = ?, model = ?, date_update = ?, status = ?, price = ? WHERE car_id = ?";
            try {
                prepare = connect.prepareStatement(updateData);
                prepare.setString( parameterIndex: 1, addCar_carID.getText());
                prepare.setString( parameterIndex: 2, addCar_car.getText());
                prepare.setString( parameterIndex: 3, addCar_model.getText());
                prepare.setDate( parameterIndex: 4, sqlDate);
                prepare.setString( parameterIndex: 5, addCar_status.getSelectionModel().getSelectedItem());
                prepare.setString( parameterIndex: 6, addCar_price.getText());
                prepare.setString( parameterIndex: 7, addCar_carID.getText());
                int updatedRows = prepare.executeUpdate();
                if (updatedRows > 0) {
                    alert.thanhcong( tinnhan: "Cập nhật thành công!");
                } else {
                    alert.thatbai( tinnhan: "Cập nhật không thành công!");
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        } else {
            alert.thatbai( tinnhan: "Hủy");
        }
    }
}

```

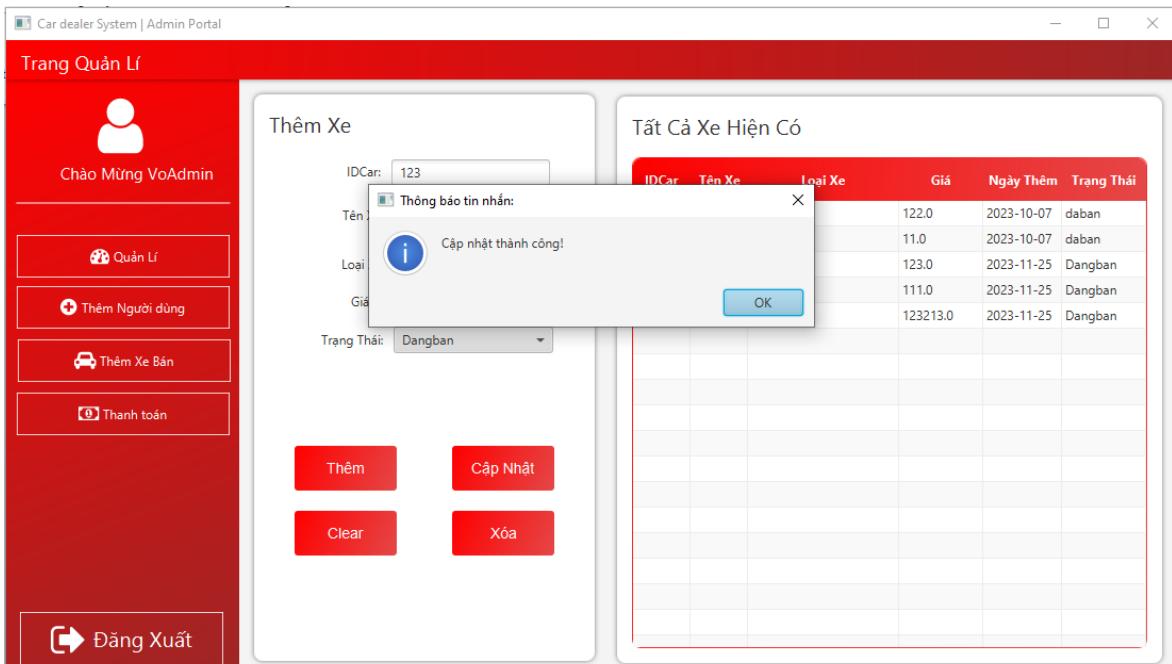
Hình 55. Sự kiện xử lý cập nhật thông tin xe



Hình 56. Giao diện thông báo nhập không đủ thông tin để cập nhật



Hình 57. Giao diện thông báo xác nhận cập nhật



Hình 58. Giao diện thông báo cập nhật thành công

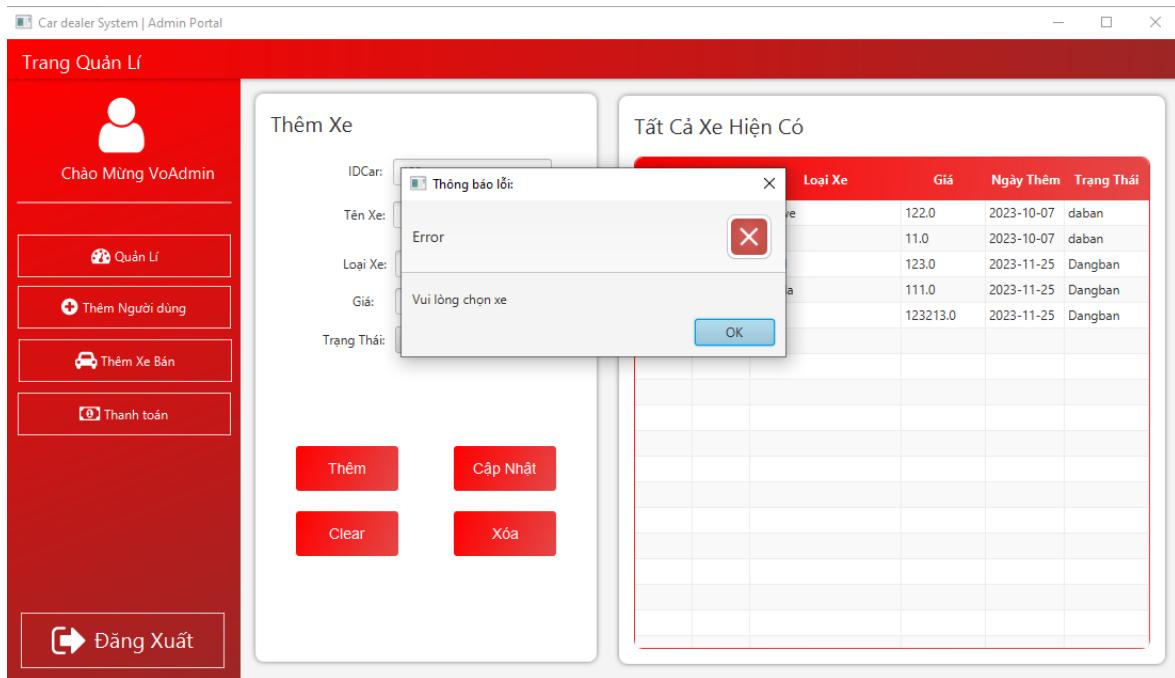
5.2.3. Sự kiện xóa xe bán

