



[Car Rental App]

By

[Mwanzia William Munguti]

[656333]

Contents

Chapter 1: INTRODUCTION	4
1. Background of the Application.....	4
1.1. Problem Statement.....	5
1.2. Objectives.....	5
1.2.1. General Objectives.....	5
1.2.2. Specific Objectives	5
1.3. Assumptions.....	6
1.4. Scope.....	6
1.5. Limitations.....	6
Chapter 2: LITERATURE REVIEW	6
2. Introduction.....	6
2.1. Body.....	7
2.1.1. Features of most car rental apps	7
7. Terms and entities.....	7
How the software works	8
Comparison to other similar software	8
Conclusion	8
Chapter 3: DEVELOPMENT METHODOLOGY	8
Advantages of Agile	9
How agile was used	9
Functional and non-functional requirements.....	9

Functional Requirements	9
Non Functional Requirements	10
Design tools	10
Development Tools.....	10
Programming and UI implementation.....	10
Database.....	11
Chapter 4: System Analysis and Design	13
 Use case diagram.....	13
Flowchart	14
 ERD	15
Class Diagrams	16
User class diagram	16
Vehicle class diagram	17
GUI design	18
Splash page	18
Log in page	18
Chapter 5: Implementation and testing	21
Splash page	21
Logging in.....	22
Managing vehicles	25
Editing vehicles.....	27

Managing customer.....	32
.....	34
Customer after login	36
Back end examples	37
.....	38
Conclusion and recommendations	38
References.....	39

Chapter 1: INTRODUCTION

1. Background of the Application

The first car to be hired was traced back in the year 1904, when a bicycle shop in the town of Minneapolis started offering cars for rent. The first car rental company was started in Germany in the year 1912 and was called **SIXT**

The high rise in the demands of renting of cars has brought this to be a pretty good investment opportunity. Renting of a car starts with the customer calling the car renting company and talking to the secretary. The customer enquiries for available cars and the secretary of the company checks the records to see if the exact type is available for renting or a vehicle similar to that is available. When the vehicle is available the secretary will then declare the status of the

vehicle booked. This process sometimes takes a long time since the secretary has to get the callers' information which includes their address and the mode of payment as well.

The result is that in rush hours, mostly weekends, a lot of calls are missed due to the time taken for collection of this data and the possible solution to this was to develop an application that can capture data in minutes and can be done by multiple users. Moreover, the available cars will be categorized and a picture of the vehicle and registration number will be available to the user.

1.1. Problem Statement

The client, owner of **Rent a Ride** car rental business, which has a unique feature where a customer can book a car and it can either be delivered to an agreed location or the customer will come pick it up at the store, has gotten complaints from the regular customers that the calls they make in order to book a car is a bit too expensive. The idea to make booking easier and less expensive for the customers is to come up with a mobile application

Rent a ride is a successful car business and is well known, therefore a mobile application will be ideal. The application will enable user to choose their location from available listed towns, since Rent a Ride has different branches all over the country. All this will be done in a matter of 5 minutes on the app as compared to the long calls where the secretary has to keep searching through the system, which causes time wastage and using up of airtime.

1.2. Objectives

1.2.1. General Objectives

Create a software that enables a user to select a car to rent according to the client's location and choose whether to collect the car or have it delivered at the comfort of their homes.

1.2.2. Specific Objectives

The Application will be able to:

1. List all branches of **Rent a ride** and user will select the location they are located at.
2. List all cars available at a branch
3. Allow user to book a car
4. Give feedback after is booked

1.3. Assumptions

- i. The user is booking a car from the location they will choose
- ii. The user knows the location of the physical office

1.4. Scope

- i. To help users to rent a car
- ii. To enable user to select day, date.
- iii. To enable user to see a list of available cars
- iv. To give feedback to the user

1.5. Limitations

- i. There is no guest user available

Chapter 2: LITERATURE REVIEW

2. Introduction

This chapter analyses in depth the knowledge, research and findings on a car rental management application. The topics that were researched on the application include: the accuracy of the system, the working principle the problems associated with existing systems which include: interface design.

Renting of cars gives people the opportunity to move around even though they do not have their own personal vehicles. It doesn't mean that just because you don't own a vehicle and when you need one you will have to buy it, just rent one instead. The customer who needs the vehicle will contact the company first. Then the customer will be asked to give their information that will be required by the company to give proof that you are an actual person and you are not just trying to swipe a car. If the customer has a particular request on the type of vehicle, he wants he will say and if available he will book it. If the vehicle requested is not available, the secretary will give the other available options in category then the booking on the car will be made.

This is a tedious process and takes roughly 10 minutes, why not do it in one? The development of an app that can do all the processes done by the secretary will save people a whole lot of time and money (phone bills).

2.1. Body

2.1.1. Features of most car rental apps

1. Easy to use interface- according to (M, 2021), it doesn't matter which car is up for rent if the customer is not able to communicate with the company. The company is the software that represents the company. Examples to make the software easy to use is given below.
 - the features should be neatly stacked not to confuse the customer
 - The booking, cancelation, scheduling should be easy
 - easy booking system (models, pricing plans, special offer)
2. Real time monitoring- most software have a gps system that monitors the car's movement and makes sure the car is safe and operating alright (M, 2021). The real time monitoring will tell the status of the engine and will be able to send back the condition of the car to the admin.
3. Chat bots- the chat bots are helpers that make sure the user understands areas that they may not be able to operate properly and the bots also answers frequently asked questions (M, 2021).
4. Security- Some car rental software has the ability to encrypt some personal information therefore keeping data safe from the hands of any hackers.
5. Input data- this software enable a user to enter their personal data which is then stored into the database and will recognize the user the next time they use the software again
6. Linked to the cloud- they are linked to the cloud that way the information will be updating every time and will be easily accessible to the users

7. Terms and entities

Admin

The admin is the manager of the application and will be able to see all the cars that exist.

Moreover, the admin will be able to use the application to generate reports on what is going on.

User

The user of the application in this point will be the customer of **Rent a Ride** and will be using the software to book a specific car. There are existing users and also there are new users who want to rent a car.

How the software works

The user will start the app and will choose between client or admin. Assuming the user is an existing client they will log in with pre-existing details and the home page will be visible to the user. The home page is where you select the town you are located at, then the app will move to a different page that displays the available cars, in that town and then one will click on the book button which will open a form, the user will fill this form and it will be submitted to the database. The user will then get feedback that the car has been booked and the history page will show the car booked and when it was booked.

On the admin side the admin will be able to view the number of cars that have been booked and will be able to see the total amount earned from the lending of cars to customers.

Comparison to other similar software

1. Input- Rent a Ride will be able to capture data from the user and will be able to store it into the database of the software.
2. Output- Rent a Ride will be able to output data from the database and the user will be able to see information.
3. Authentication- the log in of the customer will enable the authentication of the user, if information is not in the database then the user will be prompted to sign up
4. Feedback- when the user books a car the application will give feedback that the car has been successfully booked.

Conclusion

A car rental software can have a lot of functionality and features. The type of application that you want will determine the functionalities and features that one wants, and the more the features the more complex the application is. In conclusion the scope of the app will determine the features to be included in software development time.

Chapter 3: DEVELOPMENT METHODOLOGY

In the development of this software I used agile approach. Agile development is a method of development that involves the gradual development of something that allows a developer to make changes whenever they want to. With the iterative development of the project. The

development can be done in bits or can also be done at different timings therefore the development process is affected. Because the app may have milestones, the agile development will work perfectly and will cater for changes in the app given by the client.

Advantages of Agile

1. It was cost effective- it is known to save money and also helps the investing of funds to be done properly.
2. It was flexible since it was able to accommodate the changes that were taking place during the whole development process
3. The quality of the product is usually high- when agile development is used, the app will most likely be of high quality.

How agile was used

The development of the software was gradual development of the app. The method of development will be most suitable when catering for changes within the app or also with the functionality of the application.

Functional and non-functional requirements

The requirements of an application are the various features that will make the app function properly and will be able to help the user achieve their objectives.

Functional Requirements

1. The user will be able to choose between existing user and admin.
2. The user will have to be authenticated when they log in.
3. If user is not existing, they will have to sign up entering their personal details.
4. The software should be able to store and retrieve data in a database.
5. The user should be able to select the location they are in from the listed ones.
6. The software should be able to show the list of available cars in categories
7. The software should be able to give back feedback when a customer chooses their vehicle of choice
8. The software should be able to show the user/ customer the history of previously rented cars.

9. The software should generate reports on the data to the admin.

Non Functional Requirements

1. Operating system- the software will be a desktop software
2. Usability- the software will be easy to use by people of all ages and will be self-explanatory

Design tools

The design tool that was used in the development of the software was a trial and error approach with the dragging and dropping of the objects into the IDE and trying to identify how they will best be presentable. The IDE used for this process was **APACHE NETBEANS**. The IDE was also used in the development of the software and also testing.

Development Tools

Programming and UI implementation

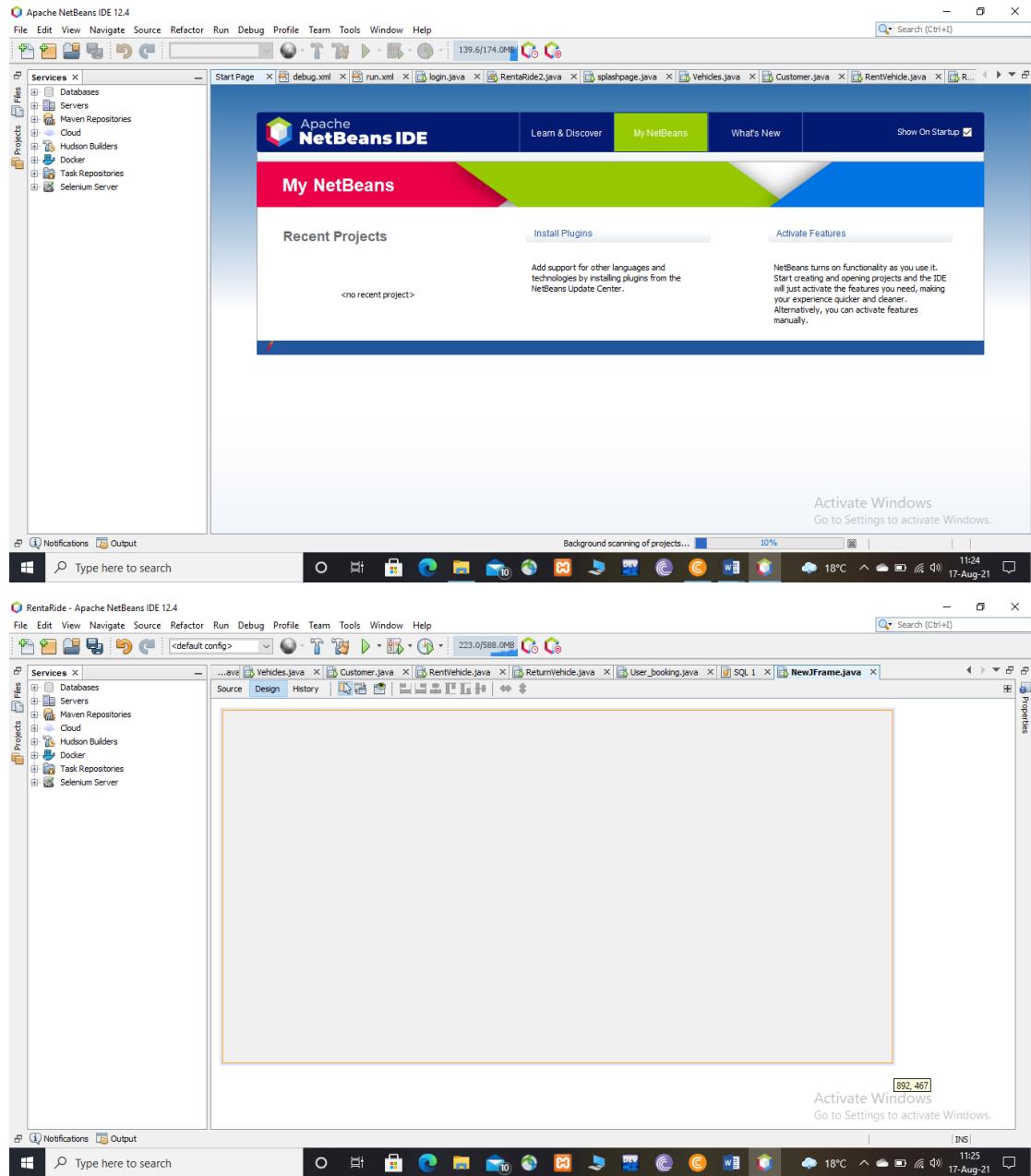
The programming process was also done using the APACHE NETBEANS ide. Java language is the language that was used in the implementation of this software. Java language was preferred since it is object oriented and the code can be re-used in some areas that have similar functionalities.

Reason for choosing the development tool

The reason for choosing APACHE NETBEANS IDE for development is because it is a generally light IDE that supports the development of software in java programming language. Moreover it also enables the use of a user interface that one designs according to their own wants and need. More over the pallets are customizable and one can manipulate them however they wish them to be.

How it was used

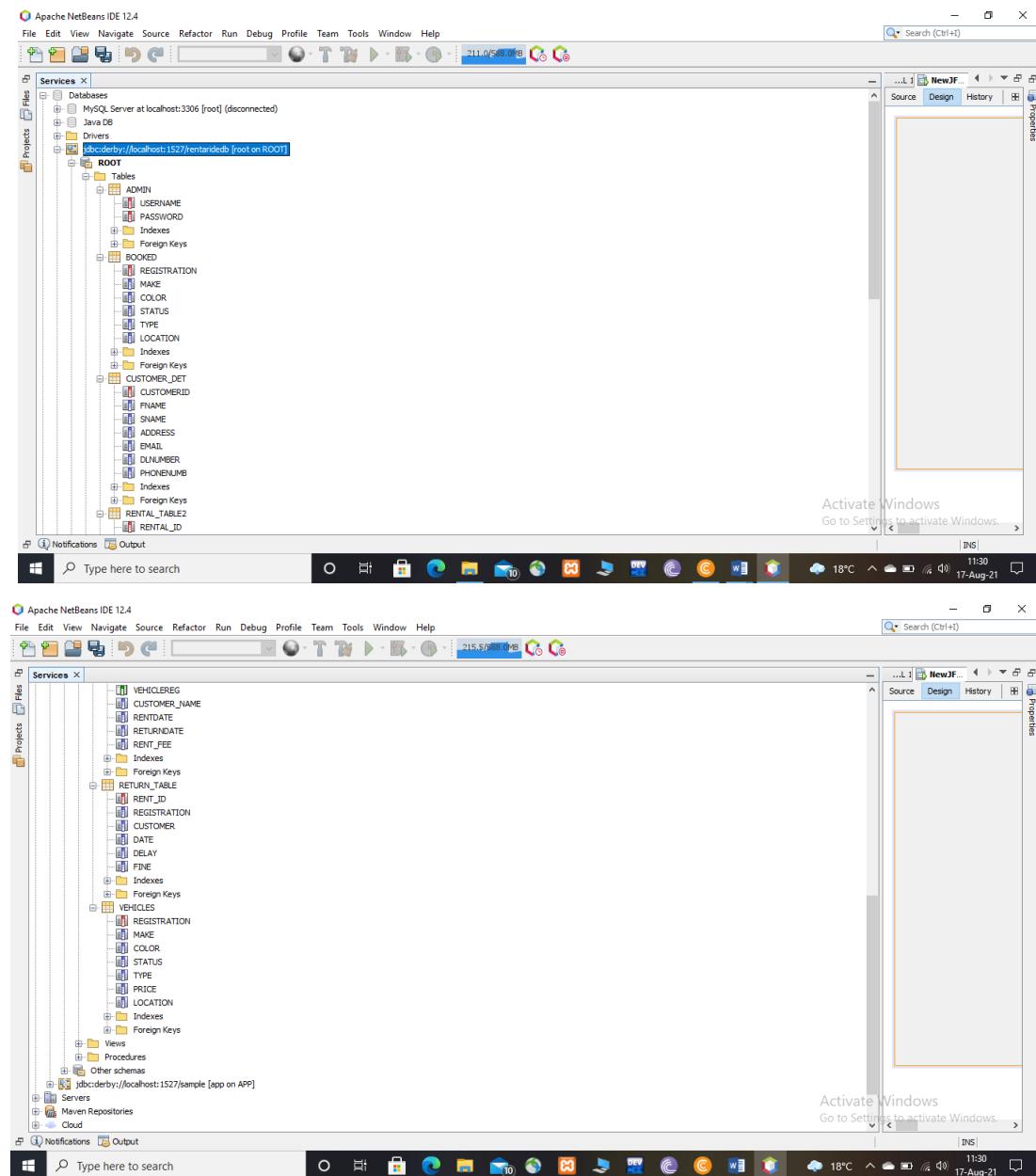
APACHE NETBEANS was used in the implementation of the software in the back end and also the designing of the user interface.



Database

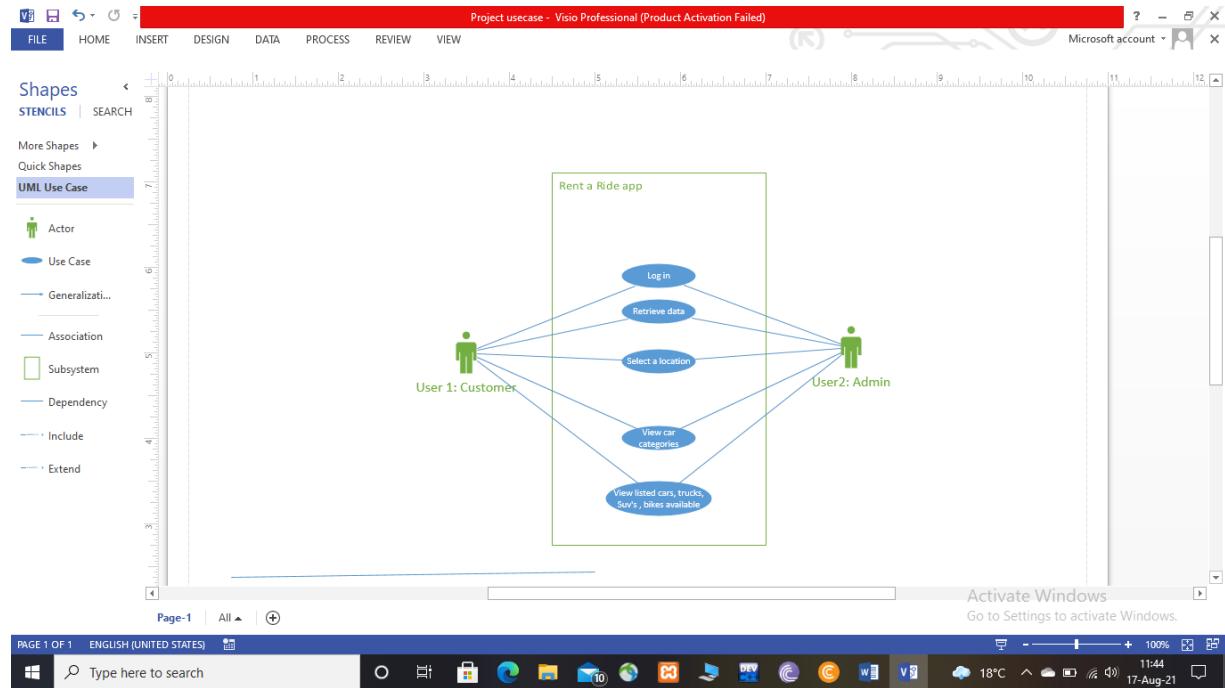
The database that was used in the implementation of this software was the derby database that is an internal database located within the IDE environment. Derby is a simple database that allows the entry and saving of data. The data is also retrieved using SQL queries.

