GROUP ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT038-3.5-2-OODJ

OBJECT-ORIENTED DEVELOPMENT WITH JAVA

HAND OUT DATE: 19th SEPTEMBER 2022

HAND IN DATE: 9th DECEMBER 2022

WEIGHTAGE: 50%

INSTRUCTIONS TO CANDIDATES:

1. **Submit your assignment at the administrative counter**
2. **Students are advised to underpin their answers with the use of references (cited using the Harvard Name System of Referencing)**
3. **Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld**
4. **Cases of plagiarism will be penalized**
5. **The assignment should be bound in an appropriate style (comb bound or stapled).**
6. **Where the assignment should be submitted in both hardcopy and softcopy, the softcopy of the written assignment and source code (where appropriate) should be on a CD in an envelope / CD cover and attached to the hardcopy.**
7. **You must obtain 50% overall to pass this module.**

Table of Contents

1.0 Introduction

This car rental system was developed for Premium Car Rental in Kuala Lumpur. The two target users for this system are the admin and the customer. The administrator of Premium Car Rental in Kuala Lumpur may manage the car information and add new cars to the system by using this car rental system. Additionally, admin may use this system to manage customer bookings by confirming when a booking is registered. Customers of Premium Car Rental in Kuala Lumpur, on the other hand, must register as customers before booking online. After placing a booking, users may access the car rental system to see their booking history. Additionally, object-oriented concepts and principles were used in the design and development of this car rental system for Premium Car Rental in Kuala Lumpur. This car rental system's data will all be stored in text files.

When user goes into the car rental system, users will have to select their user type which are customer or admin. User needs to register an account before entering into the system. Customer can login to the system with their credentials. If customer does not have an account, customer can register an account and proceed to login. After customer login, customer search for available cars to make a booking. After a booking is made, the booking status will be “Pending” until the admin confirmed the booking. Customers can also update their bookings. However, the booking status will be set as “Pending” again until the admin approved the modification of booking. To delete a booking, customer needs to delete the booking at least 2 days before the pickup datetime. Furthermore, customer can check their bookings and view the booking details. Moreover, customers are able to provide feedback and view the booking receipt. Lastly, customers will be able to edit their profile in the system by their own.

As for admin in the car rental system, there are two types of admin which are “Default Admin” and “System Admin”. The default admin is a default account that is able to approve and reject the registration of system admin. To access the system, the system admin needs approval. The system admin will be able to manage car data. For instance, add new cars, edit car information, and delete cars. Next, system admin can also manage all customers' booking of the Premium Car Rental in Kuala Lumpur. The system admin will be able to make booking, update booking and delete booking. In addition, the system admin will be able to manage booking confirmation by approving or rejecting the booking registered by customer. It is also

indisputable that the system admin will be able to see all of the customer booking information as well as the booking receipts. After a booking is approved, the customer will approach the pickup location and make the final payment for the car rental. At this stage, the system admin may manage the payment status by updating the payment collection of the booking. Whenever a car is broken or used for other reasons, the system admin can block the car and unblock the car once the car is available again. Besides, the system admin can manage customer registration including adding customers, updating customers' profiles, and deleting customers. After a car rental ended, the customer will return the car to the return location. The car will be returned and the fine will be calculated if the car return datetime is over 24 hours of the actual return datetime. The fine calculated will be double the car's normal daily rate. The is also report generated in the system which contains the rental summary, sales report, and car report.

As there are two types of admin in this car rental system for Premium Car Rental in Kuala Lumpur, the default admin has two additional features that system admin cannot access. The first feature will be managing administrator registration, the default admin will be able to add admin, approve admin registration, reject admin registration and delete admin. Last but not least, the second additional feature of default admin will be to view the login record of the system which includes the login time and logout time of all users in the system.