```

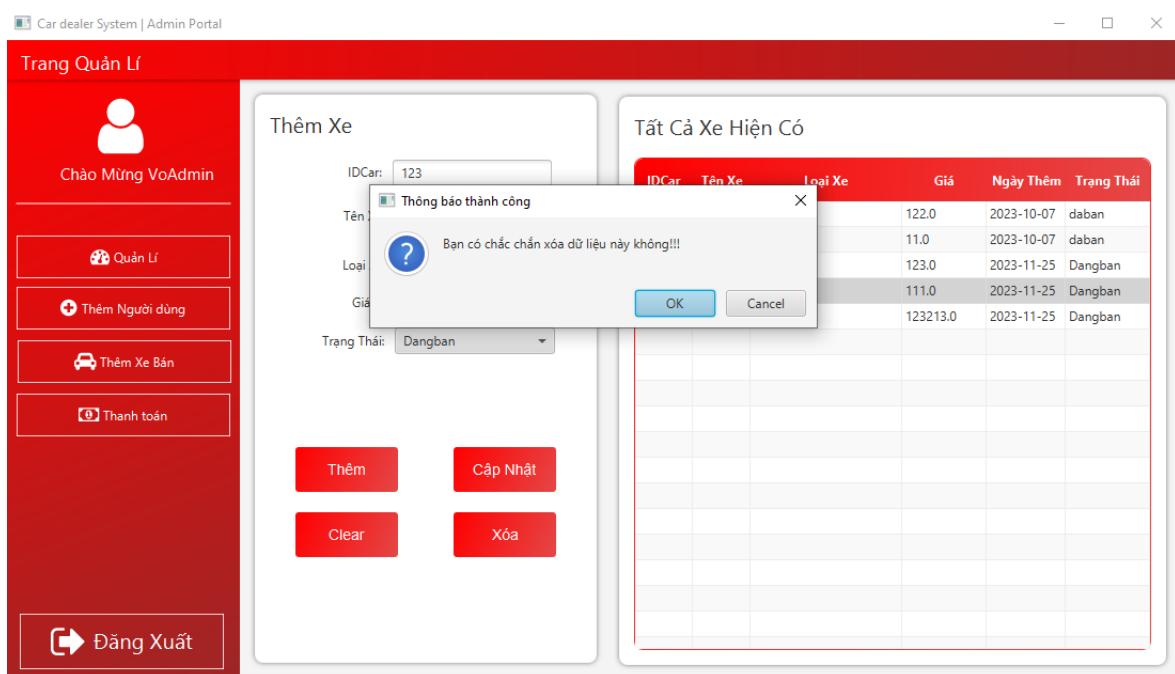
public void addCarDeleteBtn() {
    Timer timer = new Timer();
    long delay = 24 * 60 * 60 * 1000;
    long period = 24 * 60 * 60 * 1000;
    timer.scheduleAtFixedRate(() -> {
        String deleteQuery = "DELETE FROM car WHERE date_remove IS NOT NULL";
        connect = Database.connectDB();
        try {
            prepare = connect.prepareStatement(deleteQuery);
            prepare.executeUpdate();
        } catch (Exception e) {
            e.printStackTrace();
        }
        System.out.println("Xóa dữ liệu cũ...");
    }, delay, period);
    int num = addCar_tableView.getSelectionModel().getSelectedIndex();
    if ((num - 1) < -1 ) {
        alert.thatbai(tinhan: "Vui lòng chọn xe");
        return;
    }else{
        if (alert.xacnhuan(tinhan: "Bạn có chắc chắn xóa dữ liệu này không!!!")){
            Date date = new Date();
            java.sql.Date sqlDate = new java.sql.Date(date.getTime());
            String deleteData = "UPDATE car SET date_remove = ? WHERE car_id = ?";