How it was used

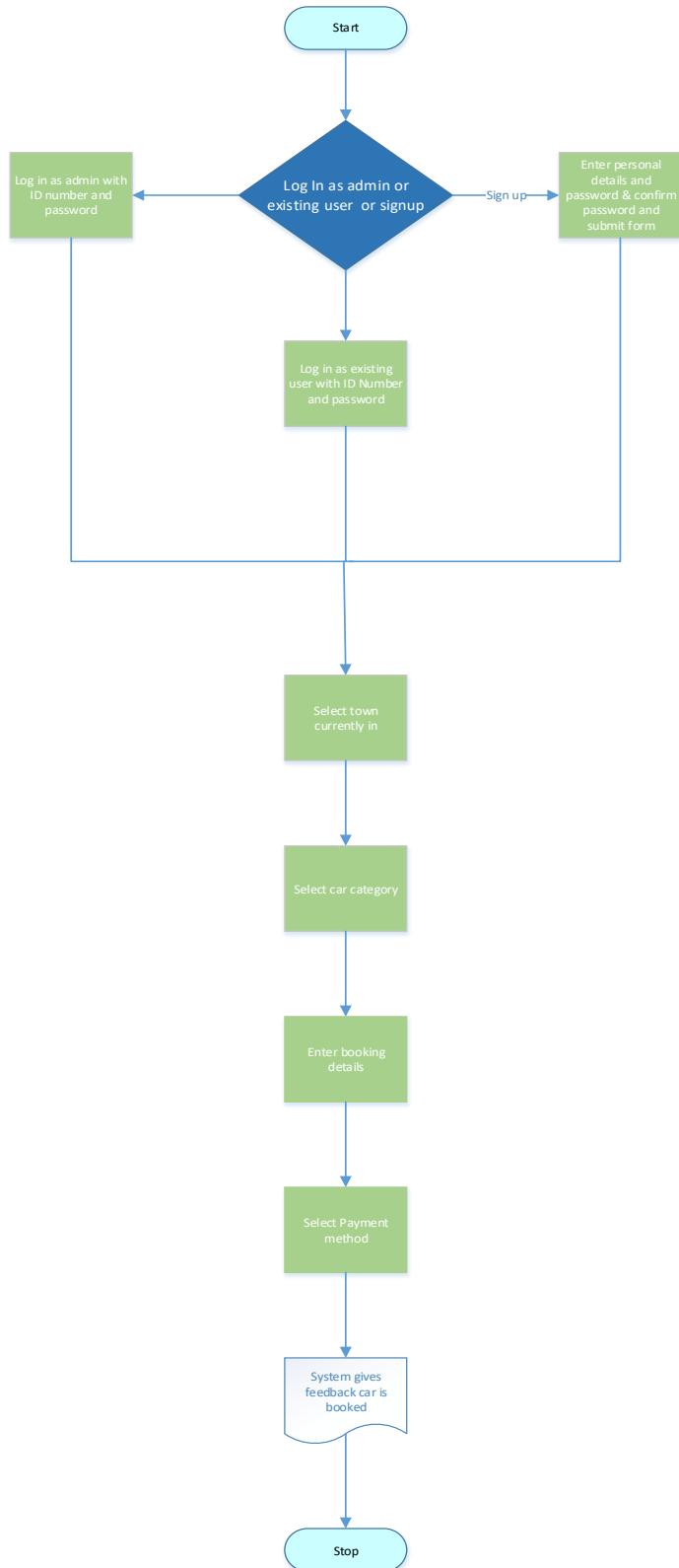


Chapter 4: System Analysis and Design

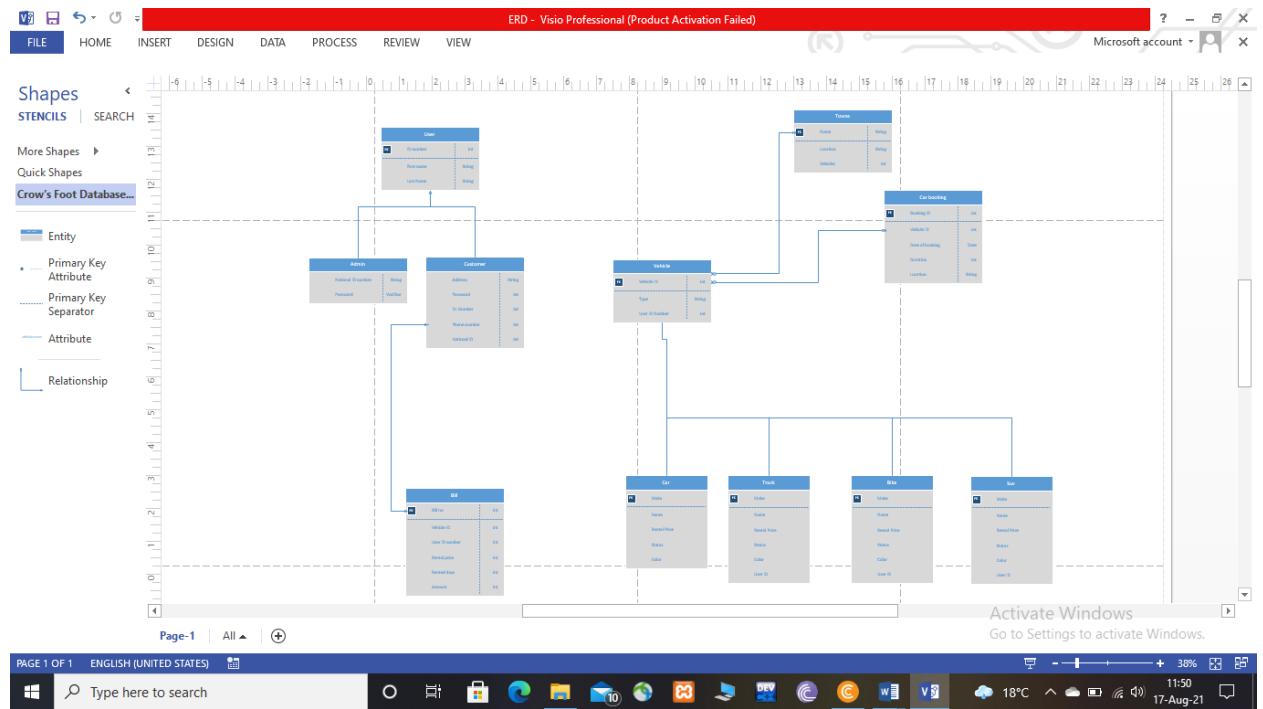
Use case diagram



Flowchart

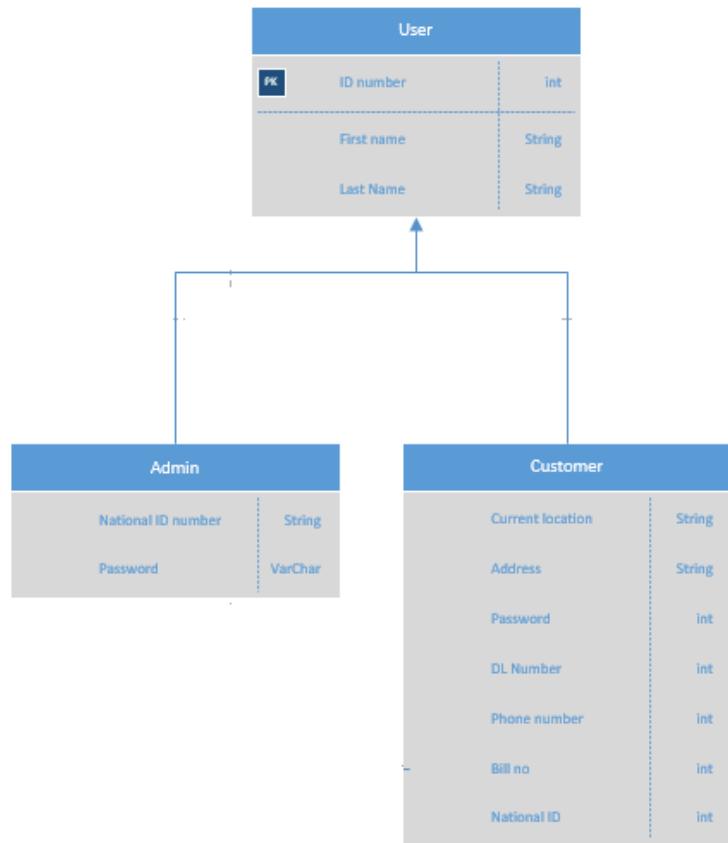


ERD

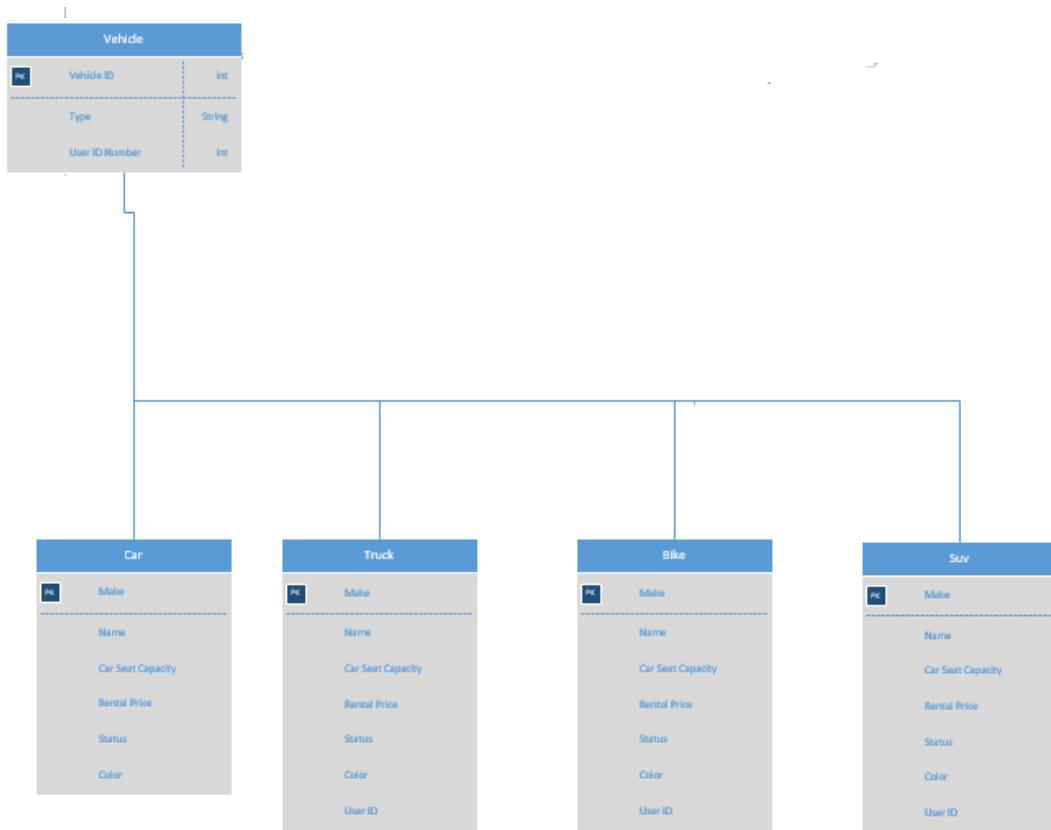


Class Diagrams

User class diagram



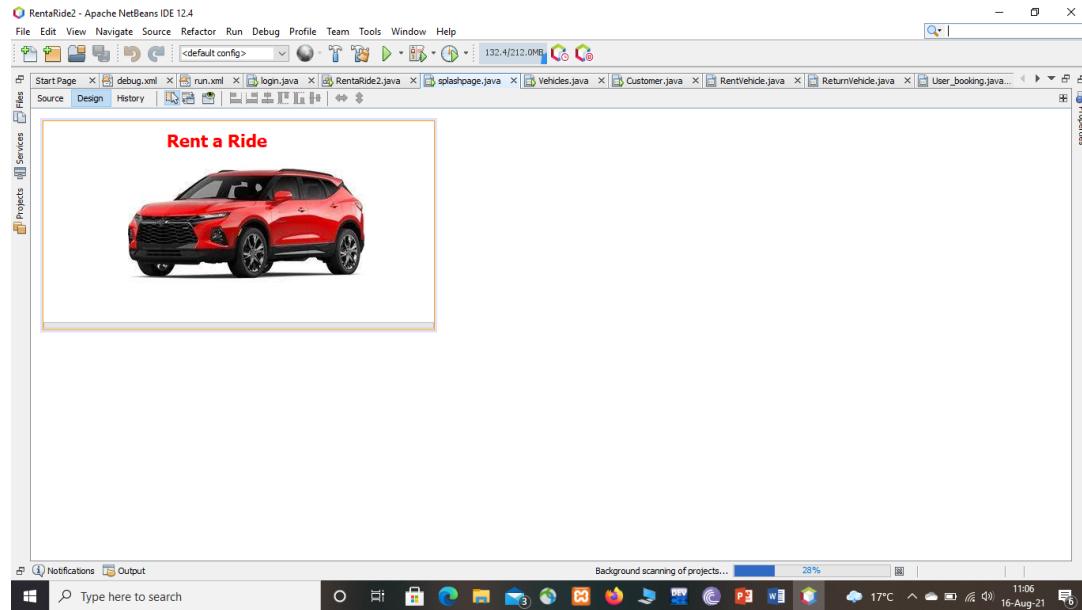
Vehicle class diagram



GUI design

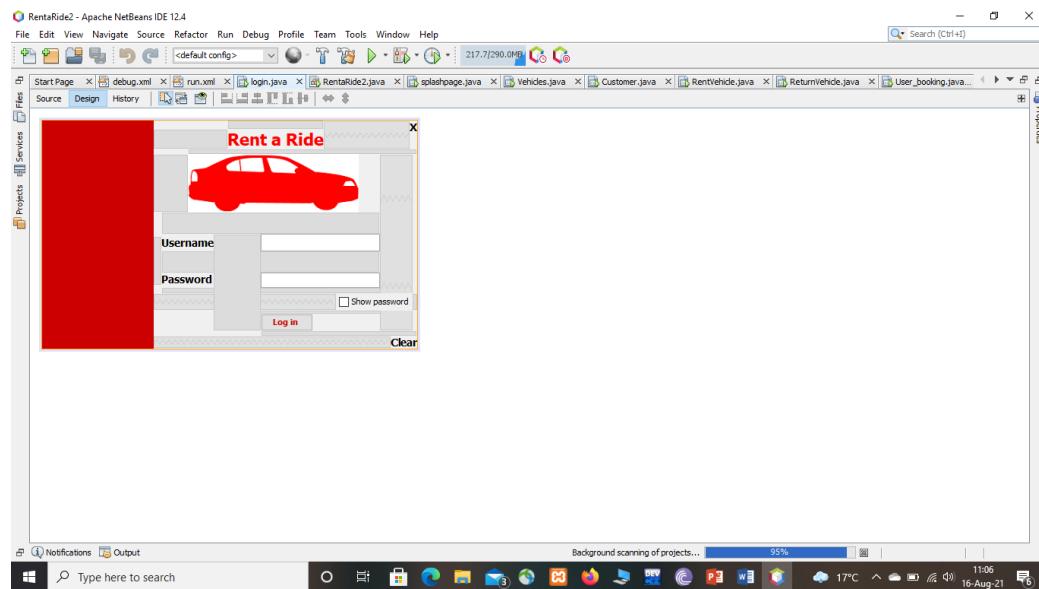
Splash page

This is the page that will start up the software



Log in page

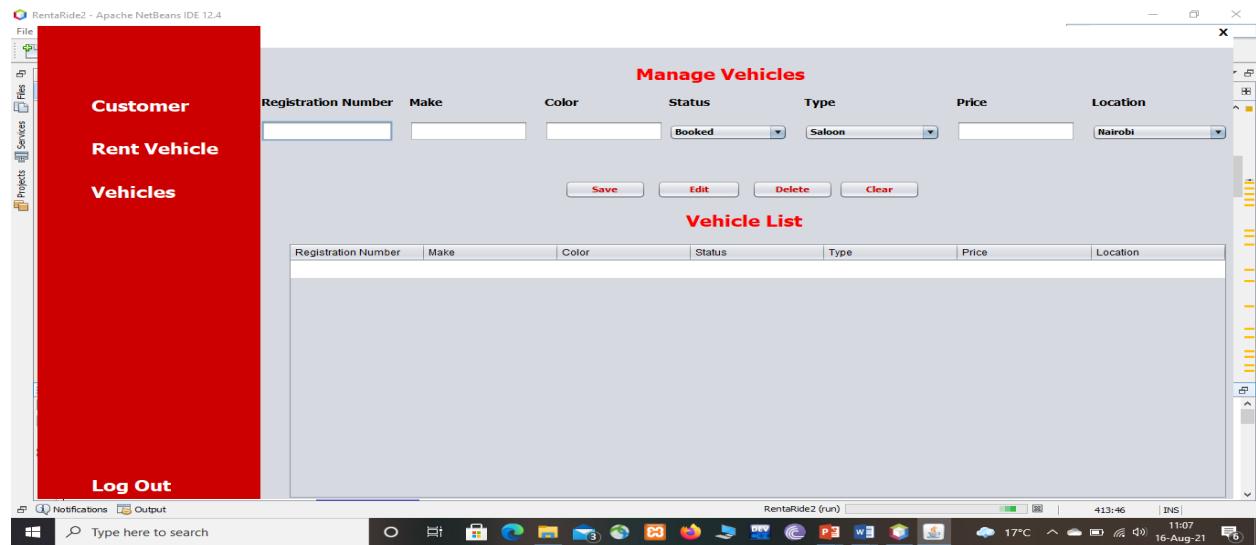
This is the UI that will be displayed after the splash page finishes loading to enable user to log in.