2.0 Requirement Analysis

2.1 Use case diagram with description

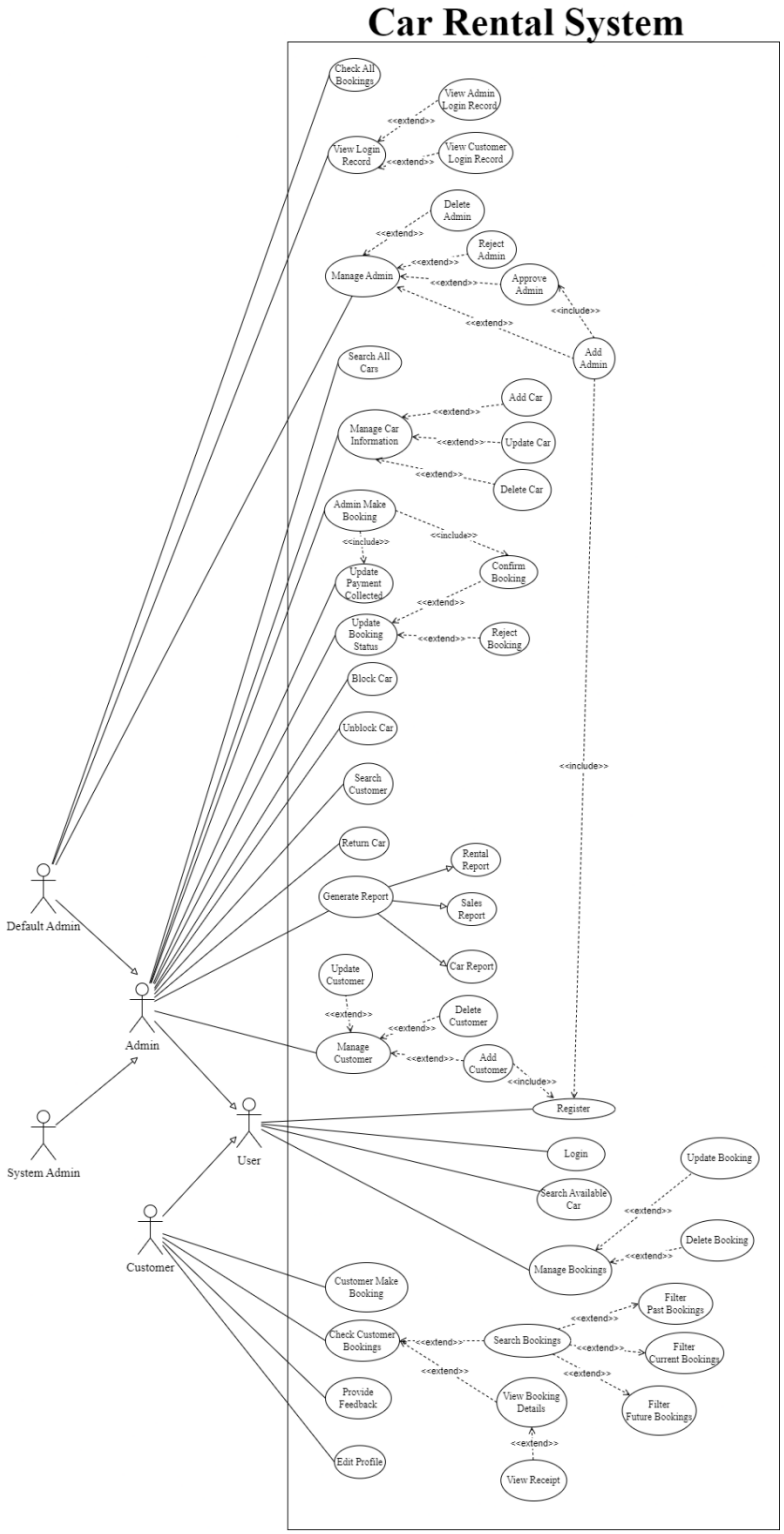


Figure 1 Use Case Diagram

2.1.1 Use Case Specification

Use Case	Register
Brief Description	Actors may register for a system account using this use case.
Actors	User, Admin, Customer, Default Admin, System Admin
Preconditions	The actors never created a system account before.
Main Flow	(a) When a user selects their user type and decides to register an account, this use case begins. (b) The system asks the user to complete all of the registration form's required fields (ref. Alternative flow). (c) The system does a database search and displays success message when the entered personal information is valid (ref. Alternative flow). (d) The Default admin creates an account for the system admin. (ref. Alternative flow).
Alternative Flows	(a) (i) The actor does not enter valid information while filling in the form. (b) (i) The system searched the database and identified the actor had registered an account previously. The actor will be asked to proceed to login. (ii) The system searched the database and identified the actor does not register an account before. The username chosen by the actor is already taken. The actor will be asked to pick another username. (c) (i) The system admin account that the default admin wants to create never create an admin account in the system before.

Use Case	Login
Brief Description	Customers and administrators may log into the system using this use case.
Actors	User, Customer, Admin, Default Admin, System Admin

Preconditions	The actors must register for an account, remember their username and password.
Main Flow	(a) The actor chooses their user type to begin this use case. (b) The actors are prompted by the system to submit a valid login and password (ref. Alternative flow). (c) If the username and password are correct, the system does a database search and shows a success message (ref. Alternative flow).
Alternative Flows	(b) (i) The actor does not have an account registered in the system. (ii) The actor does not enter a valid username or password. (c) (i) The database is searched by the system, but it is unable to discover the actor. (ii) The system found a valid username in the database but the password entered is incorrect. (iii) The system found a valid admin account in the database but the account is not approved. To login to the system, the admin account must be approved.

Use Case	Search Available Car
Brief Description	The actors are able to look for available cars in the system with this this use case.
Actors	User, Customer, Admin, Default Admin, System Admin
Preconditions	The system must have the actors' login credentials. The database of the system must include information on the car and the booking.
Main Flow	(a) The actor searches for available cars to rent when using this use case. (b) The system asks the actor to choose a pickup location, pickup date, pickup time, and pickup date (ref. Alternative flow).

	(c) The system does a database search and shows the car details in the table when the car is available (ref. Alternative flow).
Alternative Flows	(a) (i) Not all of the information required to find a car is entered by the actor. (ii) The system will warn the actors when the return date is before pickup date. (iii) The system will warn the actors when the return date or pickup date is before today. (iv) The system will also warn the actors when the pickup datetime is later than the current datetime. (b) (i) The system searches the database but is unable to to find available car. There will be a message to notice the actor that no cars are available for the chosen location and datetime.