            connect = Database.connectDB();
            try {
                carData selectedCar = addCar_tableView.getSelectionModel().getSelectedItem();
                prepare = connect.prepareStatement(deleteData);
                prepare.setString( parameterIndex: 1, String.valueOf(sqlDate));
                prepare.setInt( parameterIndex: 2, selectedCar.getCarId());
                prepare.executeUpdate();
                addCarDisplayData();
                alert.thanhcong(tinhan: "Xóa thành công!");
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}

```

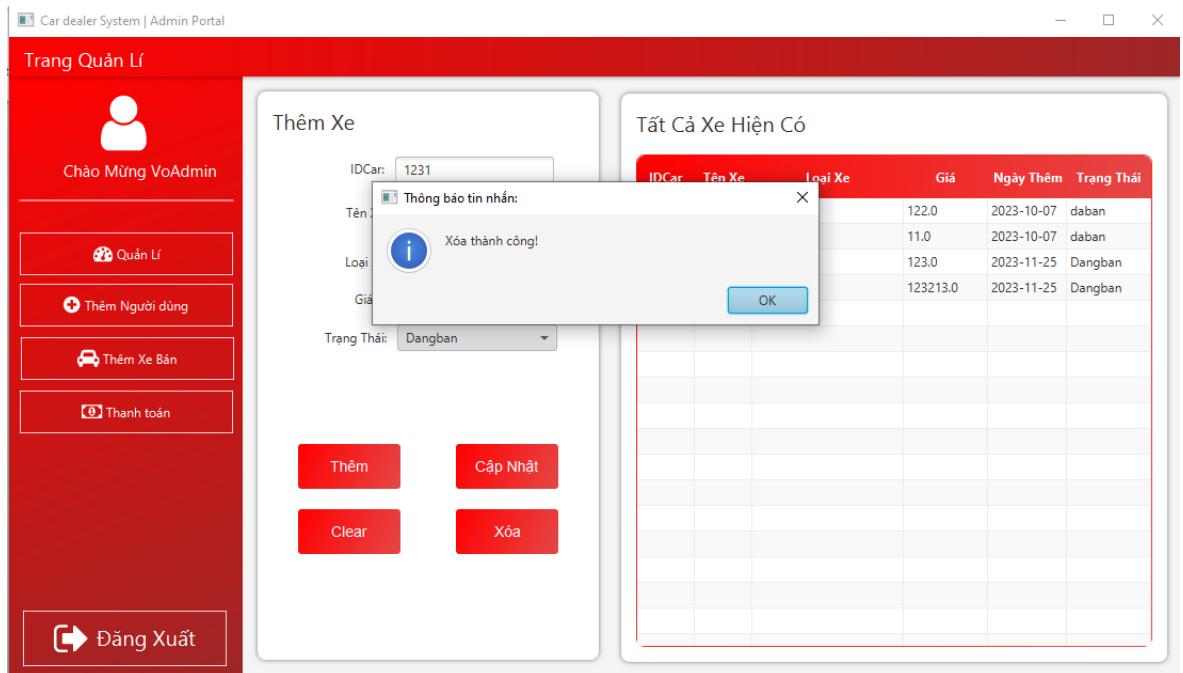
Hình 59. Sự kiện xử lý xóa xe



Hình 60. Giao diện thông báo yêu cầu chọn xe trước khi xóa



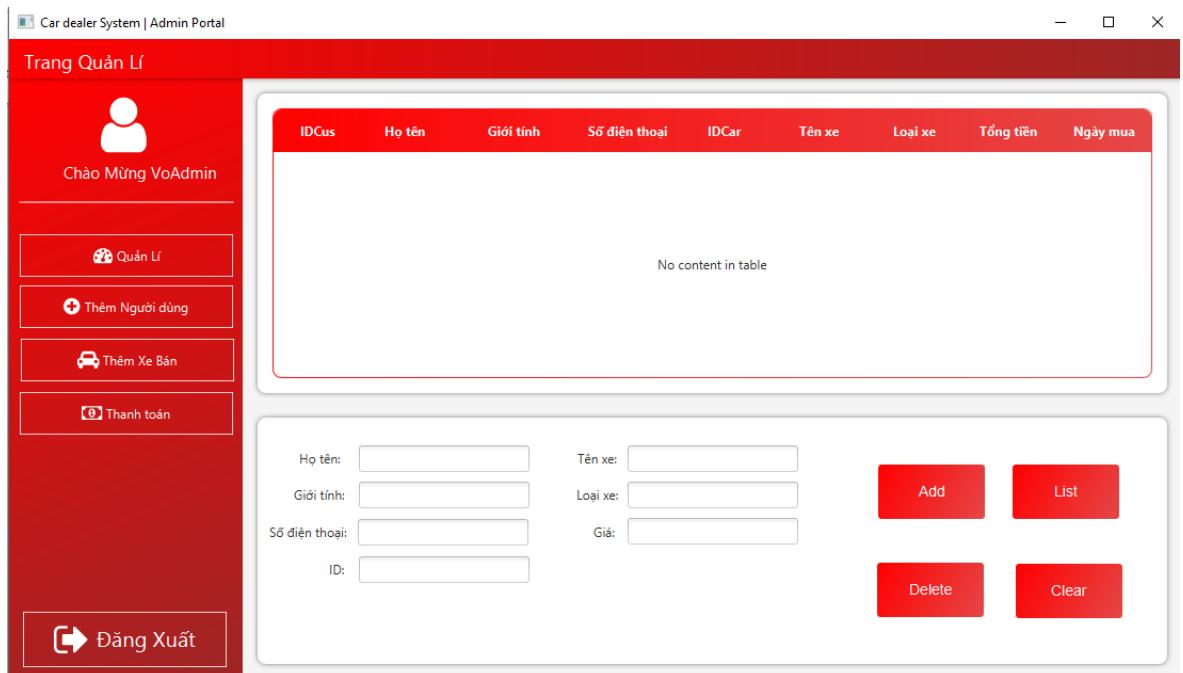
Hình 61. Giao diện thông báo xác nhận xóa dữ liệu xe



Hình 62. Giao diện thông báo xóa thành công

6. Cài đặt chức năng “Thanh toán”

6.1. Giao diện thanh toán



Hình 63. Giao diện thanh toán

6.2. Xử lý sự kiện

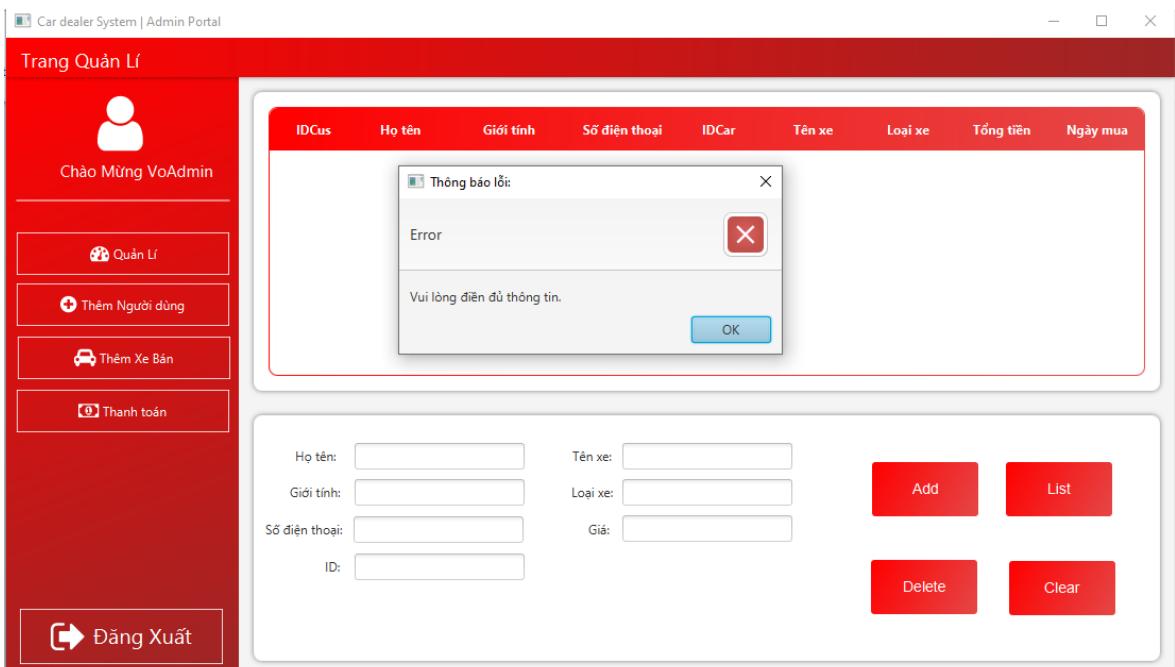
6.2.1. Sự kiện thêm thanh toán