After Admin login

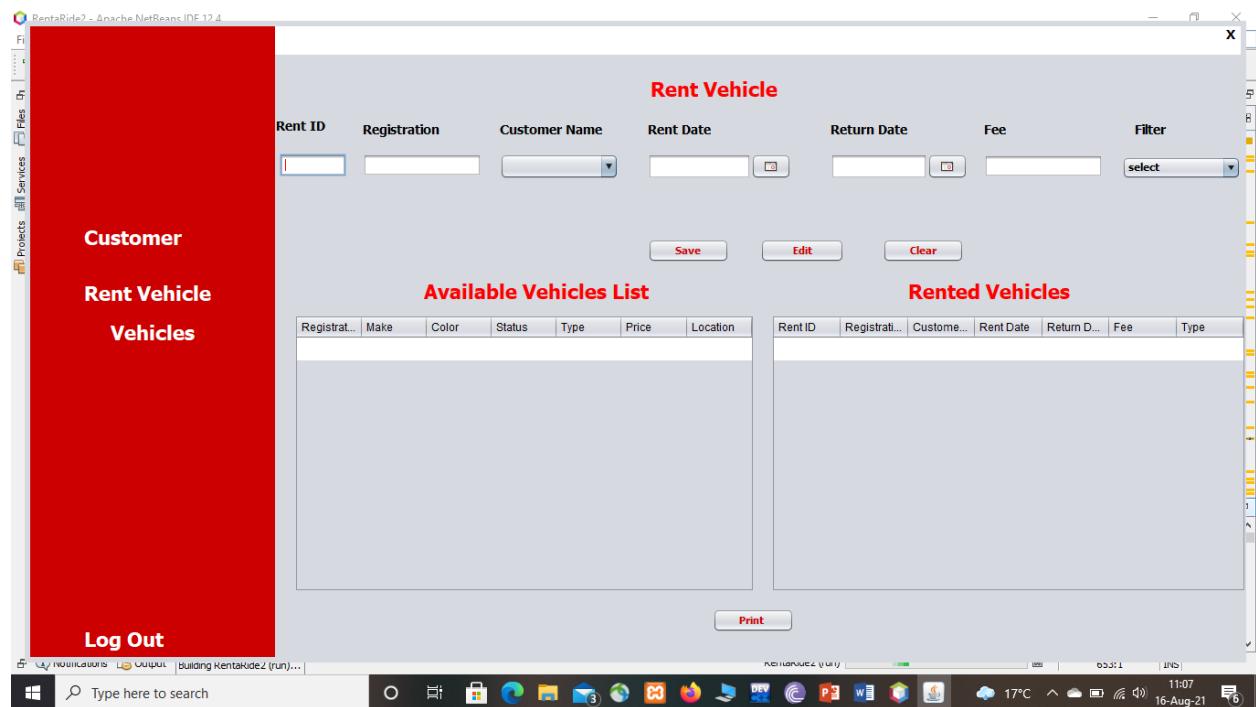
Manage vehicles

This page allows the user (admin) to add, delete and edit the vehicles that are going to be in the system.



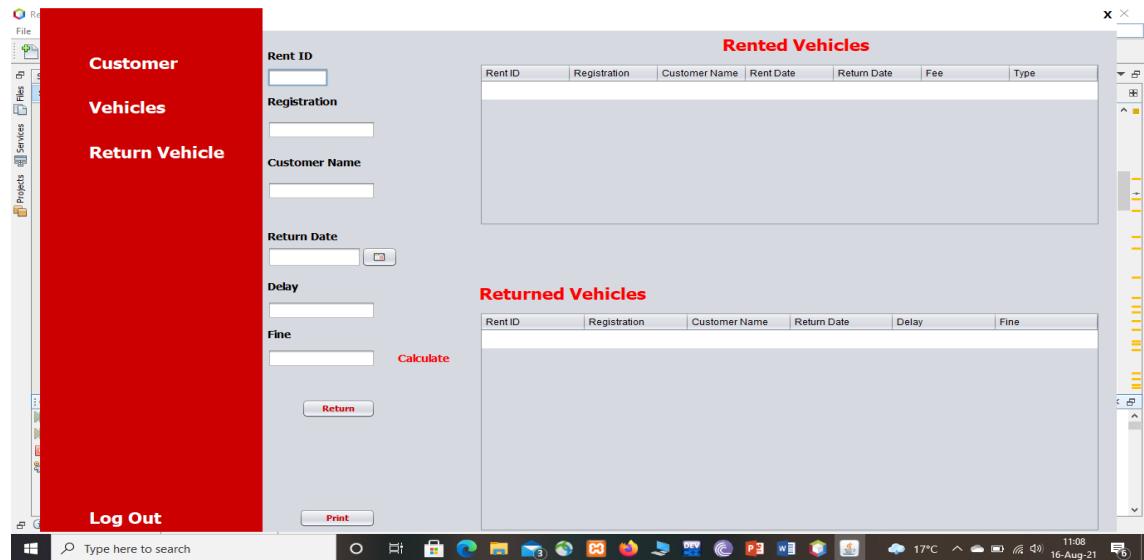
Rent vehicles

to customers already in the system



Return vehicle

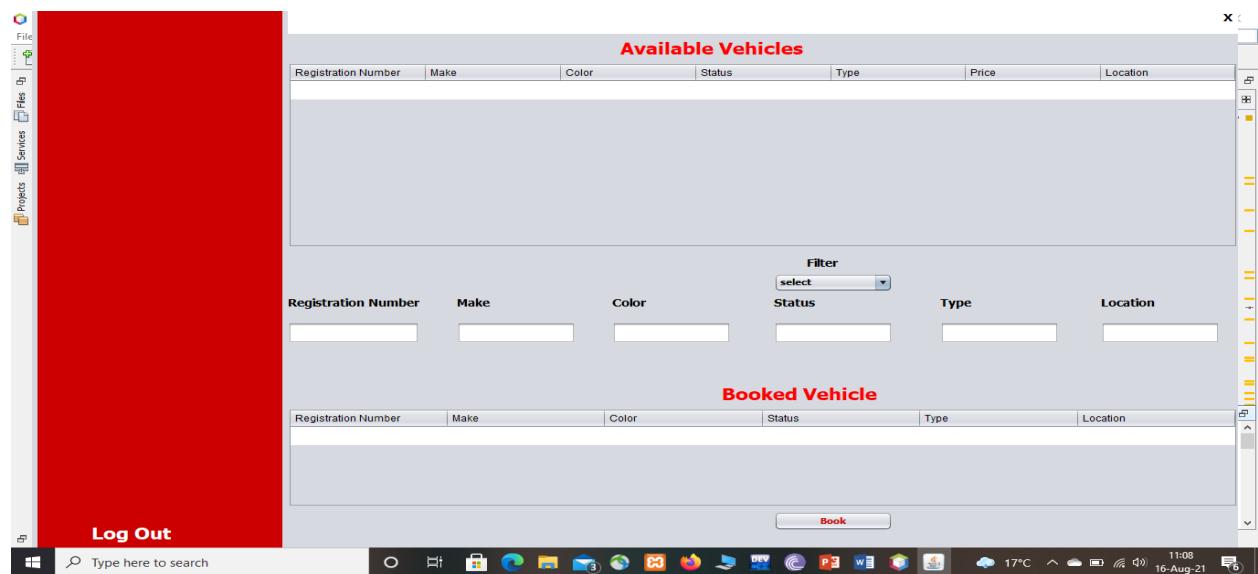
This page allows the admin to be able to have a record of all the returned vehicles and apply a fine where the vehicle has passed the supposed return date



After customer login

Book vehicle

This module allows the user that logged in as customer to book the vehicle from an already existing list entered by the admin.

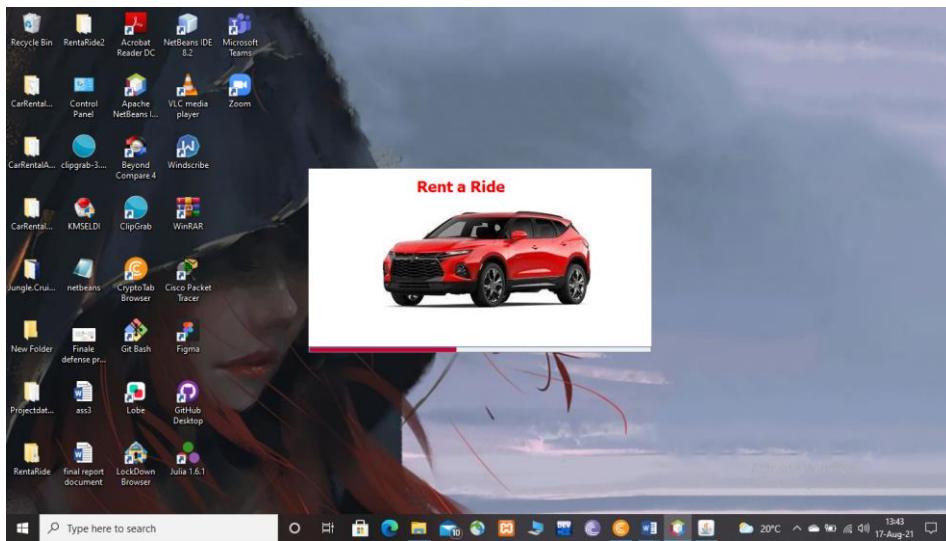


Chapter 5: Implementation and testing

The implementation of the software was done using the apache NetBeans and the back end was implemented using java programming language

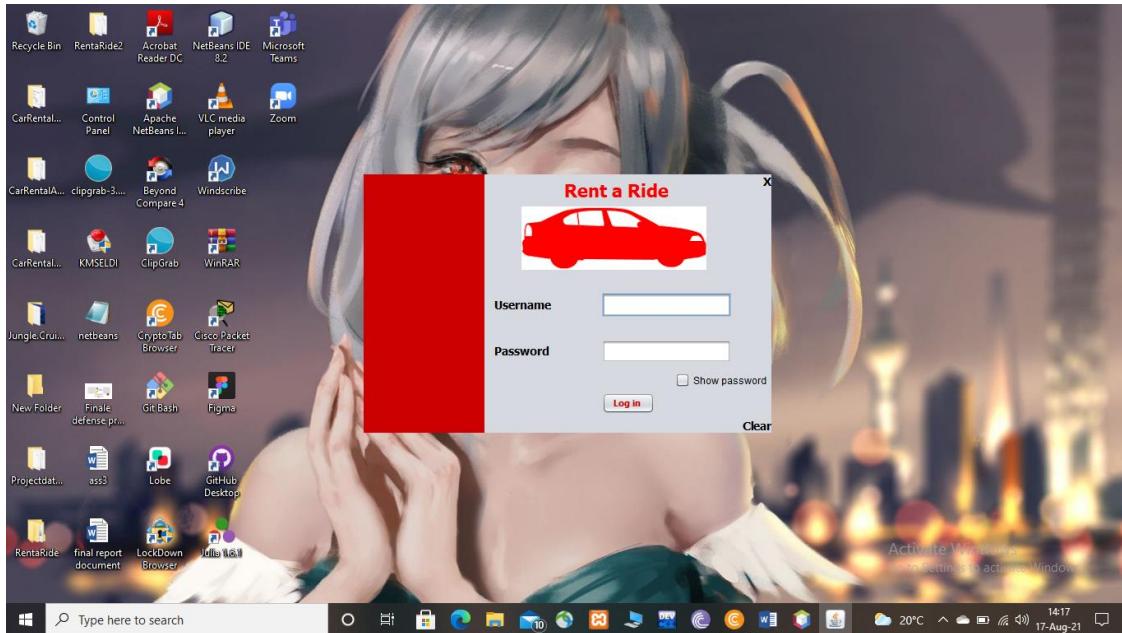
Splash page

The splash page loading

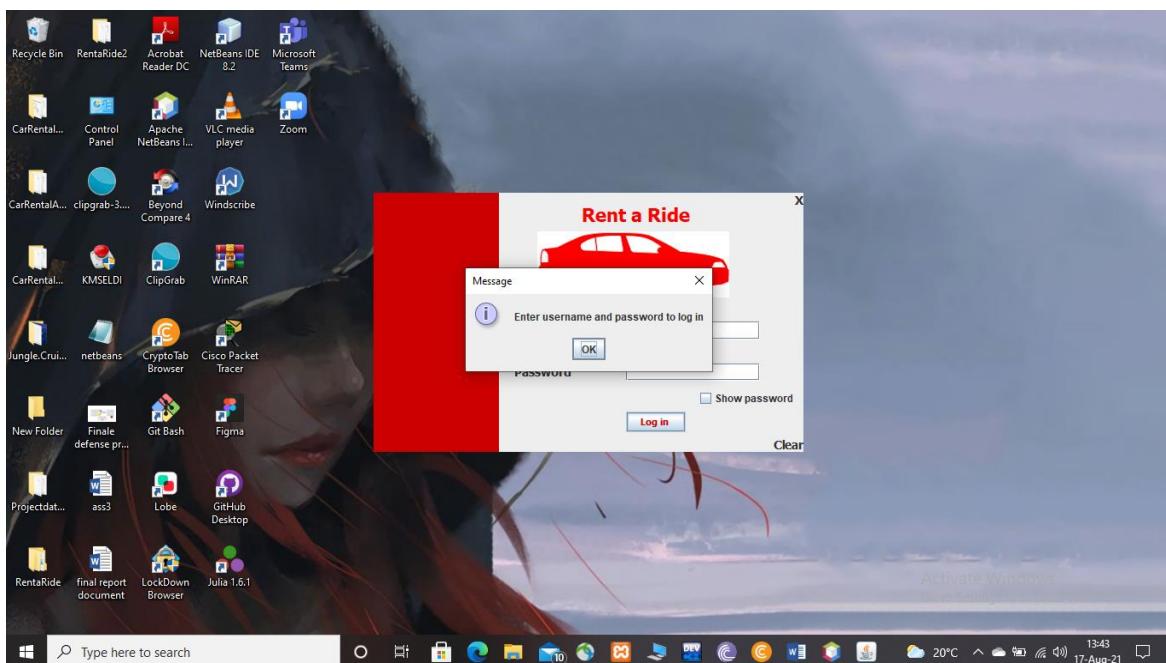


Logging in

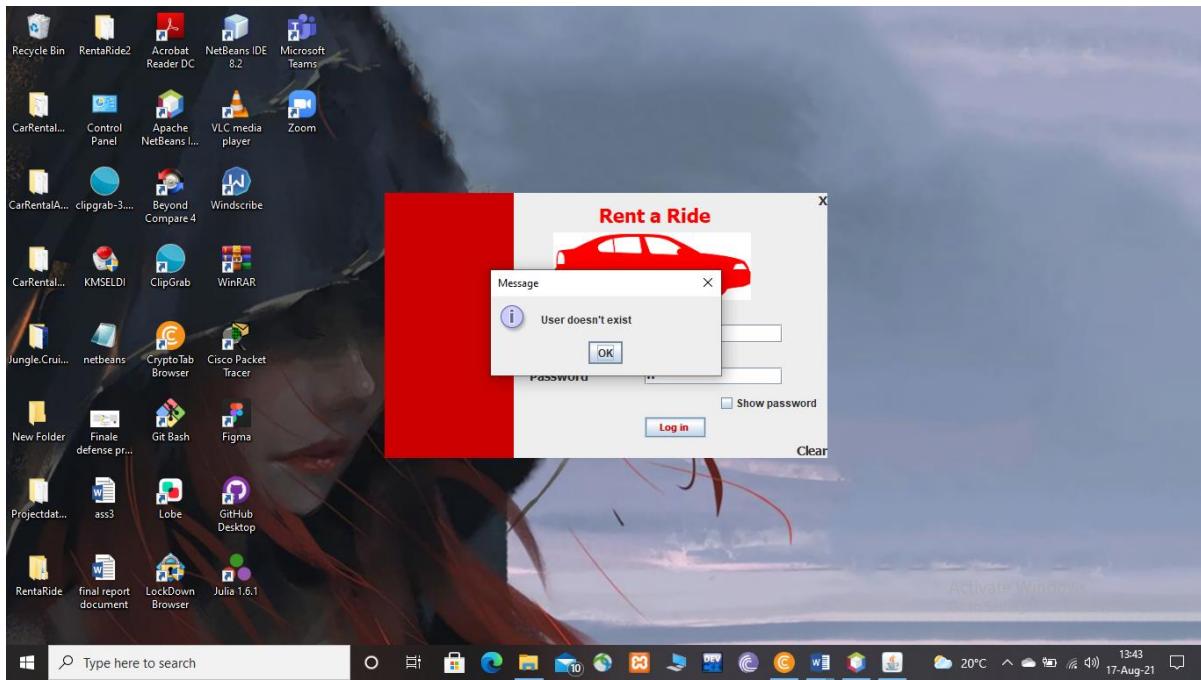
Log in page



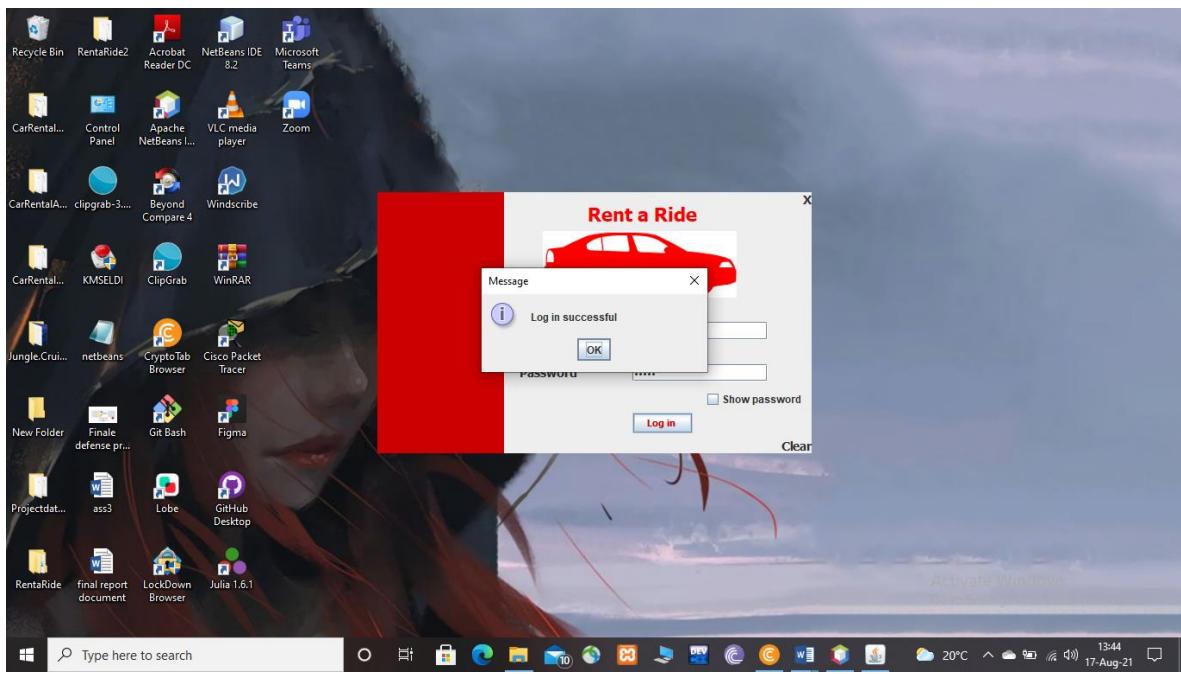
When user presses the log in button without any details present