Use Case	Customer Make Booking
Brief Description	With the help of this use case, customers may make online bookings for car rentals.
Actors	User, Customer
Preconditions	The customer must be logged into the system. The customer must provide the car number plate, pickup time, return time, pickup location, pickup date and return date.
Main Flow	(a) When a consumer decides to reserve an available car, this use case begins. (b) The system figures out the rental fee total, pay at pickup, and online payments. (c) The system requests customer to fill up all the booking information required in the form (ref. Alternative flow). (d) The system will clear all the entered booking information when the Reset option is selected.

	(e) The system will store all the booking information into database and change the status of the car to "RENTED" after validation (ref. Alternative flow).
Alternative Flows	(a) (i) The booking form is incomplete because the actor missed some necessary details. (b) (i) When the email address and re-enter email address entered do not match, the system will alert the actors. (ii) The system will warn the actors when the return date or pickup date is before today.

Use Case	Check Customer Bookings
Brief Description	Customers may verify their own bookings by using this use case.
Actors	Customer
Preconditions	A user account must be created for the consumer. The customer has previously placed a booking using the system.
Main Flow	(a) When a consumer requests to verify their own booking, this use case begins. (b) The system searches the database and shows the customer's bookings in a table (ref. Alternative flow).
Alternative Flows	(a) (i) The database search by the system did not find the customer's booking. No bookings will be shown to the customer at the table.

Use Case	Manage Bookings
Brief Description	Customers may manage their own bookings using this use case.
Actors	User, Customer
Preconditions	The customer must be logged into the system. The customer has previously placed a booking using the system.
Main Flow	(a) The customer requests to manage their own booking when this use case begins.

	(b) The system searches the database and shows the customer's bookings in a table (ref. Alternative flow).
Alternative Flows	(a) (i) The system searched the database but was unable to find the customer's booking. No bookings will be shown to the customer at the table.

Use Case	Update Bookings
Brief Description	The customer may edit their own booking using this use case.
Actors	User, Customer
Preconditions	The customer must be logged into the system. The customer had made booking in the system before.
Main Flow	(a) This use case begins when a consumer requests an update to their own booking. (b) The system searches the database and presents the customer's bookings on the Manage Bookings page (ref. Alternative flow).
Alternative Flows	(a) (i) The database search by the system did not find the customer's booking. The customer may use this feature even if no reservations will be shown on the Manage Bookings page. (ii) The system will warn if customer doesn't fill in all the info into the columns.

Use Case	Delete Bookings
Brief Description	With this use case, customers may cancel their own bookings.
Actors	User, Customer
Preconditions	The customer must be logged into the system. The customer had made booking in the system before.
Main Flow	(a) This use case begins when a customer wants to cancel their own booking.

	(b) The system searches the database and presents the customer's bookings on the Manage Bookings page (ref. Alternative flow).
Alternative Flows	(a) (i) The database search by the system did not find the customer's booking. The customer may use this feature even if no reservations will be shown on the Manage Bookings page. (ii) The system will warn if the booking is passed. (iii) The system will warn if the date between booking date no more than two days.

Use Case	Provide Feedback
Brief Description	With this use case, customers may comment on their own bookings.
Actors	User, Customer
Preconditions	The customer must be logged into the system. The customer had made booking in the system before.
Main Flow	(a) This use case begins when a consumer wants to provide feedback about their own reservation. (b) The system searches the database and shows the customer's bookings in a table (ref. Alternative flow).
Alternative Flows	(a) (i) The system searched the database but was unable to find the customer's booking. No bookings will show up in the table on the My Bookings page for the customer.

Use Case	Edit Profile
Brief Description	The consumer may change their own profile information using this use case.
Actors	User, Customer
Preconditions	The customer must be registered into the system.
Main Flow	(a) This use case begins when a customer wants to change the information on their own profile.

	(b) The system searches the database and displays the user's information in the text field (ref. Alternative flow).
Alternative Flows	(a) (i) If a customer is not registered, they cannot access the Edit Profile page. (ii) The system seek for the database and could not find the IC number of the customer. The customer will not see any details appear in the text field and will not be able to edit their profile.

Use Case	Search Bookings
Brief Description	Customers may search reservations with this use case.
Actors	User, Customer
Preconditions	The customer must be registered into the system. The customer had made booking in the system before.
Main Flow	(a) This use case begins when a customer wants to look up one of their own bookings. (b) The system searches the database and shows all of the bookings in the table (ref. Alternative flow).
Alternative Flows	(a) The system searched the database using the information in the search field but was unable to find the customer's booking. No reservations will show up in the table on the My Bookings page for the customer.

Use Case	View Booking Details
Brief Description	Customers may see their own bookings using this use case.
Actors	User, Customer
Preconditions	The customer must be logged into the system. The customer had previously made a booking in the system.
Main Flow	(a) The customer requests to view their own booking when this use case begins.

	(b) The customer selects the booking and sees the customer's booking information (ref. Alternative flow).
Alternative Flows	(a) (i) If the customer does not choose any bookings, they will not see any booking data.