```
public void thanhtoanAddBtn() {
    connect = Database.connectDB();

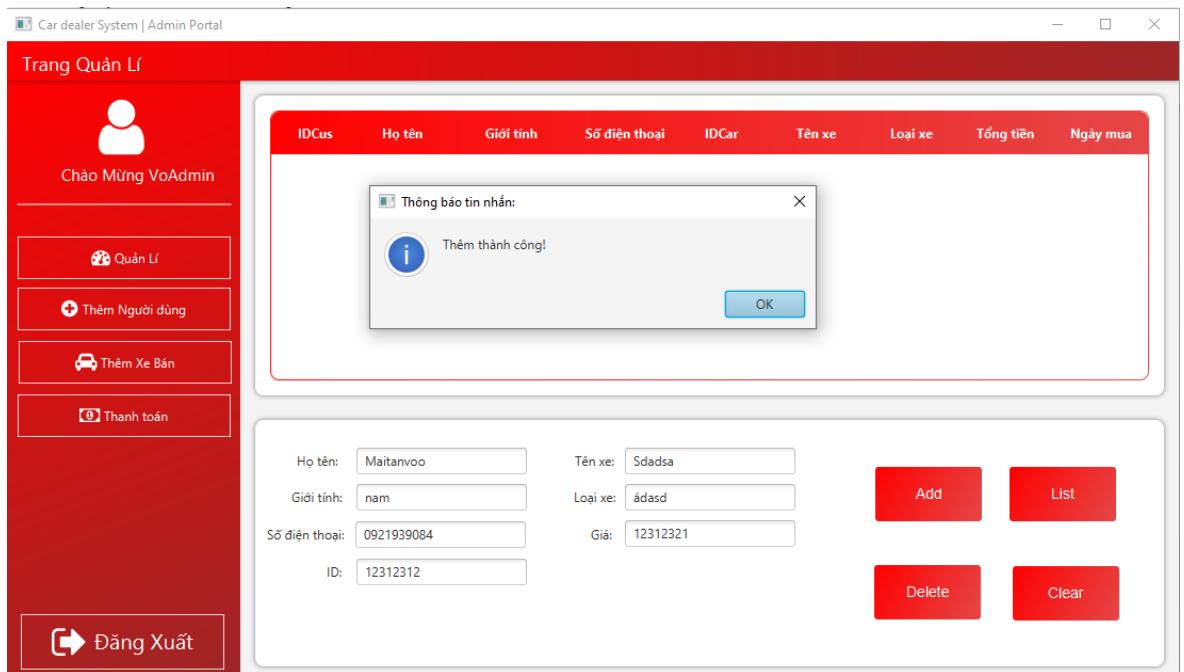
    String sql = "INSERT INTO customer (customer_id, name, gender, phone, car_id, brand, model, total, date)" + "VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?)";
    try {
        if(thanhtoan_name.getText().isEmpty()
            || thanhtoan_gender.getText().isEmpty()
            || thanhtoan_phone.getText().isEmpty()
            || thanhtoan_carid.getText().isEmpty()
            || thanhtoan_carname.getText().isEmpty()
            || thanhtoan_model.getText().isEmpty()
            || thanhtoan_gia.getText().isEmpty()){
            alert.thatbai( tinhnhan: "Vui lòng điền đủ thông tin.");
        }else {
            String sellerNum = "4622";
            int temp_cusID = Integer.parseInt(sellerNum) + cusIDGenerator();
            prepare = connect.prepareStatement(sql);
            prepare.setString( parameterIndex: 1, String.valueOf(temp_cusID));
            prepare.setString( parameterIndex: 2, thanhtoan_name.getText());
            prepare.setString( parameterIndex: 3, thanhtoan_gender.getText());
            prepare.setString( parameterIndex: 4, thanhtoan_phone.getText());
            prepare.setString( parameterIndex: 5, thanhtoan_carid.getText());
            prepare.setString( parameterIndex: 6, thanhtoan_carname.getText());
            prepare.setString( parameterIndex: 7, thanhtoan_model.getText());
            prepare.setString( parameterIndex: 8, thanhtoan_gia.getText());
            Date date = new Date();
            java.sql.Date sqlDate = new java.sql.Date(date.getTime());

            prepare.setString( parameterIndex: 9, String.valueOf(sqlDate));
            prepare.executeUpdate();
            alert.thanhcong( tinhnhan: "Thêm thành công!");
        }
    }catch(Exception e){
        e.printStackTrace();
    }
}
```

Hình 64. Sự kiện xử lý thêm thanh toán



Hình 65. Giao diện thông báo nhập không đủ thông tin để thêm thanh toán



Hình 66. Giao diện thông báo thêm thanh toán thành công

6.2.2. Sự kiện liệt kê khách hàng

```

public void ThanhToanListBtn() {
    addcustomerDisplayData();
    setFieldsTT();
}

public ObservableList<customerData> addcustomerGetData(){
    ObservableList<customerData> listdata = FXCollections.observableArrayList();
    String ten = thanhtoan_name.getText().toString();
    if (!ten.isEmpty()){
        String selectData = "SELECT * FROM customer WHERE name = '" + ten + "'";
        connect = Database.connectDB();
        customerData cusData;

        try{
            prepare = connect.prepareStatement(selectData);
            result = prepare.executeQuery();
            while (result.next()) {
                cusData = new customerData(result.getInt( columnLabel: "customer_id"),
                    result.getInt( columnLabel: "car_id"),
                    result.getString( columnLabel: "brand"),
                    result.getString( columnLabel: "gender"),
                    result.getString( columnLabel: "name"),
                    result.getInt( columnLabel: "phone"),
                    result.getString( columnLabel: "model"),
                    result.getDouble( columnLabel: "total"),
                    result.getDate( columnLabel: "date"));
                listdata.add(cusData);
            }
        }catch (Exception e){
            e.printStackTrace();
        }
    }
    return listdata;
}