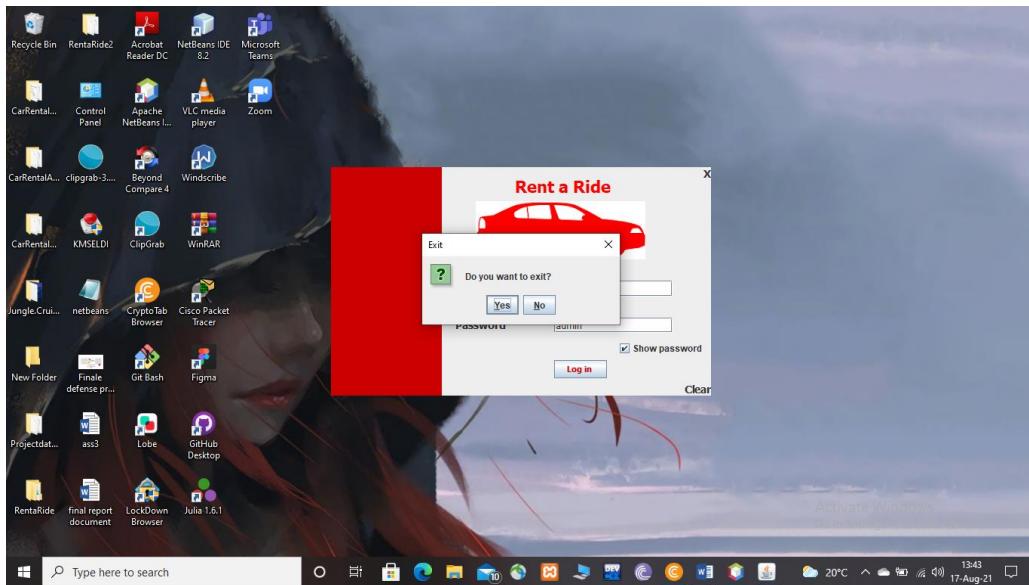
When the details entered don't exist in the database



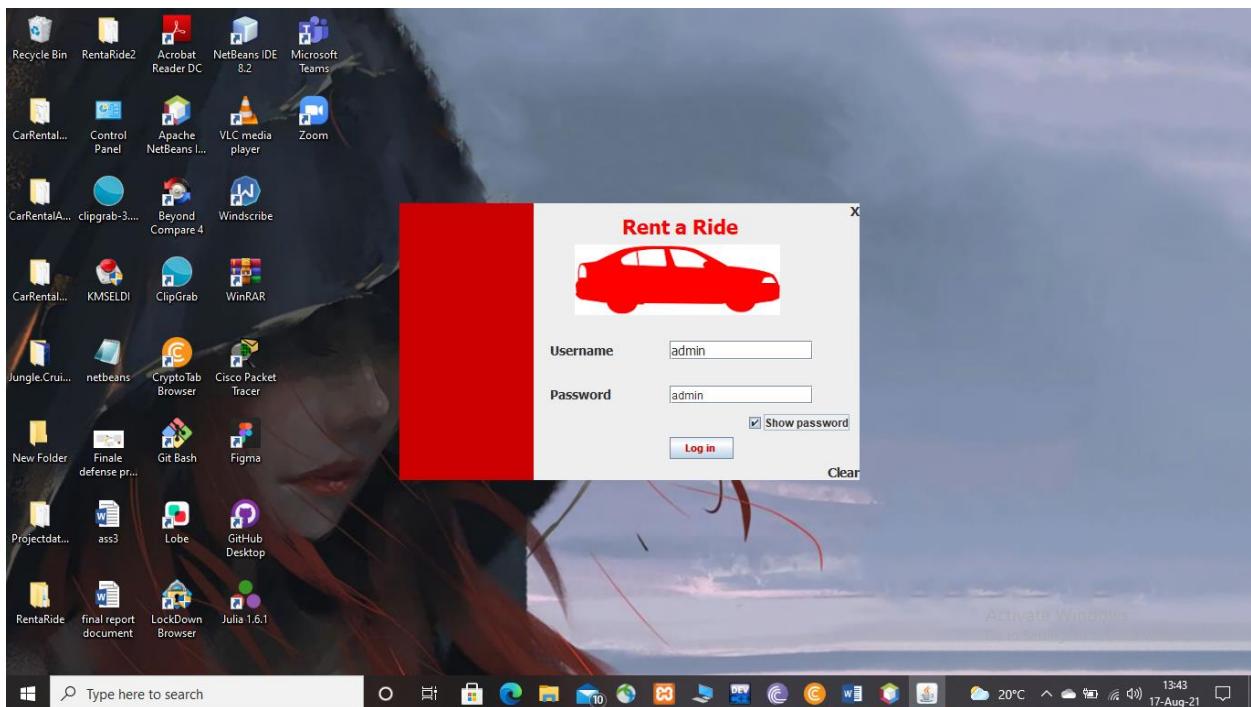
When details entered are in the database and user is verified



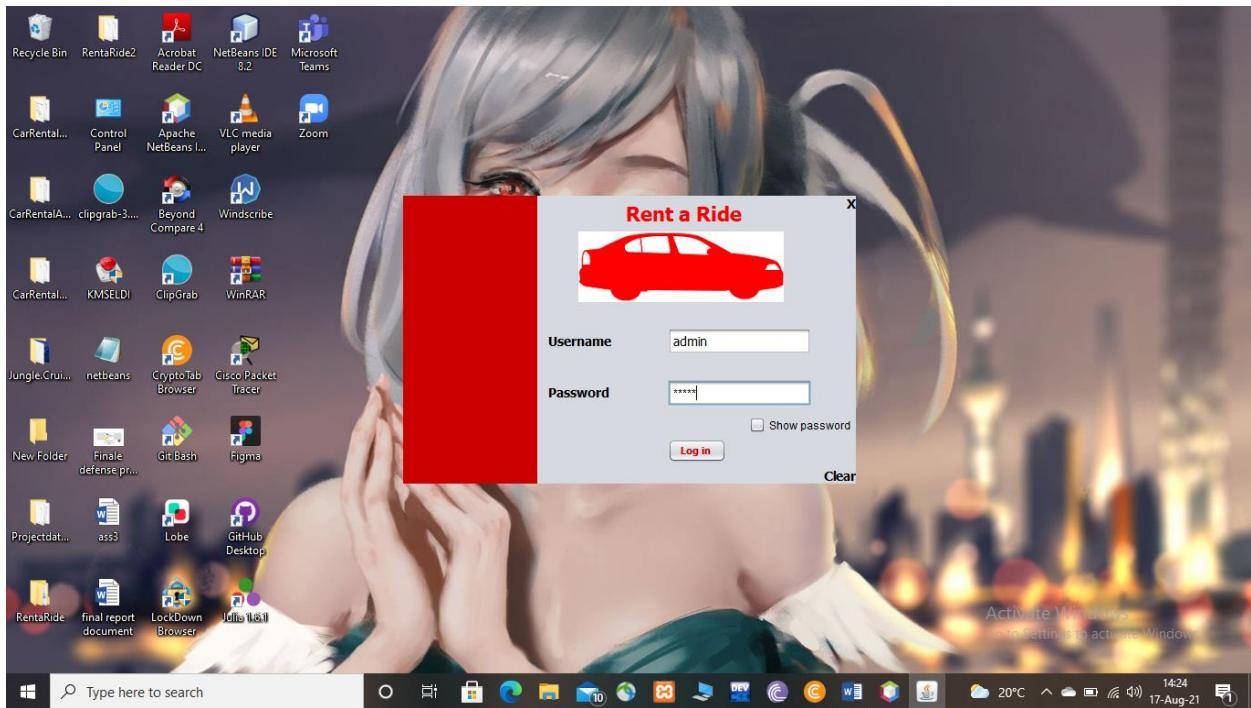
When user clicks on the close button confirmation message is displayed



When the show password is selected



When show password checkbox is not selected



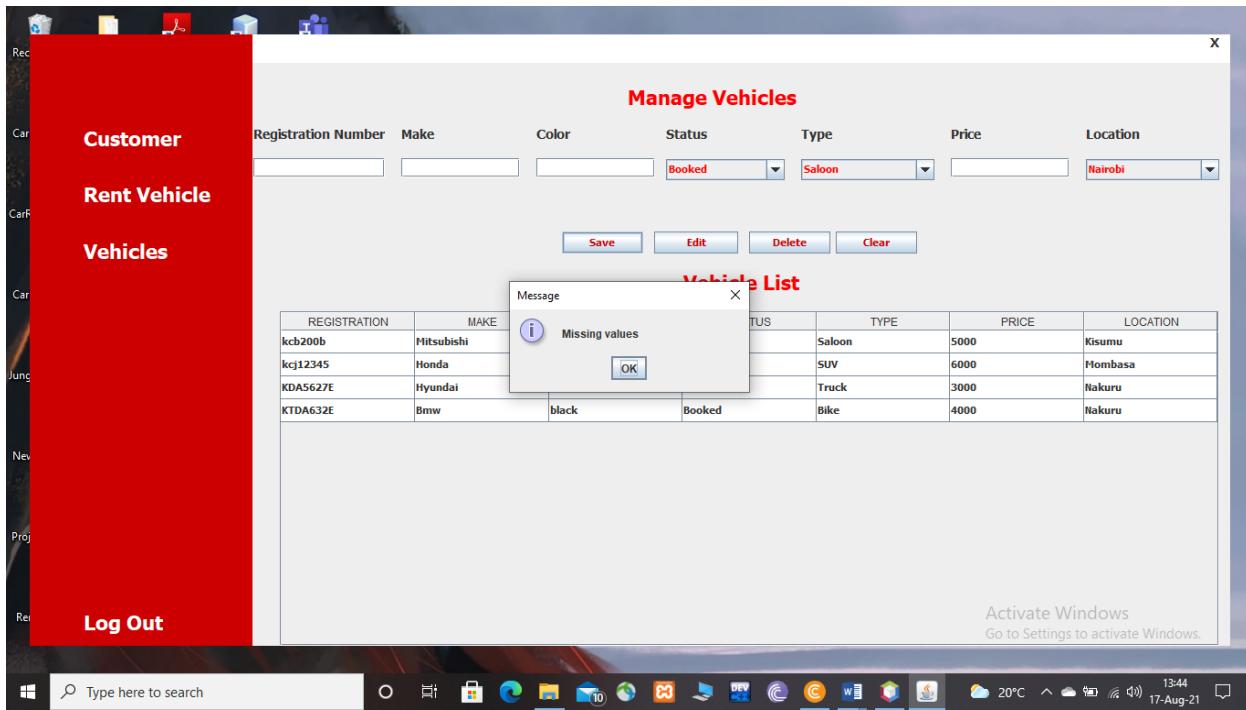
Managing vehicles

The vehicles are displayed

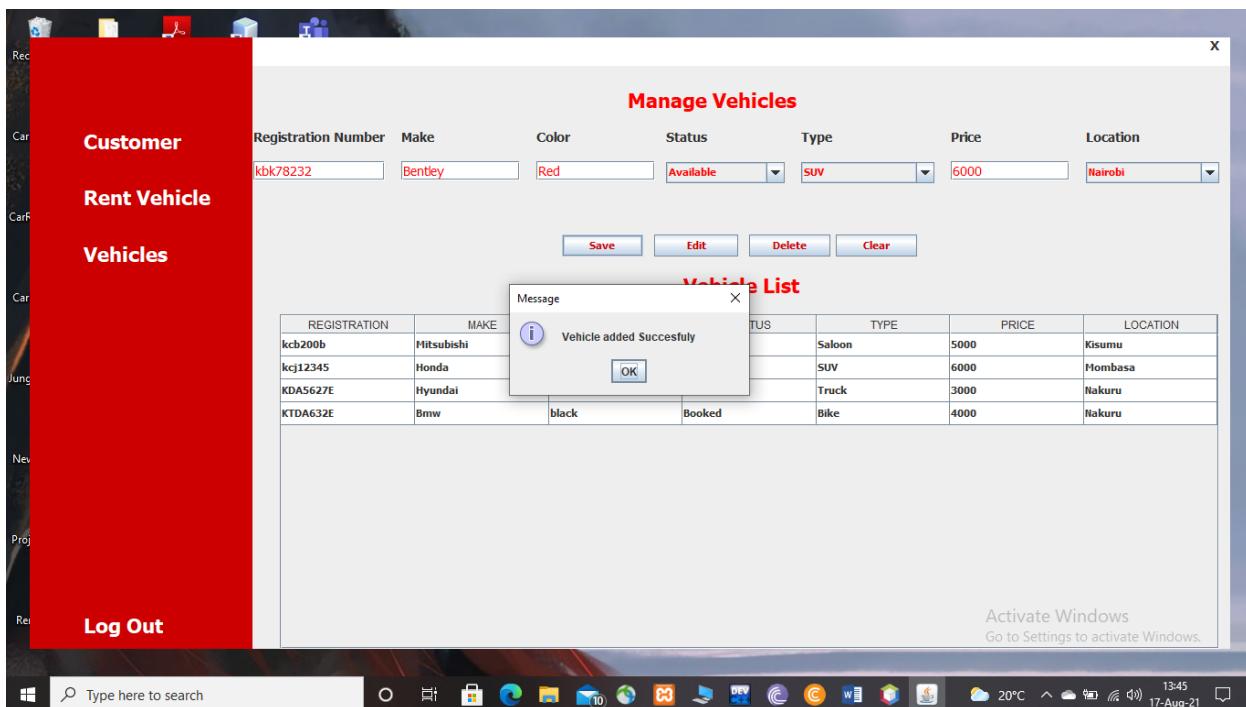
A screenshot of the RentaRide2 application running in Apache NetBeans IDE 12.4. On the left, there is a sidebar with red buttons labeled 'Customer', 'Rent Vehicle', 'Vehicles', and 'Log Out'. The main area is titled 'Manage Vehicles' and contains input fields for Registration Number, Make, Color, Status (set to 'Booked'), Type (set to 'Saloon'), Price, and Location (set to 'Nairobi'). Below these fields are buttons for 'Save', 'Edit', 'Delete', and 'Clear'. A table titled 'Vehicle List' displays a list of vehicles with columns: REGISTRATION, MAKE, COLOR, STATUS, TYPE, PRICE, and LOCATION. The table data is as follows:

Adding vehicle

When user adds vehicle with missing values error message is shown



When all details are entered as required and save button is pressed



Vehicle is then added to the vehicle list table

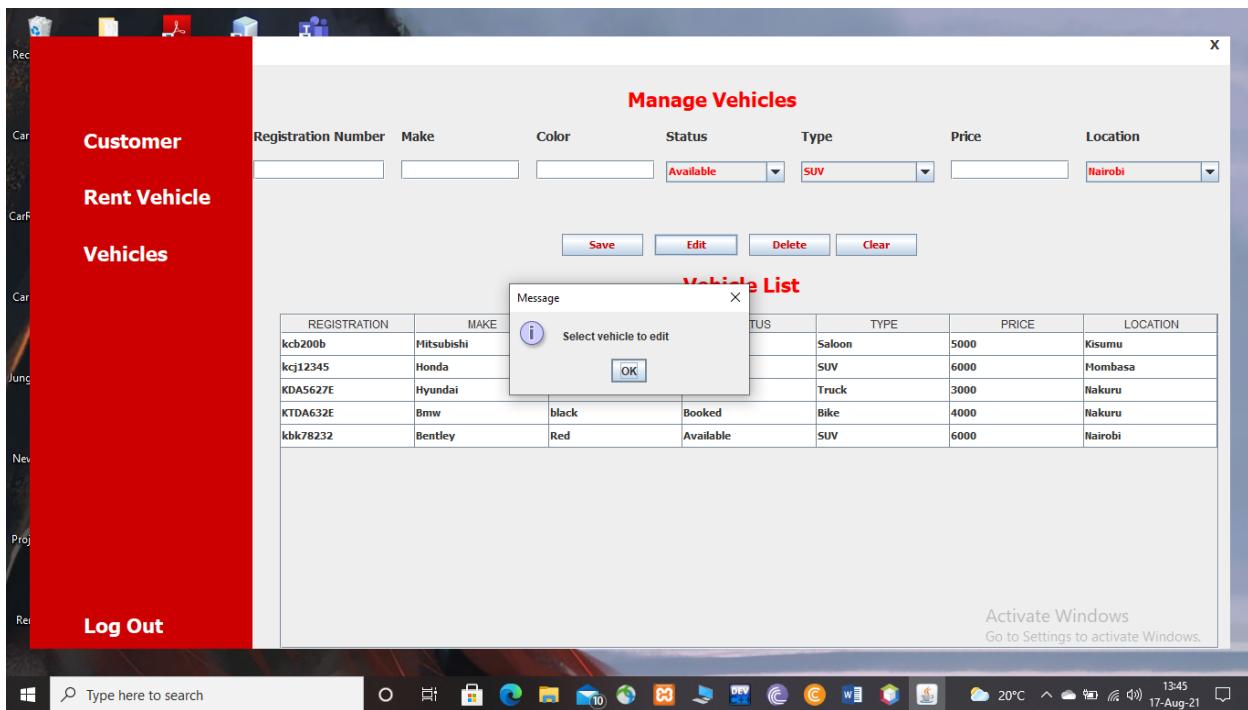
Manage Vehicles

REGISTRATION	MAKE	COLOR	STATUS	TYPE	PRICE	LOCATION
kcb200b	Mitsubishi	grey	Available	Saloon	5000	Kisumu
kgj12345	Honda	Green	Available	SUV	6000	Mombasa
KDA5627E	Hyundai	Black	Booked	Truck	3000	Nakuru
KTDA632E	Bmw	black	Booked	Bike	4000	Nakuru
kbk78232	Bentley	Red	Available	SUV	6000	Nairobi

Activate Windows
Go to Settings to activate Windows.