Use Case	View Receipt
Brief Description	The customer may view their own booking receipt with the help of this use case.
Actors	User, Customer
Preconditions	The customer must be logged into the system. The customer has already placed a booking in the system.
Main Flow	(a) The customer requests to check their own booking receipt when this use case begins. (b) The customer clicks the receipt button, which displays the customer's booking receipt (ref. Alternative flow).
Alternative Flows	(a) (i) The customer will not see any booking receipt if the customer not click the receipt button.

Use Case	Filter Past Bookings
Brief Description	This use case allows the customer to view their own booking according to the date.
Actors	User, Customer
Preconditions	The customer must be logged into the system. The customer has already placed a booking in the system.
Main Flow	(a) This use case begins when a customer requests to view their own bookings based on the date. (b) The customer hits the PAST button, which displays the customer's previous reservations (ref. Alternative flow).
Alternative Flows	(a) (i) If there have been no previous bookings, the customer will not see any booking information.

Use Case	Filter Current Bookings
Brief Description	This use case allows the customer to view their own booking according to the date.
User, Customer	User, Customer
Preconditions	The customer must be logged into the system. The customer has already placed a booking in the system.
Main Flow	(a) This use case begins when a customer requests to view their own bookings based on the date. (b) The customer hits the NOW button, which displays the customer's previous reservations (ref. Alternative flow).
Alternative Flows	(a) (i) If no current booking is available, the customer won't see any booking information.

Use Case	Filter Future Bookings
Brief Description	This use case allows the customer to view their own booking according to the date.
Actors	User, Customer
Preconditions	The customer must be logged into the system. The customer had previously made a booking in the system.
Main Flow	(a) This use case begins when a customer requests to view their own bookings based on the date. (b) The customer clicks the FUTURE button, which displays the customer's future booking (ref. Alternative flow).
Alternative Flows	(a) (i) If there are no upcoming bookings, the customer won't see any booking information.

Use Case	Manage Customer
Brief Description	Using this use case, admin may manage customer accounts.
Actors	User, Admin, System Admin, Default Admin

Preconditions	The admin must be logged into the system. The customer had completed the register form in the system.
Main Flow	(a) The start of this use case occurs when an admin wants to create, amend, or remove a customer profile. (b) The system will search through the database and discover the customer with the search phrase. (c) The admin will be able to look for the customer with any customer information (ref. Alternative flow).
Alternative Flows	(c) (i) When a customer is not found in the database, a pop-up page will appear to display the customer. (ii) The admin will be able to see the customers in the database that matches the search keyword.

Use Case	Add Customer
Brief Description	This use case enables admin to add customer accounts.
Actors	User, Admin, System Admin, Default Admin
Preconditions	The admin must be logged into the system. The customer never registers an account in the system.
Main Flow	(a) When an administrator wants to create a customer profile, this use case begins (b) The admin must complete all fields on the registration form. (c) The system will verify the admin's data input (ref. Alternative flow). (d) In order to determine if this admin had registered with the system, the system will search in the database (ref. Alternative flow).
Alternative Flows	(c) (i) A warning message would be shown to the admin and admin needs to fill in the correct form of data. (d) (i) A warning message would be shown to the admin and notice the admin that this customer already registered in the system. Admin can delete or update the customer account.

Use Case	Update Customer
Brief Description	Using this use case, an admin can update or modify a customer account.
Actors	User, Admin, System Admin, Default Admin
Preconditions	The admin must be logged into the system. The customer had an account registered in the system previously.
Main Flow	(a) The start of this use case occurs when an admin wants to edit a customer profile. (b) The admin is able to edit customer details in the register form. (c) The system will validate the data modified by the admin (ref. Alternative flow).
Alternative Flows	(c) (i) A warning message would be shown to the admin and admin needs to fill in the correct form of data.

Use Case	Delete Customer
Brief Description	A customer account may be deleted or removed by an admin using this use case.
Actors	User, Admin, System Admin, Default Admin
Preconditions	The system's admin has to be logged in. The customer had an account registered in the system previously.
Main Flow	(a) This use case begins when an admin wants to remove a customer profile. (b) The admin is able to remove customer from the system’s database.

Use Case	Generate Report
Brief Description	The admin may see the reports produced using the system's database with this use case.
Actors	User, Admin, System Admin, Default Admin

Preconditions	The admin must be logged into the system.
Main Flow	(a) When an admin wants to produce a report, this use case begins. (b) The admin will be able to see the generated reports (ref. Alternative flow).
Alternative Flows	(b) (i) The system searched the database, but there were insufficient data to compile a report. The admin won't be able to produce reports.

Use Case	Rental Report
Brief Description	This use case allows the admin to view the system's rental report.
Actors	User, Admin, System Admin, Default Admin
Preconditions	The system's admin has to be logged in.
Main Flow	(a) This use case begins when the admin wants to look into the rental report. (b) The admin will be able to see the generated rental report (ref. Alternative flow).
Alternative Flows	(c) (i) The system searched the database but couldn't find any rental information to create a report. The rental report won't be visible to the admin.