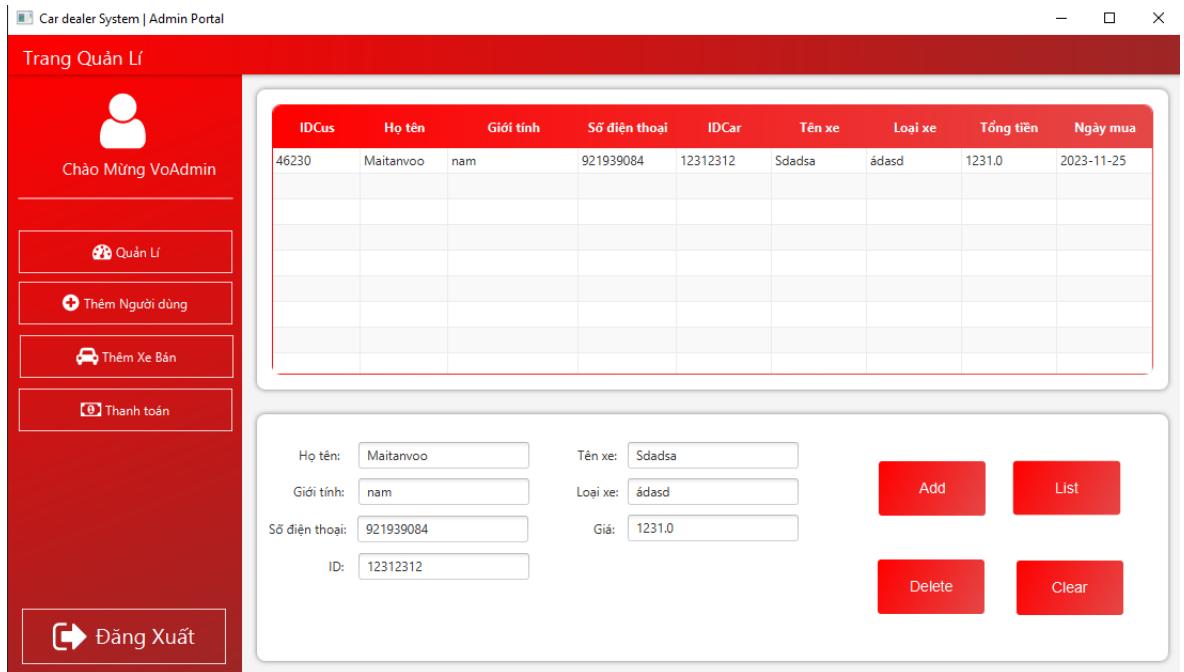
```

```

2 usages
private ObservableList<customerData> addcustomerListData;
3 usages
public void addcustomerDisplayData(){
    addcustomerListData = addcustomerGetData();
    thanhtoan_col_customer_id.setCellValueFactory(new PropertyValueFactory<>( s: "customerID"));
    thanhtoan_col_carid.setCellValueFactory(new PropertyValueFactory<>( s: "carId"));
    thanhtoan_col_brand.setCellValueFactory(new PropertyValueFactory<>( s: "brand"));
    thanhtoan_col_gender.setCellValueFactory(new PropertyValueFactory<>( s: "gender"));
    thanhtoan_col_name.setCellValueFactory(new PropertyValueFactory<>( s: "name"));
    thanhtoan_col_phone.setCellValueFactory(new PropertyValueFactory<>( s: "phone"));
    thanhtoan_col_model.setCellValueFactory(new PropertyValueFactory<>( s: "model"));
    thanhtoan_col_total.setCellValueFactory(new PropertyValueFactory<>( s: "price"));
    thanhtoan_col_date.setCellValueFactory(new PropertyValueFactory<>( s: "date"));
    thanhtoan_tableView.setItems(addcustomerListData);
    setFieldsTT();
}

```

Hình 67. Sự kiện xử lý hiển thị danh sách các thanh toán