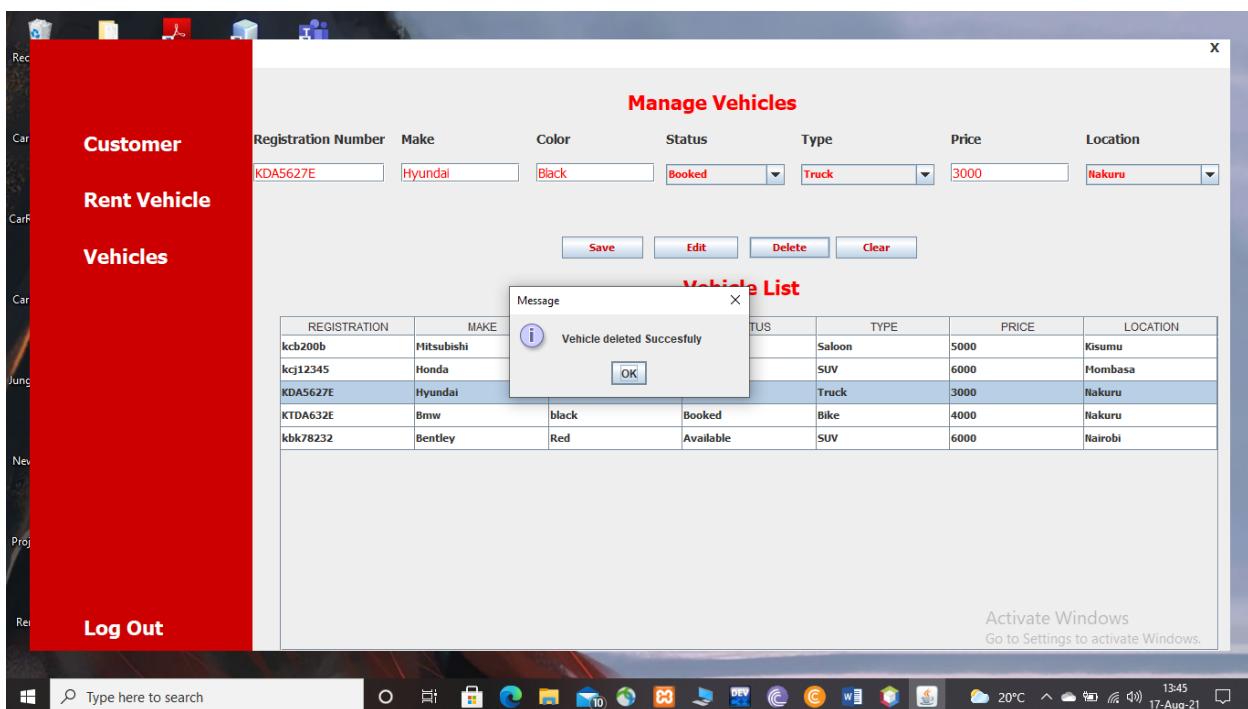
Editing vehicles

Press edit button without selecting row on table

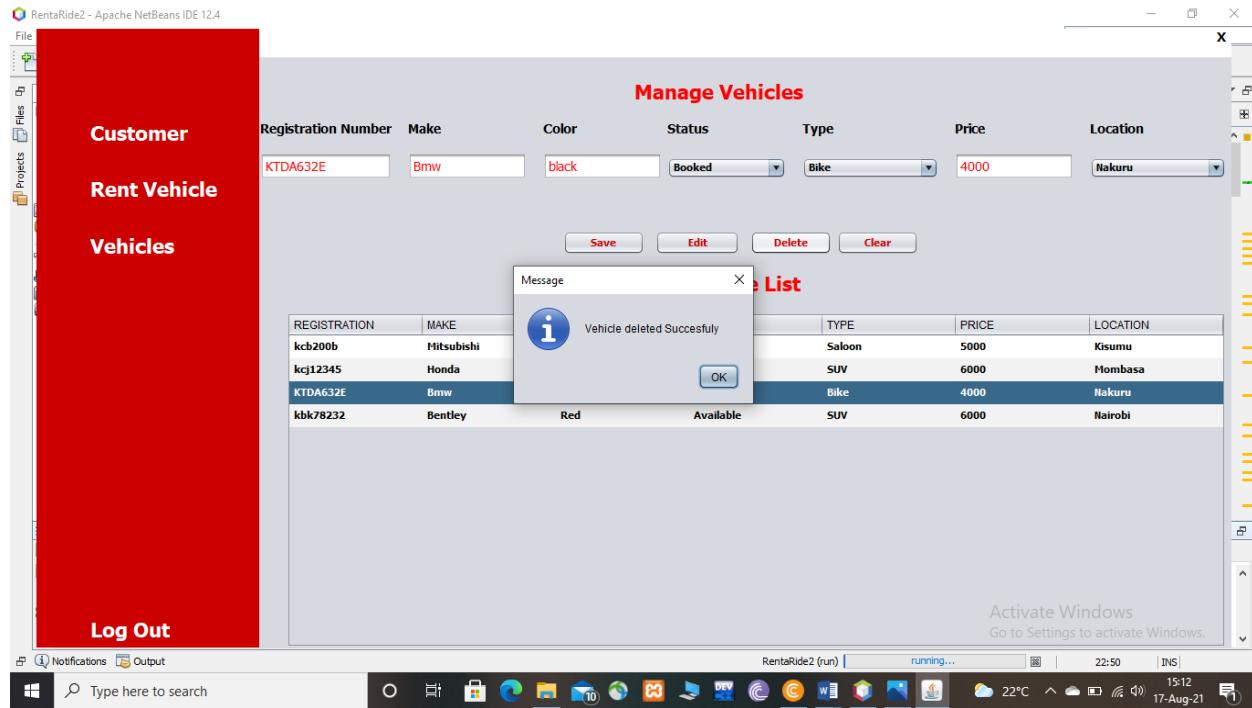


Deleting vehicle

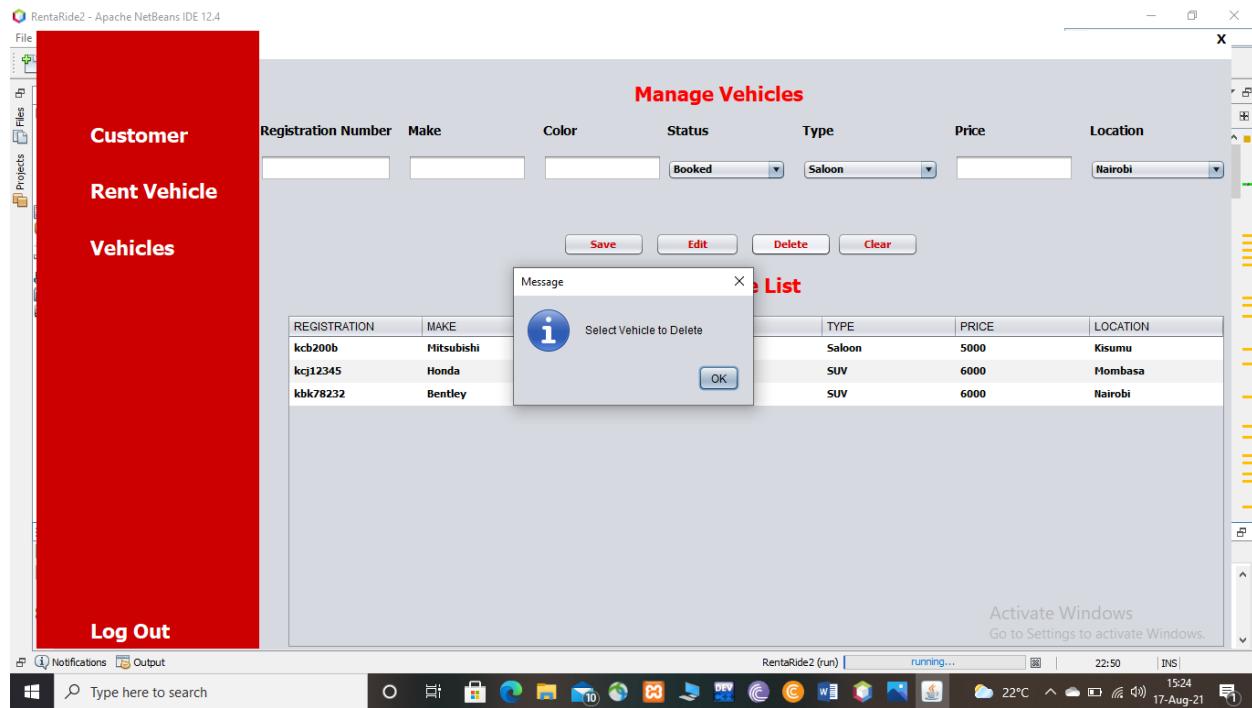
Deleting a record in the vehicles



Deleting vehicle

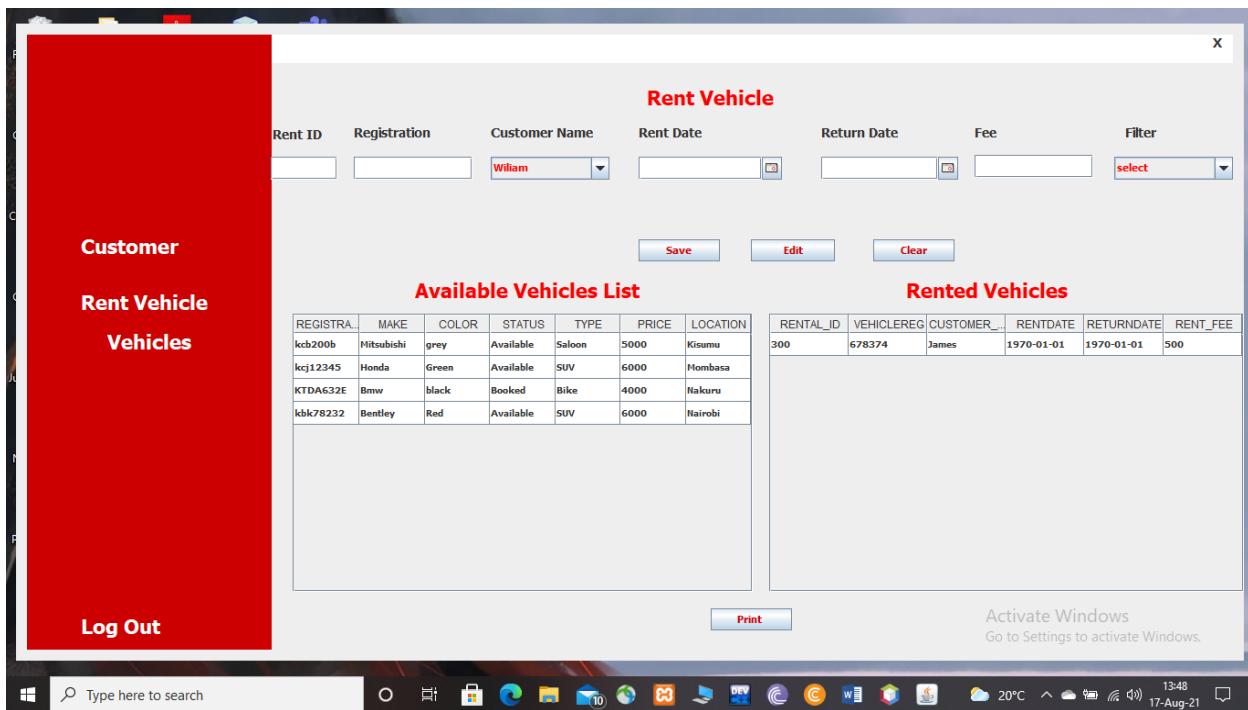


Trying to delete without selecting

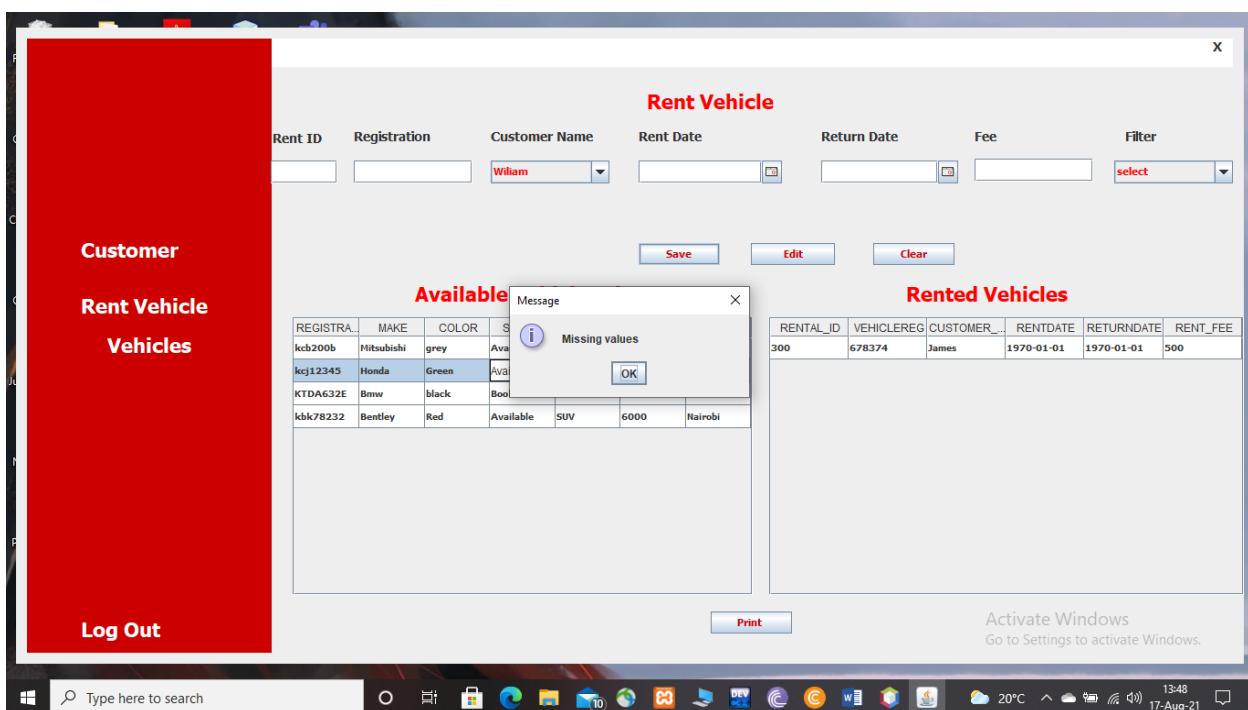


Renting vehicles

Displayed available vehicles

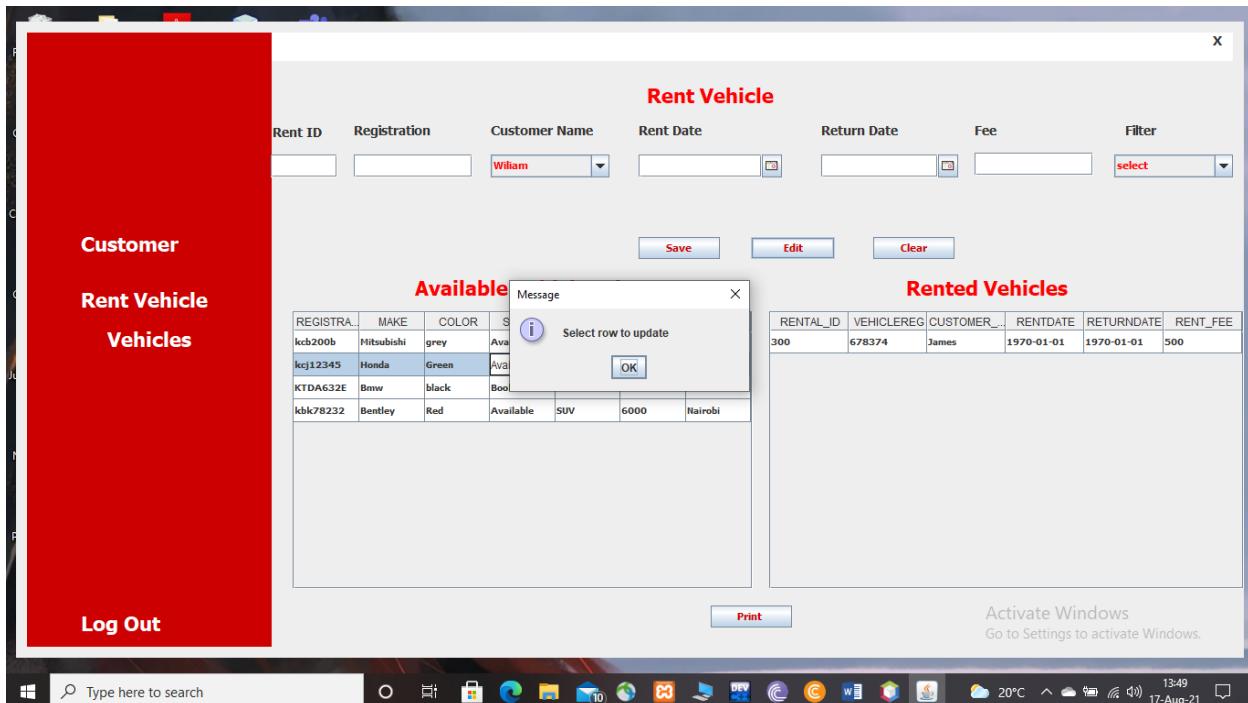


Missing values

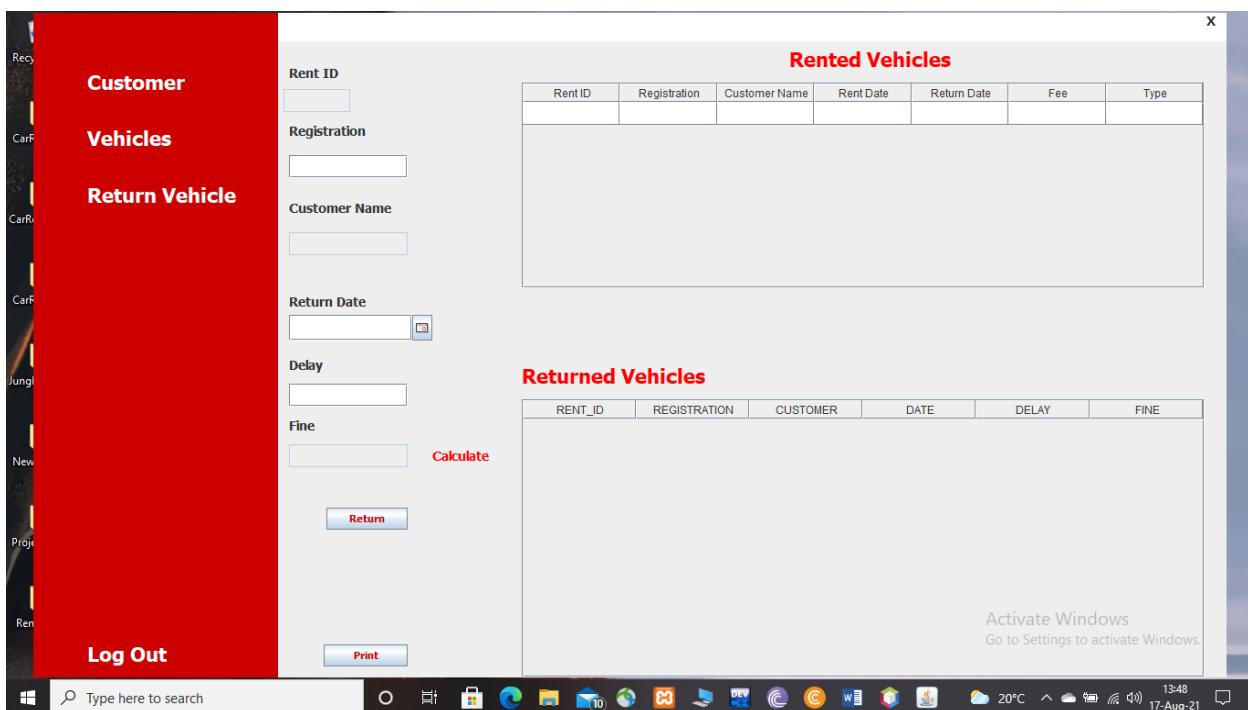


Updating rented car

Missing values

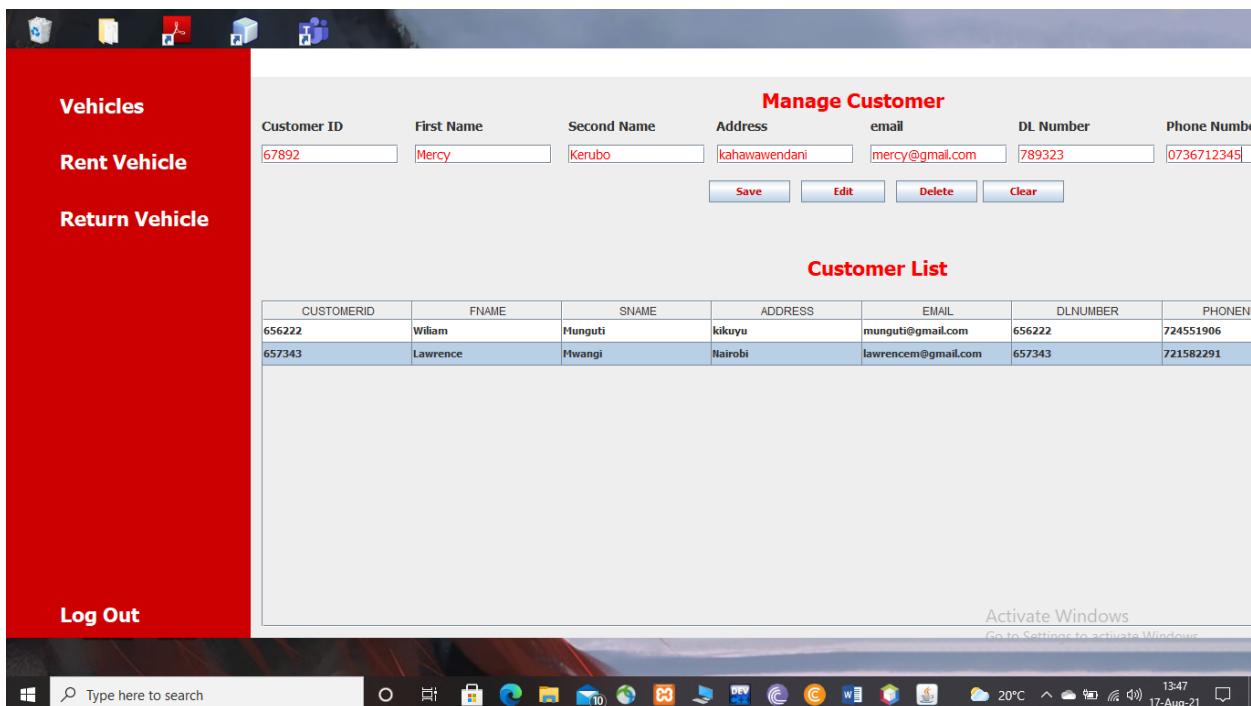


Returning car

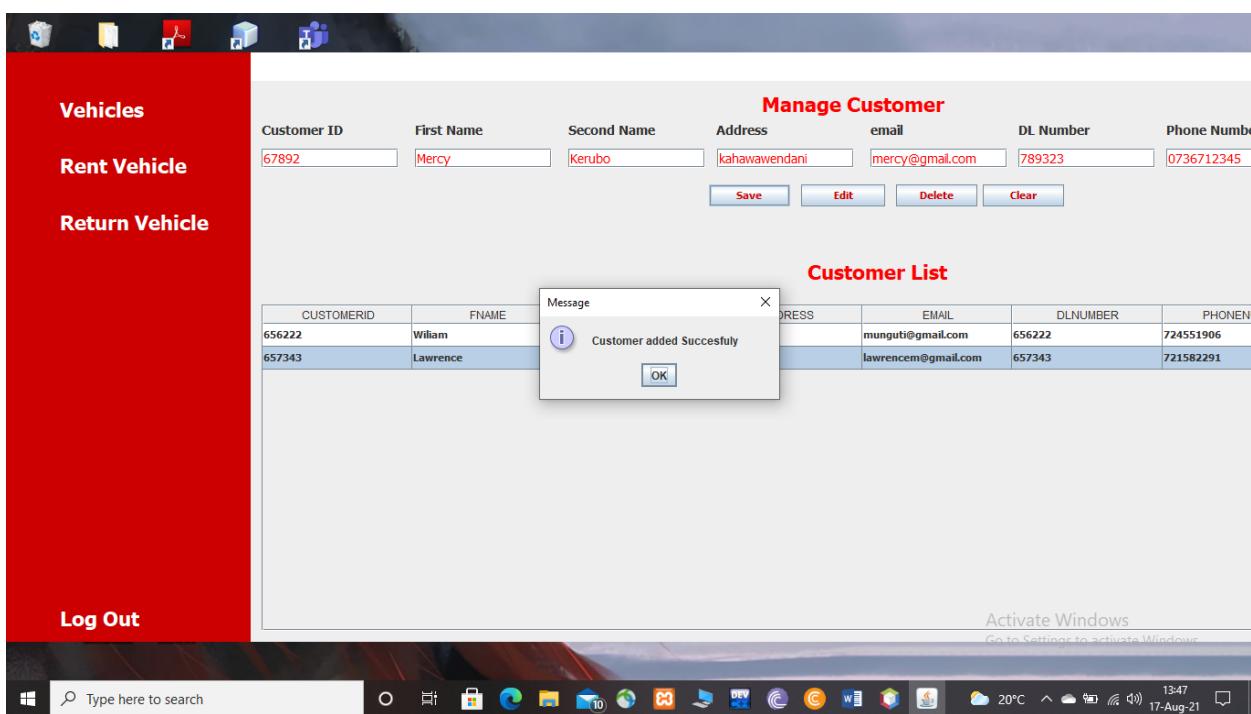


Managing customer

Display customers in system



Adding customer



Added customer displayed

Vehicles

Rent Vehicle

Return Vehicle

Log Out

Manage Customer

Customer ID	First Name	Second Name	Address	Email	DL Number	Phone Number
67892	Mercy	Kerubo	kahawwendani	mercy@gmail.com	789323	0736712345

Customer List

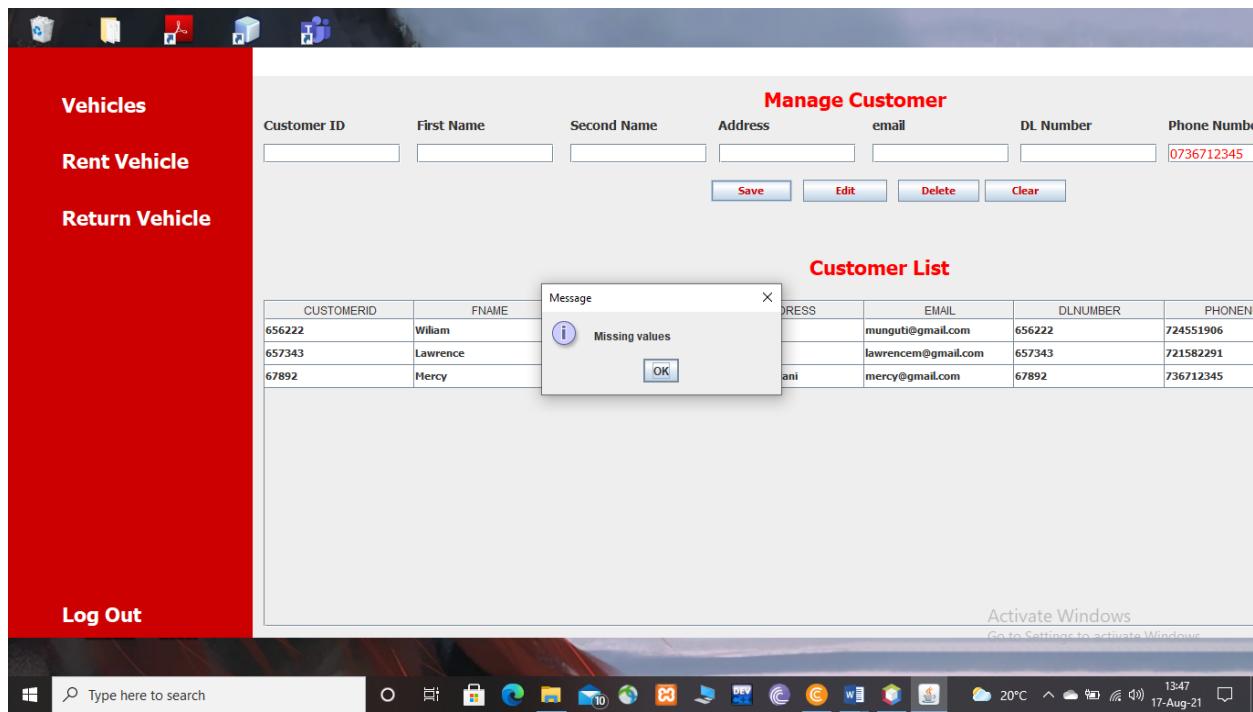
CUSTOMERID	FNAME	SNAME	ADDRESS	EMAIL	DLNUMBER	PHONEN
656222	Wilam	Munguti	kikuyu	munguti@gmail.com	656222	724551906
657343	Lawrence	Mwangi	Nairobi	lawrencem@gmail.com	657343	721582291
67892	Mercy	Kerubo	kahawwendani	mercy@gmail.com	67892	0736712345

Activate Windows
Go to Settings to activate Windows

Type here to search

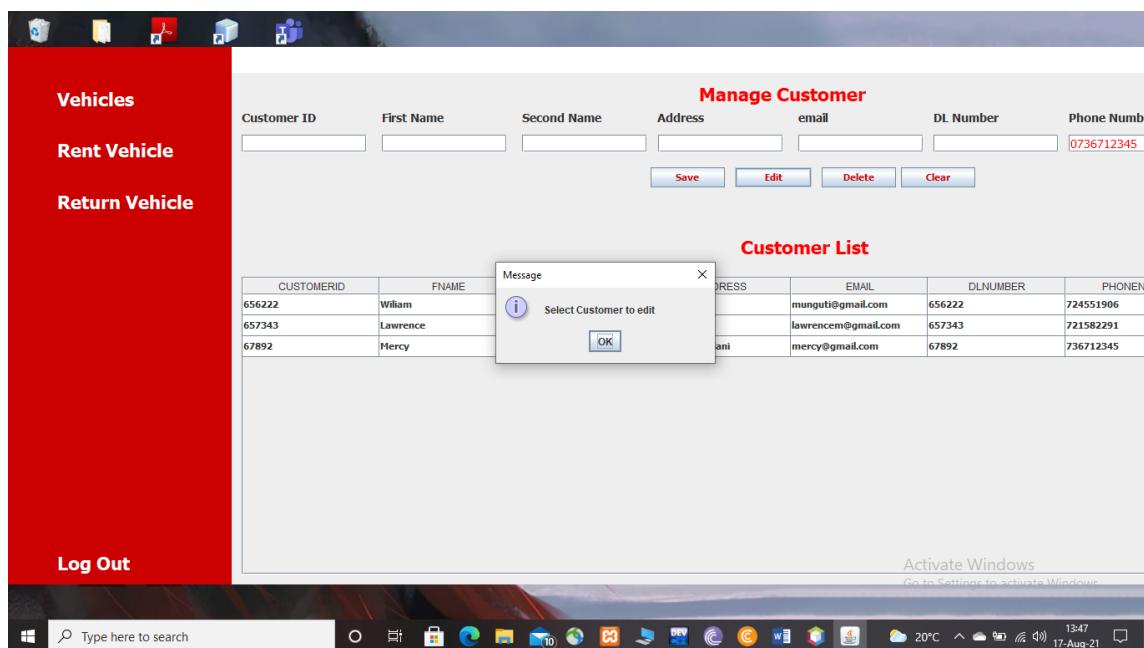
20°C 13:47 17-Aug-21

If required fields are not empty

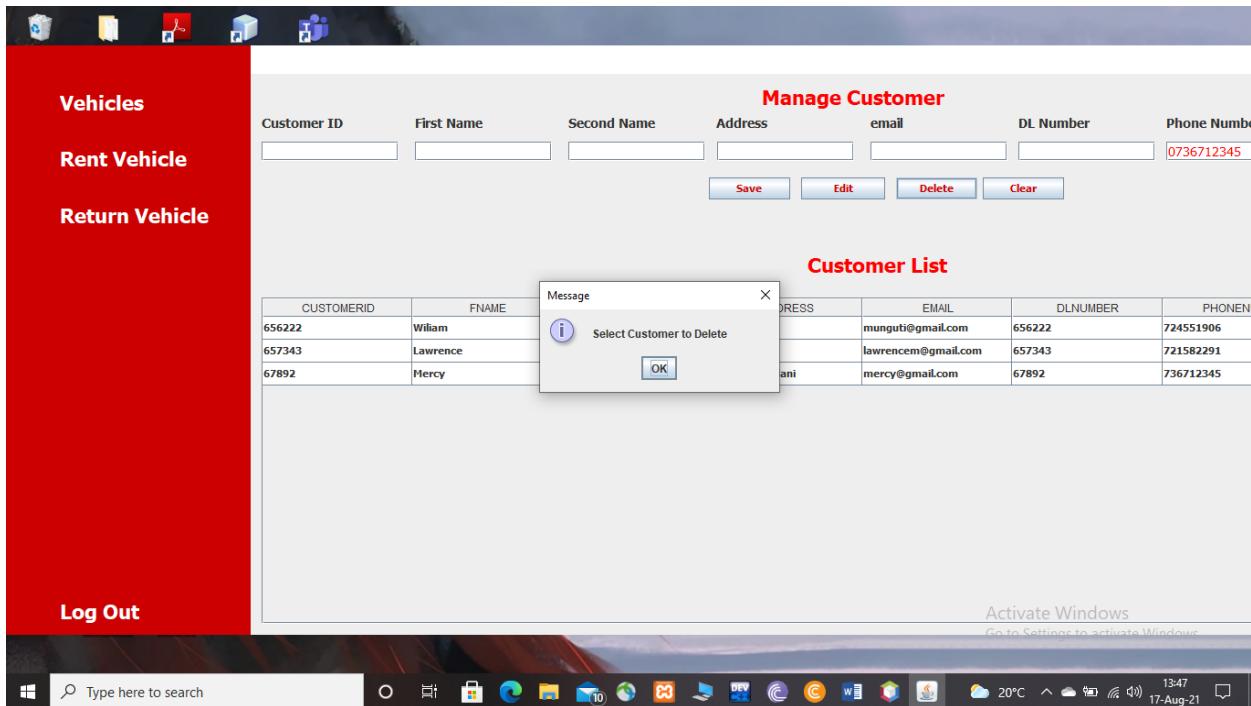


Editing customer

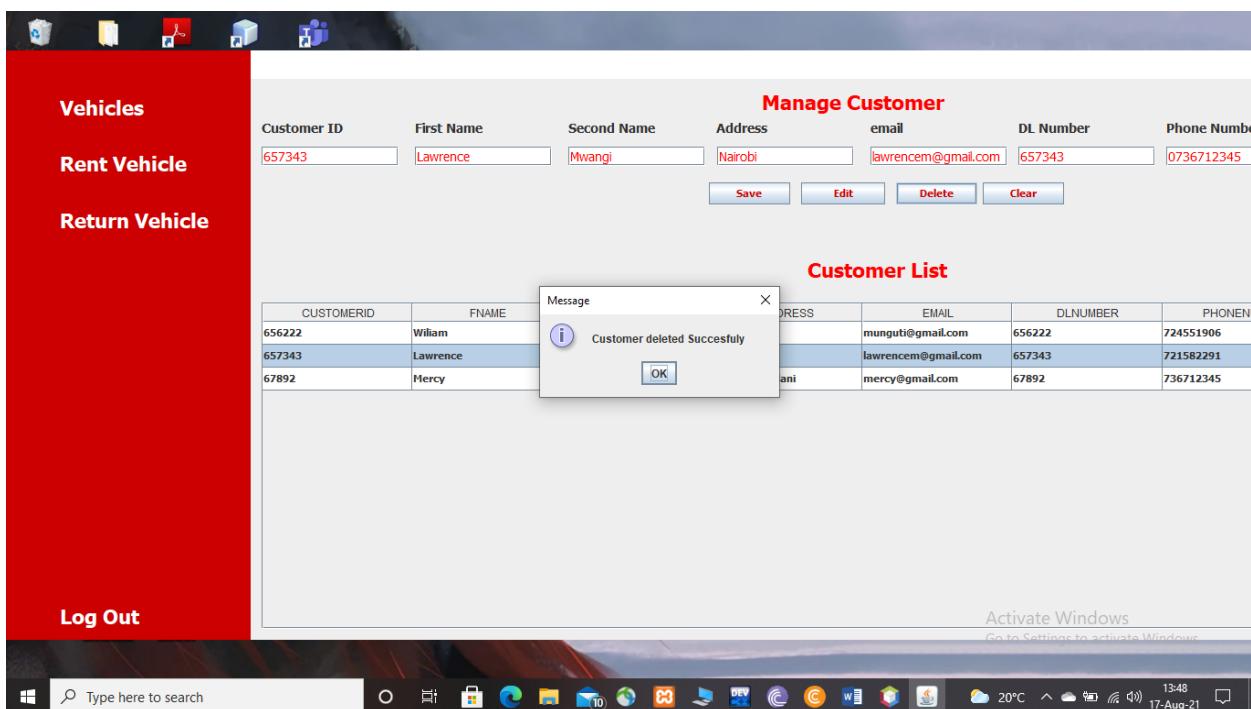