Use Case	Sales Report
Brief Description	This use case allows the admin to view the system's rental report.
Actors	User, Admin, System Admin, Default Admin
Preconditions	The admin must be logged into the system.
Main Flow	(a) This use case begins when the admin wants to look into the rental report. (b) The admin will be able to see the generated rental report (ref. Alternative flow).

Alternative Flows	(b) (i) The system search through the database but there are no sales data to generate report. The admin will not be able view the sales report.

Use Case	Car Report
Brief Description	This use case allows the admin to examine the system's
Actors	User, Admin, System Admin, Default Admin
Preconditions	The system's admin has to be logged in.
Main Flow	(a) When the admin wants to look into the car report, this use case begins. (b) The admin will be able to see the report generated (ref. Alternative flow).
Alternative Flows	(b) (i) After searching the database, the system was unable to discover any rental data needed to generate a report. The admin won't be able to see the rental report.

Use Case	Return Car
Brief Description	After the car rental period is over and the customers have returned the car, this use case allows the admin to return the car to the system.
Actors	User, Admin, System Admin, Default Admin
Preconditions	The admin must be logged into the system. The customer had rent a car that has not been returned.
Main Flow	(a) This use case begins when the admin decides to add the car back into the system. (b) The admin will have the option to look for or choose a rental car to return the car (ref. Alternative flow). (c) The admin needs to fill in the required information of return car (ref. Alternative flow). (d) Admin needs to collect fine from the customer.

Alternative Flows	(b) (i) The system searched the database using the admin-entered search term, but it found no car rentals. (c) (i) The admin is warned whenever data of return car is missing or incorrect. Admin needs to re-enter information correctly. (d) (i) When the car is returned after 24 hours of the return date, admin needs to collect the fine from the customer. After the fine is collected, the car is returned successfully. (ii) When car is returned 24 hours after the return date, no fines need to be collected. The car is successfully returned.

Use Case	Search Customer
Brief Description	This use case allows the admin to look up existing customers in the system.
Actors	User, Admin, System Admin, Default Admin
Preconditions	The system's admin has to be logged in.
Main Flow	(a) This use case begins when an administrator wants to look up a customer in the system. (b) The admin will be able to do a customer search by inputting a user's data (ref. Alternative flow).
Alternative Flows	(b) (i) The system searched the database but couldn't find any user profiles that matched the admin's search term. (ii) The system search through the database and there are user profiles that matches with the search keyword that the admin entered. The user accounts will be displayed in the table.

Use Case	Unblock Car
Brief Description	The admin may unlock cars in the system using this use case.
Actors	User, Admin, System Admin, Default Admin
Preconditions	The system's admin has to be logged in.

Main Flow	(a) This use case begins when an administrator wants to unlock a car and make it available for rental in the system. (b) The admin will be able to search car by entering a keyword of the car (ref. Alternative flow). (c) The car will be available for rent after it is unblocked (ref. Alternative flow).
Alternative Flows	(b) (i) The database was searched by the system, but there were no car records that matched the search term that the admin inputted. (ii) The system search through the database and there are car data that matches with the search keyword that the admin entered. The cars will be displayed in the table. (c) (i) The car status will be changed to "Available" and will be rent by customers.

Use Case	Block Car
Brief Description	Administrators may block cars in the system using this use case.
Actors	User, Admin, System Admin, Default Admin
Preconditions	The system's admin has to be logged in.
Main Flow	(a) This use case begins when an administrator wishes to block a car and prevent it from being rented thru the system. (b) The admin will be able to search car by entering a keyword of the car (ref. Alternative flow). (c) The car will not be available for rent after it is blocked (ref. Alternative flow).
Alternative Flows	(b) (i) The database was searched by the system, but there were no car records that matched the search term that the admin submitted. (ii) The system search through the database and there are car data that matches with the search keyword that the admin entered. The cars will be displayed in the table.

	(c) (i) The car status will be changed to "Blocked" and will not be to rent by customers. The reason of blocking car may be broken, under repair, go for services or other reasons.

Use Case	Update Booking Status
Brief Description	This use case allows the admin to modify the system's booking status.
Actors	User, Admin, System Admin, Default Admin
Preconditions	The system's admin has to be logged in.
Main Flow	(a) The start of this use case occurs when the admin wants to accept or reject a booking in the system. (b) The admin will be able to search bookings by inputting a bookings keyword (ref. Alternative flow). (c) Select the booking and decide whether to accept or reject the booking (ref. Alternative flow).
Alternative Flows	(b) (i) When the admin entered a search term, the system searched the database but found no bookings data that matched it. (ii) The system search through the database and there are bookings data that matches with the search keyword that the admin entered. The bookings will be displayed in the table. (c) (i) The bookings status will be changed according to the admin's option.

Use Case	Reject Booking
Brief Description	The admin may reject the booking in the system using this use case.
Actors	User, Admin, System Admin, Default Admin
Preconditions	The system's admin has to be logged in.
Main Flow	(a) This use case begins when the admin wants to reject the booking in the system. (b) The admin will be able to search bookings by entering a keyword of the bookings (ref. Alternative flow).