Hình 68. Giao diện liệt kê khách hàng

6.2.3. Sự kiện xóa thanh toán của khách hàng

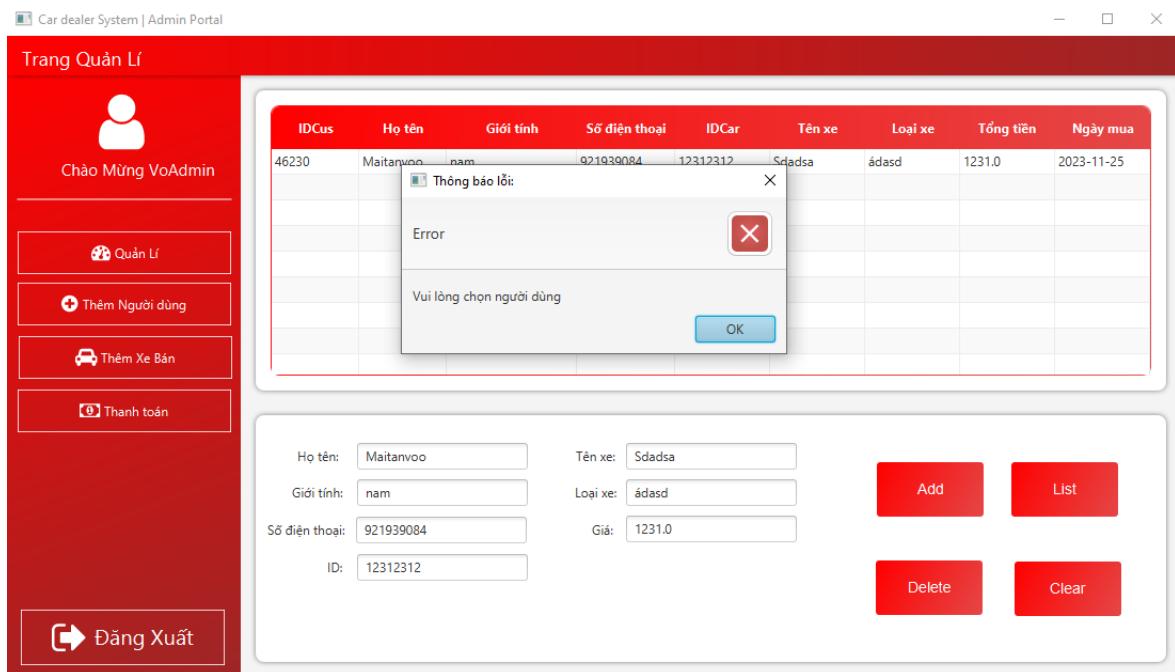
```
public void thanhtoanDeleteBtn() {
    customerData cusData = thanhtoan_tableView.getSelectionModel().getSelectedItem();
    int num = thanhtoan_tableView.getSelectionModel().getSelectedIndex();
    if ((num - 1) < -1) {
        alert.thatbai( tinhhan: "Vui lòng chọn người dùng");
        return;
    } else {
        listdata.temp_customerID = cusData.getCustomerID();
        listdata.temp_TTName = cusData.getName();
        listdata.temp_TTGT = cusData.getGender();
        listdata.temp_TTPhone = cusData.getPhone();
        listdata.temp_carID = cusData.getCarId();
        listdata.temp_carName = cusData.getBrand();
        listdata.temp_model = cusData.getModel();
        listdata.temp_gia = cusData.getPrice();
        if (alert.xacnhann( tinhhan: "Bạn có chắc chắn muốn xóa chứ: " + listdata.temp_TTName)) {
            try {
                // Tạo lệnh SQL DELETE
                String deleteData = "DELETE FROM customer WHERE customer_id = ?";