Trying to edit without selecting row on table



Deleting customer



Message shows after deleting



Deleted customer removed from list

The screenshot shows a Windows desktop environment with a red-themed application window titled "Manage Customer". The application has a sidebar on the left with options: Vehicles, Rent Vehicle, Return Vehicle, and Log Out. The main area contains a form for managing customers and a table for displaying the customer list.

Manage Customer Form:

Customer ID	First Name	Second Name	Address	email	DL Number	Phone Number
657343	Lawrence	Mwangi	Nairobi	lawrencem@gmail.com	657343	0736712345

Customer List Table:

CUSTOMERID	FNAME	SNAME	ADDRESS	EMAIL	DLNUMBER	PHONENO
656222	William	Munguti	kikuyu	munguti@gmail.com	656222	724551906
67892	Mercy	Kerubo	kahawawendani	mercy@gmail.com	67892	736712345

Buttons: Save, Edit, Delete, Clear.

Log Out button is visible at the bottom left of the sidebar.

Windows Taskbar: Shows various pinned icons like File Explorer, Edge, and File History. The system tray displays the date (17-Aug-21), time (13:48), battery level, and network status.

Customer after login

The screenshot shows a Windows desktop environment with a red-themed application window titled "Available Vehicles". The application has a sidebar on the left with options: File, Projects, Files, and Log Out. The main area contains a table for displaying available vehicles and a search/filter section below it. Below that is a section for booked vehicles with a booking button.