	(c) Select the booking and choose to reject the booking (ref. Alternative flow).
Alternative Flows	(b) (i) When the admin entered a search term, the system searched the database but found no bookings data that matched it. (ii) The system search through the database and there are bookings data that matches with the search keyword that the admin entered. The bookings will be displayed in the table. (c) (i) The bookings status will be changed to "Rejected". Admin will call the customer and inform them to modify their booking details.

Use Case	Confirm Booking
Brief Description	The admin may confirm the booking in the system using this use case.
Actors	User, Admin, System Admin, Default Admin
Preconditions	The admin must be logged into the system.
Main Flow	(a) This use case begins when the admin wants to confirm the booking in the system. (b) The admin will be able to search bookings by entering a keyword of the bookings (ref. Alternative flow). (c) Select the booking and choose to confirm the booking (ref. Alternative flow). (d) The bookings created by admin will be confirmed automatically (ref. Alternative flow).
Alternative Flows	(b) (i) When the admin entered a search term, the system searched the database but found no bookings data that matched it. (ii) The system search through the database and there are bookings data that matches with the search keyword that the admin entered. The bookings will be displayed in the table. (c) (i) The bookings status will be changed to "Approved". The customer will be able to continue with the car rental.

	(d) (i) The bookings status will be changed to “Approved” when the bookings are created.

Use Case	Update Payment Collected
Brief Description	When customers pay to rent a car, this use case allows the admin to update the money received in the system.
Actors	User, Admin, System Admin, Default Admin
Preconditions	The system's admin has to be logged in. The customer must have a booking that is registered in the system.
Main Flow	(a) This use case begins when an admin wants to change the booking's payment status in the system. (b) The admin will be able to search bookings by inputting a bookings keyword (ref. Alternative flow). (c) Select the booking and choose to update payment for the booking (ref. Alternative flow). (d) The bookings created by admin will be update the payment status automatically (ref. Alternative flow).
Alternative Flows	(b) (i) The system search through the database but there are no bookings data that matches with the search keyword that the admin entered. (ii) The system search through the database and there are bookings data that matches with the search keyword that the admin entered. The payment status of the bookings found must be “Paid Online”. The bookings will be displayed in the table. (c) (i) The bookings status will be changed to “Full Payment”. The customer will be able to continue with the car rental. (d) (i) The bookings status will be changed to “Full Payment” when the bookings are created.

Use Case	Admin Make Booking

Brief Description	This use case allows the admin to assist customers with system booking.
Actors	User, Admin, System Admin, Default Admin
Preconditions	The system's admin has to be logged in.
Main Flow	(a) This use case begins when an administrator wants to make a customer car booking in the system. (b) The administrator must input all booking data into the system (ref. Alternative flow). (c) The booking will be added into the system database (ref. Alternative flow).
Alternative Flows	(b) (i) The admin is warned with a pop-up message that the bookings data is missing or entered incorrectly. The admin will re-enter the booking information. (c) (i) The booking information will be stored into the booking database with “Approved“and “Full Payment“ status.

Use Case	Check All Bookings
Brief Description	An administrator may check all customer bookings using this use case.
Actors	User, Admin, Default Admin, System Admin
Preconditions	The admin must be logged into the system.
Main Flow	(a) This use case begins when the admin wants to review the customer's booking. (b) The system searches the database for all booking information and displays all customer bookings in a table (ref. Alternative flow).
Alternative Flows	(b) (i) The system searched the database but was unable to find any bookings. No bookings will show up in the table for the admin to check.

Use Case	View Login Record
Brief Description	This use case makes it possible for the default admin to review the login record.
Actors	User, Admin, Default Admin
Preconditions	The system has to be logged in using the default admin account.
Main Flow	(a) This use case begins when an admin wants to review all record of customers' and administrators' logins. (b) The system does a search through the whole database of login records and displays them all in a table (ref. Alternative flow).
Alternative Flows	(b) (i) The system searched the database but was unable to find any login records. The admin won't see any records enter the table and will instead get an error notice.

Use Case	View Admin Login Record
Brief Description	This use case makes it possible for the default admin to see all of the admin login history.
Actors	User, Admin, Default Admin
Preconditions	The system has to be logged in using the default admin account.
Main Flow	(a) This use case begins when an admin wants to see just the login information for administrators. (b) The system searches the database for all information about admin logins and displays that information in a table (ref. Alternative flow).
Alternative Flows	(b) (i) The system searched the database but was unable to find any login records or information of the admin. The admin won't see any records enter the table and will instead get an error notice.

Use Case	View Customer Login Record

Brief Description	This use case allows the default admin to see all login information of customers.
Actors	User, Admin, Default Admin
Preconditions	The system has to be logged in using the default admin account.
Main Flow	(a) This use case begins when an administrator wants to see all of the customer login information. (b) The system searches the database for all information about a customer's logins and displays that information in a table (ref. Alternative flow).
Alternative Flows	(b) (i) The system searched the database but was unable to locate any login records or customer data there. The admin won't see any records enter the table and will instead get an error notice.