                // Chuẩn bị và thực thi lệnh SQL DELETE
                prepare = connect.prepareStatement(deleteData);
                prepare.setString( parameterIndex: 1, String.valueOf(listdata.temp_customerID));
                prepare.executeUpdate();

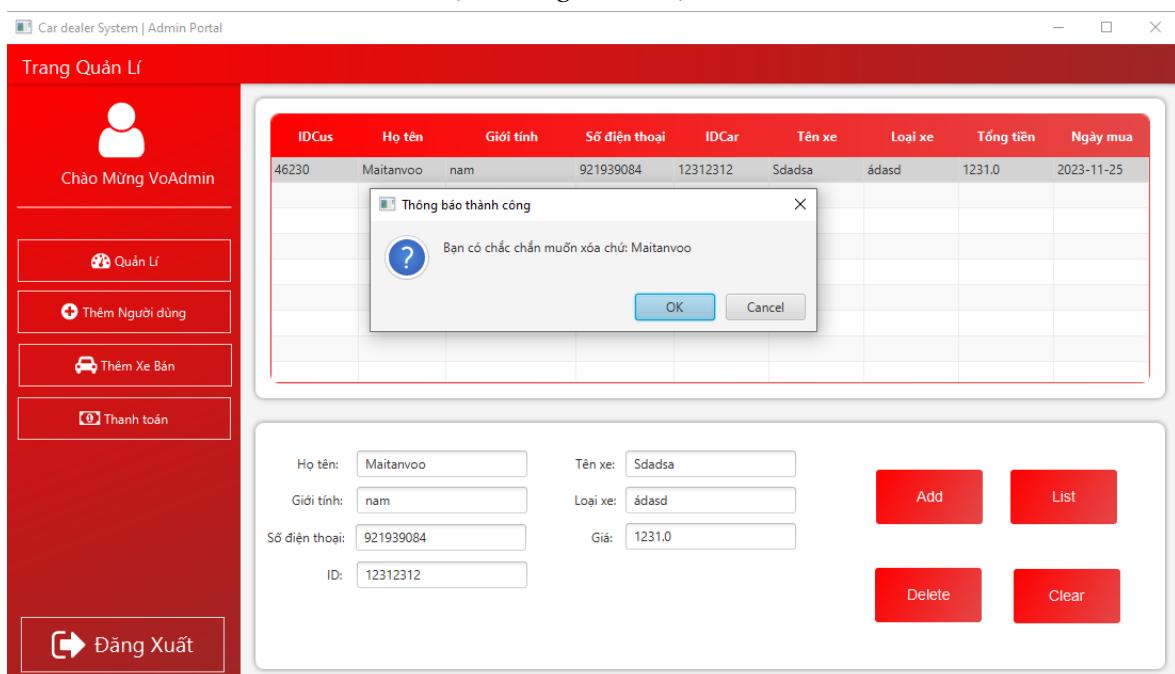
                alert.thanhcong( tinhhan: "Xóa thành công");
                addcustomerDisplayData();