Available Vehicles Table:

REGISTRATION	MAKE	COLOR	STATUS	TYPE	PRICE	LOCATION
kcb200b	Mitsubishi	grey	Available	Saloon	5000	Kisumu
kci12345	Honda	Green	Available	SUV	6000	Mombasa
kkk78232	Bentley	Red	Available	SUV	6000	Nairobi

Filter Section:

Registration Number	Make	Color	Status	Type	Location
<input type="text"/>					

Booked Vehicle Table:

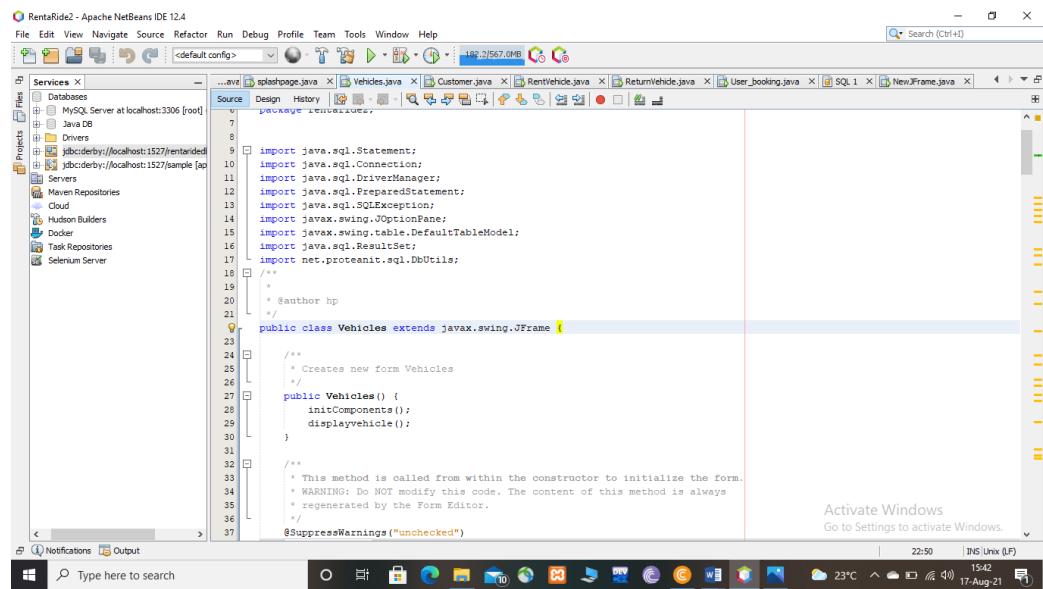
Registration Number	Make	Color	Status	Type	Location

Buttons: Book.

Log Out button is visible at the bottom left of the sidebar.

Windows Taskbar: Shows various pinned icons like File Explorer, Edge, and File History. The system tray displays the date (17-Aug-21), time (15:41), battery level, and network status.

Back end examples



RentARide2 - Apache NetBeans IDE 12.4

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Services X Databases MySQL Server at localhost:3306 [root] Projects Files Java DB Drivers Maven Repositories Hudson Builders Docker Task Repositories Selenium Server

Source Design History ... splashpage.java Vehicles.java Customer.java RentVehicle.java ReturnVehicle.java User_booking.java SQL 1 NewJFrame.java

```
...avt vehicles.java
1 package com.rentaride;
2
3 import java.sql.Statement;
4 import java.sql.Connection;
5 import java.sql.DriverManager;
6 import java.sql.PreparedStatement;
7 import java.sql.SQLException;
8 import javax.swing.JOptionPane;
9 import javax.swing.table.DefaultTableModel;
10 import java.sql.ResultSet;
11 import net.proteanit.sql.DbUtils;
12
13 /**
14  * 
15  * @author hp
16  */
17 public class Vehicles extends javax.swing.JFrame {
18
19     /**
20      * Creates new form Vehicles
21      */
22     public Vehicles() {
23         initComponents();
24         displayvehicle();
25     }
26
27     /**
28      * This method is called from within the constructor to initialize the form.
29      * WARNING: Do NOT modify this code. The content of this method is always
30      * regenerated by the Form Editor.
31      */
32     @SuppressWarnings("unchecked")
33     // Variables declaration - do not modify//GEN-BEGIN:variables
34     // End of variables declaration//GEN-END:variables
35
36     /**
37      * @author hp
38      */
39     private void displayvehicle() {
40         try {
41             Class.forName("org.apache.derby.jdbc.ClientDriver");
42             Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/rentaridedb", "root", "root");
43             Statement add = con.createStatement();
44             String Query = "Select * from vehicles";
45             ResultSet rs = add.executeQuery(Query);
46             DefaultTableModel model = (DefaultTableModel) jTable1.getModel();
47             while (rs.next()) {
48                 model.addRow(new Object[]{rs.getString("reg_num"), rs.getString("make"), rs.getString("color"), rs.getString("status")});
49             }
50         } catch (Exception e) {
51             e.printStackTrace();
52         }
53     }
54
55     /**
56      * @author hp
57      */
58     private void btn_addActionPerformed(java.awt.event.ActionEvent evt) {
59         try {
60             Class.forName("org.apache.derby.jdbc.ClientDriver");
61             Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/rentaridedb", "root", "root");
62             Statement add = con.createStatement();
63             String Query = "Insert into vehicles values ('" + jTextField1.getText() + "','" + jTextField2.getText() + "','" + jTextField3.getText() + "','" + jTextField4.getText() + "')";
64             add.executeUpdate(Query);
65             JOptionPane.showMessageDialog(this, "Added successfully");
66             displayvehicle();
67         } catch (Exception e) {
68             e.printStackTrace();
69         }
70     }
71
72     /**
73      * @author hp
74      */
75     private void btn_edit_vehActionPerformed(java.awt.event.ActionEvent evt) {
76         if (reg_num.getText().isEmpty() || veh_make.getText().isEmpty() || veh_color.getText().isEmpty() || veh_status.getSelectedIndex() == 0) {
77             JOptionPane.showMessageDialog(this, "Select vehicle to edit");
78         } else {
79             try {
80                 Class.forName("org.apache.derby.jdbc.ClientDriver");
81                 Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/rentaridedb", "root", "root");
82                 Statement add = con.createStatement();
83                 String Query = "Update vehicles set registration= '" + reg_num.getText() + "', make = '" + veh_make.getText() + "', color = '" + veh_color.getText() + "', status = '" + veh_status.getSelectedItem() + "' where id = " + jTable1.getValueAt(jTable1.getSelectedRow(), 0).toString();
84                 add.executeUpdate(Query);
85                 JOptionPane.showMessageDialog(this, "Updated successfully");
86                 displayvehicle();
87             } catch (Exception e) {
88                 e.printStackTrace();
89             }
90         }
91     }
92
93     /**
94      * @author hp
95      */
96     private void btn_del_vehActionPerformed(java.awt.event.ActionEvent evt) {
97         if (jTable1.getSelectedRow() == -1) {
98             JOptionPane.showMessageDialog(this, "Select vehicle to delete");
99         } else {
100            try {
101                Class.forName("org.apache.derby.jdbc.ClientDriver");
102                Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/rentaridedb", "root", "root");
103                Statement add = con.createStatement();
104                String Query = "Delete from vehicles where id = " + jTable1.getValueAt(jTable1.getSelectedRow(), 0).toString();
105                add.executeUpdate(Query);
106                JOptionPane.showMessageDialog(this, "Deleted successfully");
107                displayvehicle();
108            } catch (Exception e) {
109                e.printStackTrace();
110            }
111        }
112    }
113
114    /**
115      * @author hp
116      */
117    private void jTable1MouseClicked(java.awt.event.MouseEvent evt) {
118        if (jTable1.getSelectedRow() != -1) {
119            reg_num.setText(jTable1.getValueAt(jTable1.getSelectedRow(), 0).toString());
120            veh_make.setText(jTable1.getValueAt(jTable1.getSelectedRow(), 1).toString());
121            veh_color.setText(jTable1.getValueAt(jTable1.getSelectedRow(), 2).toString());
122            veh_status.setSelectedIndex(Integer.parseInt(jTable1.getValueAt(jTable1.getSelectedRow(), 3).toString()));
123        }
124    }
125
126    /**
127      * @author hp
128      */
129    private void jTextField1KeyTyped(java.awt.event.KeyEvent evt) {
130        if (jTextField1.getText().length() > 10) {
131            jTextField1.setEditable(false);
132        }
133    }
134
135    /**
136      * @author hp
137      */
138    private void jTextField2KeyTyped(java.awt.event.KeyEvent evt) {
139        if (jTextField2.getText().length() > 10) {
140            jTextField2.setEditable(false);
141        }
142    }
143
144    /**
145      * @author hp
146      */
147    private void jTextField3KeyTyped(java.awt.event.KeyEvent evt) {
148        if (jTextField3.getText().length() > 10) {
149            jTextField3.setEditable(false);
150        }
151    }
152
153    /**
154      * @author hp
155      */
156    private void jTextField4KeyTyped(java.awt.event.KeyEvent evt) {
157        if (jTextField4.getText().length() > 10) {
158            jTextField4.setEditable(false);
159        }
160    }
161
162    /**
163      * @author hp
164      */
165    private void jTextField5KeyTyped(java.awt.event.KeyEvent evt) {
166        if (jTextField5.getText().length() > 10) {
167            jTextField5.setEditable(false);
168        }
169    }
170
171    /**
172      * @author hp
173      */
174    private void jTextField6KeyTyped(java.awt.event.KeyEvent evt) {
175        if (jTextField6.getText().length() > 10) {
176            jTextField6.setEditable(false);
177        }
178    }
179
180    /**
181      * @author hp
182      */
183    private void jTextField7KeyTyped(java.awt.event.KeyEvent evt) {
184        if (jTextField7.getText().length() > 10) {
185            jTextField7.setEditable(false);
186        }
187    }
188
189    /**
190      * @author hp
191      */
192    private void jTextField8KeyTyped(java.awt.event.KeyEvent evt) {
193        if (jTextField8.getText().length() > 10) {
194            jTextField8.setEditable(false);
195        }
196    }
197
198    /**
199      * @author hp
200      */
201    private void jTextField9KeyTyped(java.awt.event.KeyEvent evt) {
202        if (jTextField9.getText().length() > 10) {
203            jTextField9.setEditable(false);
204        }
205    }
206
207    /**
208      * @author hp
209      */
210    private void jTextField10KeyTyped(java.awt.event.KeyEvent evt) {
211        if (jTextField10.getText().length() > 10) {
212            jTextField10.setEditable(false);
213        }
214    }
215
216    /**
217      * @author hp
218      */
219    private void jTextField11KeyTyped(java.awt.event.KeyEvent evt) {
220        if (jTextField11.getText().length() > 10) {
221            jTextField11.setEditable(false);
222        }
223    }
224
225    /**
226      * @author hp
227      */
228    private void jTextField12KeyTyped(java.awt.event.KeyEvent evt) {
229        if (jTextField12.getText().length() > 10) {
230            jTextField12.setEditable(false);
231        }
232    }
233
234    /**
235      * @author hp
236      */
237    private void jTextField13KeyTyped(java.awt.event.KeyEvent evt) {
238        if (jTextField13.getText().length() > 10) {
239            jTextField13.setEditable(false);
240        }
241    }
242
243    /**
244      * @author hp
245      */
246    private void jTextField14KeyTyped(java.awt.event.KeyEvent evt) {
247        if (jTextField14.getText().length() > 10) {
248            jTextField14.setEditable(false);
249        }
250    }
251
252    /**
253      * @author hp
254      */
255    private void jTextField15KeyTyped(java.awt.event.KeyEvent evt) {
256        if (jTextField15.getText().length() > 10) {
257            jTextField15.setEditable(false);
258        }
259    }
260
261    /**
262      * @author hp
263      */
264    private void jTextField16KeyTyped(java.awt.event.KeyEvent evt) {
265        if (jTextField16.getText().length() > 10) {
266            jTextField16.setEditable(false);
267        }
268    }
269
270    /**
271      * @author hp
272      */
273    private void jTextField17KeyTyped(java.awt.event.KeyEvent evt) {
274        if (jTextField17.getText().length() > 10) {
275            jTextField17.setEditable(false);
276        }
277    }
278
279    /**
280      * @author hp
281      */
282    private void jTextField18KeyTyped(java.awt.event.KeyEvent evt) {
283        if (jTextField18.getText().length() > 10) {
284            jTextField18.setEditable(false);
285        }
286    }
287
288    /**
289      * @author hp
290      */
291    private void jTextField19KeyTyped(java.awt.event.KeyEvent evt) {
292        if (jTextField19.getText().length() > 10) {
293            jTextField19.setEditable(false);
294        }
295    }
296
297    /**
298      * @author hp
299      */
300    private void jTextField20KeyTyped(java.awt.event.KeyEvent evt) {
301        if (jTextField20.getText().length() > 10) {
302            jTextField20.setEditable(false);
303        }
304    }
305
306    /**
307      * @author hp
308      */
309    private void jTextField21KeyTyped(java.awt.event.KeyEvent evt) {
310        if (jTextField21.getText().length() > 10) {
311            jTextField21.setEditable(false);
312        }
313    }
314
315    /**
316      * @author hp
317      */
318    private void jTextField22KeyTyped(java.awt.event.KeyEvent evt) {
319        if (jTextField22.getText().length() > 10) {
320            jTextField22.setEditable(false);
321        }
322    }
323
324    /**
325      * @author hp
326      */
327    private void jTextField23KeyTyped(java.awt.event.KeyEvent evt) {
328        if (jTextField23.getText().length() > 10) {
329            jTextField23.setEditable(false);
330        }
331    }
332
333    /**
334      * @author hp
335      */
336    private void jTextField24KeyTyped(java.awt.event.KeyEvent evt) {
337        if (jTextField24.getText().length() > 10) {
338            jTextField24.setEditable(false);
339        }
340    }
341
342    /**
343      * @author hp
344      */
345    private void jTextField25KeyTyped(java.awt.event.KeyEvent evt) {
346        if (jTextField25.getText().length() > 10) {
347            jTextField25.setEditable(false);
348        }
349    }
350
351    /**
352      * @author hp
353      */
354    private void jTextField26KeyTyped(java.awt.event.KeyEvent evt) {
355        if (jTextField26.getText().length() > 10) {
356            jTextField26.setEditable(false);
357        }
358    }
359
360    /**
361      * @author hp
362      */
363    private void jTextField27KeyTyped(java.awt.event.KeyEvent evt) {
364        if (jTextField27.getText().length() > 10) {
365            jTextField27.setEditable(false);
366        }
367    }
368
369    /**
370      * @author hp
371      */
372    private void jTextField28KeyTyped(java.awt.event.KeyEvent evt) {
373        if (jTextField28.getText().length() > 10) {
374            jTextField28.setEditable(false);
375        }
376    }
377
378    /**
379      * @author hp
380      */
381    private void jTextField29KeyTyped(java.awt.event.KeyEvent evt) {
382        if (jTextField29.getText().length() > 10) {
383            jTextField29.setEditable(false);
384        }
385    }
386
387    /**
388      * @author hp
389      */
390    private void jTextField30KeyTyped(java.awt.event.KeyEvent evt) {
391        if (jTextField30.getText().length() > 10) {
392            jTextField30.setEditable(false);
393        }
394    }
395
396    /**
397      * @author hp
398      */
399    private void jTextField31KeyTyped(java.awt.event.KeyEvent evt) {
400        if (jTextField31.getText().length() > 10) {
401            jTextField31.setEditable(false);
402        }
403    }
404
405    /**
406      * @author hp
407      */
408    private void jTextField32KeyTyped(java.awt.event.KeyEvent evt) {
409        if (jTextField32.getText().length() > 10) {
410            jTextField32.setEditable(false);
411        }
412    }
413
414    /**
415      * @author hp
416      */
417    private void jTextField33KeyTyped(java.awt.event.KeyEvent evt) {
418        if (jTextField33.getText().length() > 10) {
419            jTextField33.setEditable(false);
420        }
421    }
422
423    /**
424      * @author hp
425      */
426    private void jTextField34KeyTyped(java.awt.event.KeyEvent evt) {
427        if (jTextField34.getText().length() > 10) {
428            jTextField34.setEditable(false);
429        }
430    }
431
432    /**
433      * @author hp
434      */
435    private void jTextField35KeyTyped(java.awt.event.KeyEvent evt) {
436        if (jTextField35.getText().length() > 10) {
437            jTextField35.setEditable(false);
438        }
439    }
440
441    /**
442      * @author hp
443      */
444    private void jTextField36KeyTyped(java.awt.event.KeyEvent evt) {
445        if (jTextField36.getText().length() > 10) {
446            jTextField36.setEditable(false);
447        }
448    }
449
450    /**
451      * @author hp
452      */
453    private void jTextField37KeyTyped(java.awt.event.KeyEvent evt) {
454        if (jTextField37.getText().length() > 10) {
455            jTextField37.setEditable(false);
456        }
457    }
458
459    /**
460      * @author hp
461      */
462    private void jTextField38KeyTyped(java.awt.event.KeyEvent evt) {
463        if (jTextField38.getText().length() > 10) {
464            jTextField38.setEditable(false);
465        }
466    }
467
468    /**
469      * @author hp
470      */
471    private void jTextField39KeyTyped(java.awt.event.KeyEvent evt) {
472        if (jTextField39.getText().length() > 10) {
473            jTextField39.setEditable(false);
474        }
475    }
476
477    /**
478      * @author hp
479      */
480    private void jTextField40KeyTyped(java.awt.event.KeyEvent evt) {
481        if (jTextField40.getText().length() > 10) {
482            jTextField40.setEditable(false);
483        }
484    }
485
486    /**
487      * @author hp
488      */
489    private void jTextField41KeyTyped(java.awt.event.KeyEvent evt) {
490        if (jTextField41.getText().length() > 10) {
491            jTextField41.setEditable(false);
492        }
493    }
494
495    /**
496      * @author hp
497      */
498    private void jTextField42KeyTyped(java.awt.event.KeyEvent evt) {
499        if (jTextField42.getText().length() > 10) {
500            jTextField42.setEditable(false);
501        }
502    }
503
504    /**
505      * @author hp
506      */
507    private void jTextField43KeyTyped(java.awt.event.KeyEvent evt) {
508        if (jTextField43.getText().length() > 10) {
509            jTextField43.setEditable(false);
510        }
511    }
512
513    /**
514      * @author hp
515      */
516    private void jTextField44KeyTyped(java.awt.event.KeyEvent evt) {
517        if (jTextField44.getText().length() > 10) {
518            jTextField44.setEditable(false);
519        }
520    }
521
522    /**
523      * @author hp
524      */
525    private void jTextField45KeyTyped(java.awt.event.KeyEvent evt) {
526        if (jTextField45.getText().length() > 10) {
527            jTextField45.setEditable(false);
528        }
529    }
530
531    /**
532      * @author hp
533      */
534    private void jTextField46KeyTyped(java.awt.event.KeyEvent evt) {
535        if (jTextField46.getText().length() > 10) {
536            jTextField46.setEditable(false);
537        }
538    }
539
540    /**
541      * @author hp
542      */
543    private void jTextField47KeyTyped(java.awt.event.KeyEvent evt) {
544        if (jTextField47.getText().length() > 10) {
545            jTextField47.setEditable(false);
546        }
547    }
548
549    /**
550      * @author hp
551      */
552    private void jTextField48KeyTyped(java.awt.event.KeyEvent evt) {
553        if (jTextField48.getText().length() > 10) {
554            jTextField48.setEditable(false);
555        }
556    }
557
558    /**
559      * @author hp
560      */
561    private void jTextField49KeyTyped(java.awt.event.KeyEvent evt) {
562        if (jTextField49.getText().length() > 10) {
563            jTextField49.setEditable(false);
564        }
565    }
566
567    /**
568      * @author hp
569      */
570    private void jTextField50KeyTyped(java.awt.event.KeyEvent evt) {
571        if (jTextField50.getText().length() > 10) {
572            jTextField50.setEditable(false);
573        }
574    }
575
576    /**
577      * @author hp
578      */
579    private void jTextField51KeyTyped(java.awt.event.KeyEvent evt) {
580        if (jTextField51.getText().length() > 10) {
581            jTextField51.setEditable(false);
582        }
583    }
584
585    /**
586      * @author hp
587      */
588    private void jTextField52KeyTyped(java.awt.event.KeyEvent evt) {
589        if (jTextField52.getText().length() > 10) {
590            jTextField52.setEditable(false);
591        }
592    }
593
594    /**
595      * @author hp
596      */
597    private void jTextField53KeyTyped(java.awt.event.KeyEvent evt) {
598        if (jTextField53.getText().length() > 10) {
599            jTextField53.setEditable(false);
600        }
601    }
602
603    /**
604      * @author hp
605      */
606    private void jTextField54KeyTyped(java.awt.event.KeyEvent evt) {
607        if (jTextField54.getText().length() > 10) {
608            jTextField54.setEditable(false);
609        }
610    }
611
612    /**
613      * @author hp
614      */
615    private void jTextField55KeyTyped(java.awt.event.KeyEvent evt) {
616        if (jTextField55.getText().length() > 10) {
617            jTextField55.setEditable(false);
618        }
619    }
620
621    /**
622      * @author hp
623      */
624    private void jTextField56KeyTyped(java.awt.event.KeyEvent evt) {
625        if (jTextField56.getText().length() > 10) {
626            jTextField56.setEditable(false);
627        }
628    }
629
630    /**
631      * @author hp
632      */
633    private void jTextField57KeyTyped(java.awt.event.KeyEvent evt) {
634        if (jTextField57.getText().length() > 10) {
635            jTextField57.setEditable(false);
636        }
637    }
638
639    /**
640      * @author hp
641      */
642    private void jTextField58KeyTyped(java.awt.event.KeyEvent evt) {
643        if (jTextField58.getText().length() > 10) {
644            jTextField58.setEditable(false);
645        }
646    }
647
648    /**
649      * @author hp
650      */
651    private void jTextField59KeyTyped(java.awt.event.KeyEvent evt) {
652        if (jTextField59.getText().length() > 10) {
653            jTextField59.setEditable(false);
654        }
655    }
656
657    /**
658      * @author hp
659      */
660    private void jTextField60KeyTyped(java.awt.event.KeyEvent evt) {
661        if (jTextField60.getText().length() > 10) {
662            jTextField60.setEditable(false);
663        }
664    }
665
666    /**
667      * @author hp
668      */
669    private void jTextField61KeyTyped(java.awt.event.KeyEvent evt) {
670        if (jTextField61.getText().length() > 10) {
671            jTextField61.setEditable(false);
672        }
673    }
674
675    /**
676      * @author hp
677      */
678    private void jTextField62KeyTyped(java.awt.event.KeyEvent evt) {
679        if (jTextField62.getText().length() > 10) {
680            jTextField62.setEditable(false);
681        }
682    }
683
684    /**
685      * @author hp
686      */
687    private void jTextField63KeyTyped(java.awt.event.KeyEvent evt) {
688        if (jTextField63.getText().length() > 10) {
689            jTextField63.setEditable(false);
690        }
691    }
692
693    /**
694      * @author hp
695      */
696    private void jTextField64KeyTyped(java.awt.event.KeyEvent evt) {
697        if (jTextField64.getText().length() > 10) {
698            jTextField64.setEditable(false);
699        }
700    }
701
702    /**
703      * @author hp
704      */
705    private void jTextField65KeyTyped(java.awt.event.KeyEvent evt) {
706        if (jTextField65.getText().length() > 10) {
707            jTextField65.setEditable(false);
708        }
709    }
710
711    /**
712      * @author hp
713      */
714    private void jTextField66KeyTyped(java.awt.event.KeyEvent evt) {
715        if (jTextField66.getText().length() > 10) {
716            jTextField66.setEditable(false);
717        }
718    }
719
720    /**
721      * @author hp
722      */
723    private void jTextField67KeyTyped(java.awt.event.KeyEvent evt) {
724        if (jTextField67.getText().length() > 10) {
725            jTextField67.setEditable(false);
726        }
727    }
728
729    /**
730      * @author hp
731      */
732    private void jTextField68KeyTyped(java.awt.event.KeyEvent evt) {
733        if (jTextField68.getText().length() > 10) {
734            jTextField68.setEditable(false);
735        }
736    }
737
738    /**
739      * @author hp
740      */
741    private void jTextField69KeyTyped(java.awt.event.KeyEvent evt) {
742        if (jTextField69.getText().length() > 10) {
743            jTextField69.setEditable(false);
744        }
745    }
746
747    /**
748      * @author hp
749      */
750    private void jTextField70KeyTyped(java.awt.event.KeyEvent evt) {
751        if (jTextField70.getText().length() > 10) {
752            jTextField70.setEditable(false);
753        }
754    }
755
756    /**
757      * @author hp
758      */
759    private void jTextField71KeyTyped(java.awt.event.KeyEvent evt) {
760        if (jTextField71.getText().length() > 10) {
761            jTextField71.setEditable(false);
762        }
763    }
764
765    /**
766      * @author hp
767      */
768    private void jTextField72KeyTyped(java.awt.event.KeyEvent evt) {
769        if (jTextField72.getText().length() > 10) {
770            jTextField72.setEditable(false);
771        }
772    }
773
774    /**
775      * @author hp
776      */
777    private void jTextField73KeyTyped(java.awt.event.KeyEvent evt) {
778        if (jTextField73.getText().length() > 10) {
779            jTextField73.setEditable(false);
780        }
781    }
782
783    /**
784      * @author hp
785      */
786    private void jTextField74KeyTyped(java.awt.event.KeyEvent evt) {
787        if (jTextField74.getText().length() > 10) {
788            jTextField74.setEditable(false);
789        }
790    }
791
792    /**
793      * @author hp
794      */
795    private void jTextField75KeyTyped(java.awt.event.KeyEvent evt) {
796        if (jTextField75.getText().length() > 10) {
797            jTextField75.setEditable(false);
798        }
799    }
799
800    /**
801      * @author hp
802      */
803    private void jTextField76KeyTyped(java.awt.event.KeyEvent evt) {
804        if (jTextField76.getText().length() > 10) {
805            jTextField76.setEditable(false);
806        }
807    }
808
809    /**
810      * @author hp
811      */
812    private void jTextField77KeyTyped(java.awt.event.KeyEvent evt) {
813        if (jTextField77.getText().length() > 10) {
814            jTextField77.setEditable(false);
815        }
816    }
817
818    /**
819      * @author hp
820      */
821    private void jTextField78KeyTyped(java.awt.event.KeyEvent evt) {
822        if (jTextField78.getText().length() > 10) {
823            jTextField78.setEditable(false);
824        }
825    }
826
827    /**
828      * @author hp
829      */
830    private void jTextField79KeyTyped(java.awt.event.KeyEvent evt) {
831        if (jTextField79.getText().length() > 10) {
832            jTextField79.setEditable(false);
833        }
834    }
835
836    /**
837      * @author hp
838      */
839    private void jTextField80KeyTyped(java.awt.event.KeyEvent evt) {
840        if (jTextField80.getText().length() > 10) {
841            jTextField80.setEditable(false);
842        }
843    }
844
845    /**
846      * @author hp
847      */
848    private void jTextField81KeyTyped(java.awt.event.KeyEvent evt) {
849        if (jTextField81.getText().length() > 10) {
850            jTextField81.setEditable(false);
851        }
852    }
853
854    /**
855      * @author hp
856      */
857    private void jTextField82KeyTyped(java.awt.event.KeyEvent evt) {
858        if (jTextField82.getText().length() > 10) {
859            jTextField82.setEditable(false);
860        }
861    }
862
863    /**
864      * @author hp
865      */
866    private void jTextField83KeyTyped(java.awt.event.KeyEvent evt) {
867        if (jTextField83.getText().length() > 10) {
868            jTextField83.setEditable(false);
869        }
870    }
871
872    /**
873      * @author hp
874      */
875    private void jTextField84KeyTyped(java.awt.event.KeyEvent evt) {
876        if (jTextField84.getText().length() > 10) {
877            jTextField84.setEditable(false);
878        }
879    }
879
880    /**
881      * @author hp
882      */
883    private void jTextField85KeyTyped(java.awt.event.KeyEvent evt) {
884        if (jTextField85.getText().length() > 10) {
885            jTextField85.setEditable(false);
886        }
887    }
887
888    /**
889      * @author hp
890      */
891    private void jTextField86KeyTyped(java.awt.event.KeyEvent evt) {
892        if (jTextField86.getText().length() > 10) {
893            jTextField86.setEditable(false);
894        }
895    }
895
896    /**
897      * @author hp
898      */
899    private void jTextField87KeyTyped(java.awt.event.KeyEvent evt) {
900        if (jTextField87.getText().length() > 10) {
901            jTextField87.setEditable(false);
902        }
903    }
903
904    /**
905      * @author hp
906      */
907    private void jTextField88KeyTyped(java.awt.event.KeyEvent evt) {
908        if (jTextField88.getText().length() > 10) {
909            jTextField88.setEditable(false);
910        }
911    }
911
912    /**
913      * @author hp
914      */
915    private void jTextField89KeyTyped(java.awt.event.KeyEvent evt) {
916        if (jTextField89.getText().length() > 10) {
917            jTextField89.setEditable(false);
918        }
919    }
919
920    /**
921      * @author hp
922      */
923    private void jTextField90KeyTyped(java.awt.event.KeyEvent evt) {
924        if (jTextField90.getText().length() > 10) {
925            jTextField90.setEditable(false);
926        }
927    }
927
928    /**
929      * @author hp
930      */
931    private void jTextField91KeyTyped(java.awt.event.KeyEvent evt) {
932        if (jTextField91.getText().length() > 10) {
933            jTextField91.setEditable(false);
934        }
935    }
935
936    /**
937      * @author hp
938      */
939    private void jTextField92KeyTyped(java.awt.event.KeyEvent evt) {
940        if (jTextField92.getText().length() > 10) {
941            jTextField92.setEditable(false);
942        }
943    }
943
944    /**
945      * @author hp
946      */
947    private void jTextField93KeyTyped(java.awt.event.KeyEvent evt) {
948        if (jTextField93.getText().length() > 10) {
949            jTextField93.setEditable(false);
950        }
951    }
951
952    /**
953      * @author hp
954      */
955    private void jTextField94KeyTyped(java.awt.event.KeyEvent evt) {
956        if (jTextField94.getText().length() > 10) {
957            jTextField94.setEditable(false);
958        }
959    }
959
960    /**
961      * @author hp
962      */
963    private void jTextField95KeyTyped(java.awt.event.KeyEvent evt) {
964        if (jTextField95.getText().length() > 10) {
965            jTextField95.setEditable(false);
966        }
967    }
967
968    /**
969      * @author hp
970      */
971    private void jTextField96KeyTyped(java.awt.event.KeyEvent evt) {
972        if (jTextField96.getText().length() > 10) {
973            jTextField96.setEditable(false);
974        }
975    }
975
976    /**
977      * @author hp
978      */
979    private void jTextField97KeyTyped(java.awt.event.KeyEvent evt) {
980        if (jTextField97.getText().length() > 10) {
981            jTextField97.setEditable(false);
982        }
983    }
983
984    /**
985      * @author hp
986      */
987    private void jTextField98KeyTyped(java.awt.event.KeyEvent evt) {
988        if (jTextField98.getText().length() > 10) {
989            jTextField98.setEditable(false);
990        }
991    }
991
992    /**
993      * @author hp
994      */
995    private void jTextField99KeyTyped(java.awt.event.KeyEvent evt) {
996        if (jTextField99.getText().length() > 10) {
997            jTextField99.setEditable(false);
998        }
999    }
999
1000    /**
1001      * @author hp
1002      */
1003    private void jTextField100KeyTyped(java.awt.event.KeyEvent evt) {
1004        if (jTextField100.getText().length() > 10) {
1005            jTextField100.setEditable(false);
1006        }
1007    }
1007
1008    /**
1009      * @author hp
1010      */
1011    private void jTextField101KeyTyped(java.awt.event.KeyEvent evt) {
1012        if (jTextField101.getText().length() > 10) {
1013            jTextField101.setEditable(false);
1014        }
1015    }
1015
1016    /**
1017      * @author hp
1018      */
1019    private void jTextField102KeyTyped(java.awt.event.KeyEvent evt) {
1020        if (jTextField102.getText().length() > 10) {
1021            jTextField102.setEditable(false);
1022        }
1023    }
1023
1024    /**
1025      * @author hp
1026      */
1027    private void jTextField103KeyTyped(java.awt.event.KeyEvent evt) {
1028        if (jTextField103.getText().length() > 10) {
1029            jTextField103.setEditable(false);
1030        }
1031    }
1031
1032    /**
1033      * @author hp
1034      */
1035    private void jTextField104KeyTyped(java.awt.event.KeyEvent evt) {
1036        if (jTextField104.getText().length() > 10) {
1037            jTextField104.setEditable(false);
1038        }
1039    }
1039
1040    /**
1041      * @author hp
1042      */
1043    private void jTextField105KeyTyped(java.awt.event.KeyEvent evt) {
1044        if (jTextField105.getText().length() > 10) {
1045            jTextField105.setEditable(false);
1046        }
1047    }
1047
1048    /**
1049      * @author hp
1050      */
1051    private void jTextField106KeyTyped(java.awt.event.KeyEvent evt) {
1052        if (jTextField106.getText().length() > 10) {
1053            jTextField106.setEditable(false);
1054        }
1055    }
1055
1056    /**
1057      * @author hp
1058      */
1059    private void jTextField107KeyTyped(java.awt.event.KeyEvent evt) {
1060        if (jTextField107.getText().length() > 10) {
1061            jTextField107.setEditable(false);
1062        }
1063    }
1063
1064    /**
1065      * @author hp
1066      */
1067    private void jTextField108KeyTyped(java.awt.event.KeyEvent evt) {
1068        if (jTextField108.getText().length() > 10) {
1069            jTextField108.setEditable(false);
1070        }
1071    }
1071
1072    /**
1073      * @author hp
1074      */
1075    private void jTextField109KeyTyped(java.awt.event.KeyEvent evt) {
1076        if (jTextField109.getText().length() > 10) {
1077            jTextField109.setEditable(false);
1078        }
1079    }
1079
1080    /**
1081      * @author hp
1082      */
1083    private void jTextField110KeyTyped(java.awt.event.KeyEvent evt) {
1084        if (jTextField110.getText().length() > 10) {
1085            jTextField110.setEditable(false);
1086        }
1087    }
1087
1088    /**
1089      * @author hp
1090      */
1091    private void jTextField111KeyTyped(java.awt.event.KeyEvent evt) {
1092        if (jTextField111.getText().length() > 10) {
1093            jTextField111.setEditable(false);
1094        }
1095    }
1095
1096    /**
1097      * @author hp
1098      */
1099    private void jTextField112KeyTyped(java.awt.event.KeyEvent evt) {
1100        if (jTextField112.getText().length() > 10) {
1101            jTextField112.setEditable(false);
1102        }
1103    }
1103
1104    /**
1105      * @author hp
1106      */
1107    private void jTextField113KeyTyped(java.awt.event.KeyEvent evt) {
1108        if (jTextField113.getText().length() > 10) {
1109            jTextField113.setEditable(false);
1110        }
1111    }
1111
1112    /**
1113      * @author hp
1114      */
1115    private void jTextField114KeyTyped(java.awt.event.KeyEvent evt) {
1116        if (jTextField114.getText().length() > 10) {
1117            jTextField114.setEditable(false);
1118        }
1119    }
1119
1120    /**
1121      * @author hp
1122      */
1123    private void jTextField115KeyTyped(java.awt.event.KeyEvent evt) {
1124        if (jTextField115.getText().length() > 10) {
1125            jTextField115.setEditable(false);
1126        }
1127    }
1127
1128    /**
1129      * @author hp
1130      */
1131    private void jTextField116KeyTyped(java.awt.event.KeyEvent evt) {
1132        if (jTextField116.getText().length() > 10) {
1133            jTextField116.setEditable(false);
1134        }
1135    }
1135
1136    /**
1137      * @author hp
1138      */
1139    private void jTextField117KeyTyped(java.awt.event.KeyEvent evt) {
1140        if (jTextField117.getText().length() > 10) {
1141            jTextField117.setEditable(false);
1142        }
1143    }
1143