Use Case	Manage Admin
Brief Description	The default admin may manage all admin accounts with this use case.
Actors	User, Admin, Default Admin
Preconditions	The system has to be logged in using the default admin account.
Main Flow	(a) The start of this use case occurs when an admin wants to control the admin accounts. (b) The system searches all admin account in the database and displays every admin account in a table (ref. Alternative flow).
Alternative Flows	(b) (i) The system searched the database but was unable to find any information on admin accounts. The admin won't see any records enter the table and will instead get an error notice.

Use Case	Delete Admin
Brief Description	The default admin may remove the admin account with this use case.
Actors	User, Admin, Default Admin
Preconditions	The system has to be logged in using the default admin account.

Main Flow	(a) When the admin wants to remove the admin account, this use case begins. (b) The default admin chooses a user account, clicks the delete button, and then confirms the action (ref. Alternative flow).
Alternative Flows	(b) (i) If the admin clicks the delete button without selecting any accounts, the system will display an error message. Admin will be unable to remove.

Use Case	Reject Admin
Brief Description	The default admin might reject the admin account with this use case.
Actors	User, Admin, Default Admin
Preconditions	The system has to be logged in using the default admin account.
Main Flow	(a) When the admin wants to reject the admin account, this use case begins. (b) The default admin chooses one account, clicks the "Reject" button, and then confirms their choice (ref. Alternative flow).
Alternative Flows	(b) (i) If the admin presses the reject button without selecting any accounts, the system will display an error message. The admin account won't succeed in rejecting.

Use Case	Approve Admin
Brief Description	In this use case, the admin account may be approved by default admin.
Actors	User, Admin, Default Admin
Preconditions	The system has to be logged in using the default admin account.
Main Flow	(a) When the admin wishes to approve the admin account, this use case begins. (b) The default admin chooses a user account, clicks the approve button, and then confirms the choice (ref. Alternative flow).

Alternative Flows	(b) (i) If the admin presses the approve button without selecting any accounts, the system will display an error notice. The admin account won't be approved.

Use Case	Add Admin
Brief Description	The default admin may create a new admin account with this use case.
Actors	User, Admin, Default Admin
Preconditions	The system has to be logged in using the default admin account.
Main Flow	(a) This use case begins when an admin wants to create a new admin account. (b) The add button is clicked by the default admin, which directs the user to the register page. (c) The default admin must provide all the details for the new admin and click the Add button (ref. Alternative flow).
Alternative Flows	(b) (i) If an admin clicks the add button without entering all necessary information, the system will display an error message. The admin account won't be added.

Use Case	Search All Car
Brief Description	The default admin may view all car in the database with this use case.
Actors	Default Admin, System Admin, Admin, User
Preconditions	The system's admin has to be logged in.
Main Flow	(a) This use case begins when an administrator wants to view the list of cars. (b) The system searches the database for all car information and displays every car in a table (ref. Alternative flow).
Alternative Flows	(b) (i) The database search by the system did not find any car data. The admin won't see any records enter the table and will instead get an error notice.

Use Case	Manage Car Information
Brief Description	Using this use case, an admin may manage car information.
Actors	Default Admin, System Admin, Admin, User
Preconditions	The system's admin has to be logged in.
Main Flow	(a) When an admin wants to manage a car's information, this use case begins. (b) The system searches the database and lists every car in the table (ref. Alternative flow).
Alternative Flows	(b) (i) The system searched the database but was unable to find the car. The admin won't be able to choose the car and won't see any information display in the table.

Use Case	Add Car
Brief Description	A new car may be added by the admin with this use case.
Actors	Default Admin, System Admin, Admin, User
Preconditions	The system's admin has to be logged in.
Main Flow	(a) When admin decides to add a new car, this use case begins. (b) The admin hits the add button, the button will takes them to the page where they can add a car. (c) The admin has to input all the new car information and click add button (ref. Alternative flow).
Alternative Flows	(c) (i) If an admin clicks the add button without entering all necessary information, the system will display an error message. The car will not be added.

Use Case	Update Car
Brief Description	The admin may update the car details with this use case.
Actors	Default Admin, System Admin, Admin, User
Preconditions	The admin must be logged into the system.

Main Flow	(a) The start of this use case occurs when the admin wants to update the car information. (b) The admin clicks the add button and will redirect to the manage car page. (c) The admin must input all the new car information and click update button (ref. Alternative flow).
Alternative Flows	(c) (i) If an admin presses the update button without entering all the necessary information, the system will display an error message. The car won't update.

Use Case	Delete Car
Brief Description	The car and the data may be deleted by the admin with this use case.
Actors	Default Admin, System Admin, Admin, User
Preconditions	The system's admin has to be logged in.
Main Flow	(a) This use case begins when the admin wants to remove the car and the data. (b) The admin selects the delete button, which directs them to the manage cars page. (c) The admin presses the delete button and confirm the decision (ref. Alternative flow).
Alternative Flows	(c) (i) If the admin presses the delete button without the number plate, the system will display an error message. The car won't be able to delete.