            } catch (Exception e) {
                e.printStackTrace();
                alert.thatbai( tinhhan: "Xóa thất bại");
            }
        } else {
            alert.thatbai( tinhhan: "Hủy");
        }
    }
}
```

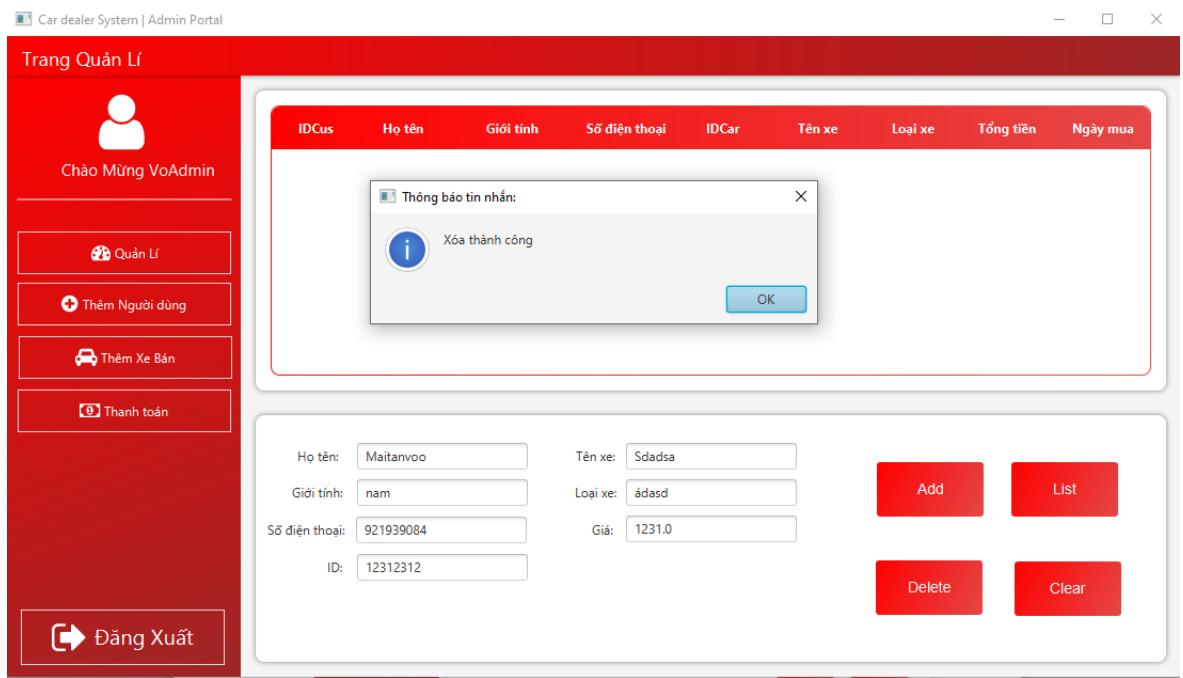
Hình 69. Sự kiện xử lý xóa thanh toán



Hình 70. Giao diện thông báo chọn thanh toán để xóa



Hình 71. Giao diện thông báo xác nhận xóa thanh toán



Hình 72. Giao diện thông báo xóa thành công

CHƯƠNG 4. HƯỚNG DẪN CÀI ĐẶT, ĐÁNH GIÁ, KIỂM THỬ

1. Hướng dẫn cài đặt

1.1. Cài đặt môi trường phát triển

- Download và cài đặt IntelliJ IDEA.

Link download : <https://www.jetbrains.com/idea/download/#section=windows>

- Download và cài đặt JDK 19.

Link download : <https://www.oracle.com/java/technologies/downloads/#jdk19-Windows>

- Download và giải nén JavaFX SDK.

Link download : <https://gluonhq.com/products/javafx/>

- Download và cài đặt MySQL Server và MySQL Workbench.

Link download MySQL Server :

<https://dev.mysql.com/downloads/windows/installer/8.0.html>

Link download MySQL Workbench :

<https://dev.mysql.com/downloads/workbench/>

2. Đánh giá, kiểm thử

Kiểm thử tính khả dụng là kiểm tra ứng dụng có thân thiện với người dùng hay không. Người dùng có thể hiểu ứng dụng dễ dàng hay không:

- Nội dung chính xác, không có bất kỳ lỗi ngữ pháp nào.
- Người dùng dễ dàng sử dụng.

Kiểm thử chức năng là để xác minh sản phẩm có đáp ứng các đặc điểm chức năng, nghiệp vụ được đề cập trong tài liệu đặc tả hay không:

- Sản phẩm đáp ứng các chức năng đã được đề cập trong đ^et^et^e t^ea.

Kiểm thử cơ sở dữ liệu là việc kiểm tra dữ liệu được hiển thị trong ứng dụng có khớp với dữ liệu được lưu trữ trong cơ sở dữ liệu hay không. Dữ liệu thao tác trên ứng dụng có được thêm vào cơ sở dữ liệu một cách chính xác hay không:

- Dữ liệu hiển thị trong ứng dụng khớp với dữ liệu được lưu trữ trong cơ sở dữ liệu.
- Dữ liệu thao tác trên ứng dụng được thêm vào cơ sở dữ liệu là chính xác

III. PHẦN KẾT LUẬN

1. KẾT QUẢ ĐẠT ĐƯỢC

- Hiểu được cách thức hoạt động của các thành phần ứng dụng trong JavaFX
- Tiếp cận đến các công nghệ mới
- Xây dựng được một ứng dụng quản lý cơ bản, hoạt động ổn, không có lỗi

2. HẠN CHẾ

- Chưa liên kết với nhiều cửa hàng với nhau
- Chưa sử dụng trên internet

3. HƯỚNG PHÁT TRIỂN

- Phát triển thêm tính năng liên kết với nhiều cửa hàng với nhau để trao đổi thông tin
- Phát triển ứng dụng có thể sử dụng trên internet
- Phát triển tính năng cho thuê xe

IV. TÀI LIỆU THAM THẢO

Tài liệu về JavaFX:

Link 1: <https://openjfx.io/openjfx-docs/>

Link 2: <https://fxdocs.github.io/docs/html5/>

Link 3: <https://jenkov.com/tutorials/javafx/index.html>

Tài liệu về Scene Builder:

Link 1: <https://howkteam.vn/course/lap-trinh-javafx-co-ban/su-dung-scene-builder-trong-javafx-2646>