```

The screenshot shows the Apache NetBeans IDE 12.4 interface. The title bar reads "RentaRide2 - Apache NetBeans IDE 12.4". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help. The toolbar has icons for New Project, Open Project, Save, Run, Stop, and Exit. The left sidebar shows the Projects tree with "RentaRide2" selected, and the Services tree with various database and server connections. The main editor window displays Java code for a class named "Vehicles.java". The code handles action events for vehicle status, type, location, and save actions, and performs database insertions. The code is annotated with TODO comments for handling code. The bottom status bar shows the time as 22:50, date as 17-Aug-21, and system information like temperature (23°C) and battery level (15:42).

```
private void veh_statusActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void veh_typeActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void veh_locationActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void btn_save_vehActionPerformed(java.awt.event.ActionEvent evt) {
    if(reg_numb.getText().isEmpty() || veh_make.getText().isEmpty() || veh_color.getText().isEmpty() || veh_status.getSelectedIndex() == 0)
    {
        JOptionPane.showMessageDialog(this, "Missing values");
    }
    else{
        try {
            con= DriverManager.getConnection("jdbc:derby://localhost:1527/rentaridedb","root","root");
            PreparedStatement add=con.prepareStatement("insert into vehicles values(?,?,?,?,?,?)");
            add.setString(1,reg_numb.getText());
            add.setString(2,veh_make.getText());
            add.setString(3,veh_color.getText());
            add.setString(4,veh_status.getSelectedItem().toString());
            add.setString(5,veh_type.getSelectedItem().toString());
            add.setInt(6, Integer.valueOf(veh_price.getText()));
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

Conclusion and recommendations

The conclusion is that every software required a considerable amount of planning and executing in order for it to be a success or else it will not be properly developed. Moreover, factors should be considered when choosing the IDE to implement since the database will also affect the IDE that one is going to choose in implementing the software.

The development process also doesn't always go as planned and therefore the using of agile methods is the best for application and software development in order to have the best version of the final product.

References

- M, K., 2021. *Car Rental Software: The Best Guide 2021* *Car Rental Software: The Best Guide 2021*. [online] Fleetroot.com. Available at: <<https://fleetroot.com/blog/car-rental-software-the-best-guide-2020/>> [Accessed 2 June 2021].
- TestMyPrep.com. 2021. *Car Hire System Literature Review*. [online] Available at: <<https://testmyprep.com/subject/computer-science/car-hire-system-literature-review>> [Accessed 2 June 2021].
- Design, 1., & Yalanska, M. (2021). 17 Inspiring Examples of Mobile Interaction Design | Design4Users. Retrieved 22 June 2021, from <https://design4users.com/mobile-interaction-design-examples/>
- UML - Use Case Diagrams - Tutorialspoint. (2021). Retrieved 22 June 2021, from https://www.tutorialspoint.com/uml/uml_use_case_diagram.htm
- creately.com. (2021). Retrieved 22 June 2021, from <https://creately.com/blog/diagrams/uml-diagram-types-examples/>