2.2 Class Diagram

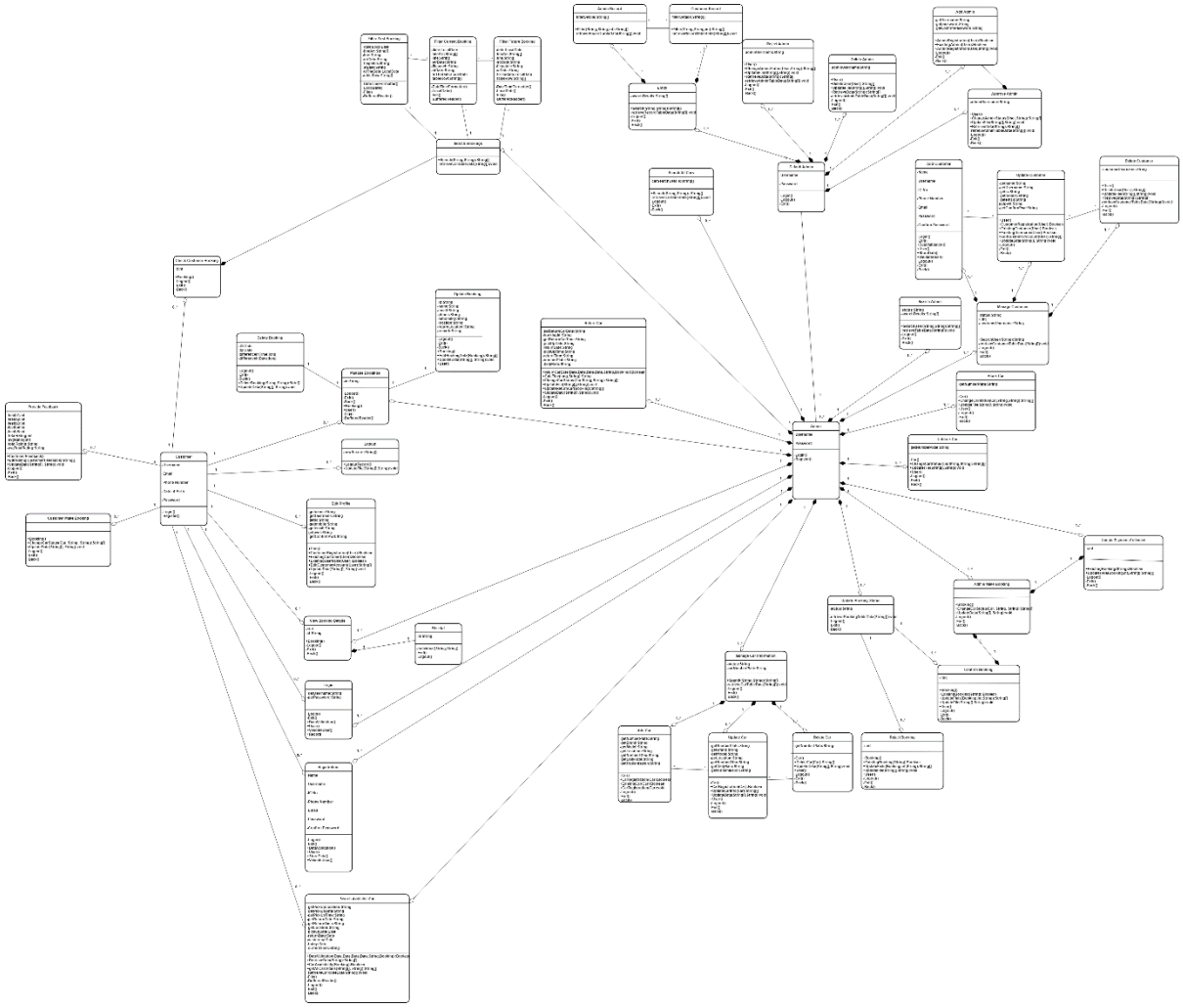


Figure 2 Class Diagram

3.0 Source Code with Explanation

3.1 Java Packages and API

There are some built-in packages in Java which is part of the Java Application Programming Interfaces (API) library. In Java, a package is a way to bring together related classes, packages, and interfaces. Uses for packages include avoiding name conflicts. Access control is provided at the package level for protected and default, limiting access. A protected member may be accessed by classes belonging to the same package and its subclasses. Classes belonging to the same package are the only ones having access to a default member. Packages are an example of data encapsulation. After that, we can utilise it in our system by writing a simple import class from the pre-existing packages. A package is a container for a collection of related classes, where some classes are maintained for internal use while others are made available and exposed. As many times as we need to in our software, we may reuse already-existing classes from the packages (Packages in Java 2022).

3.1.1 Import a Package

```
import data.*;
import model.*;
import java.util.*;
```

Figure 3 Import Packages

The packages in Java can be imported with the use of the asterisk sign (*) after the package name. This will help to import the entire package. The table below will describe the purpose of the packages shown above.

Table: Java Packages

Package	Purpose
data	Self-created package to validate data, check data, store data, update data and delete data for the database from the text files.
model	Self-created package to define private variables and allows all variable to be access with the getter and setter method.
java. util	This package is for collections framework, legacy collection classes, event model, datetime facilities, internalisation and other utility classes (*Java™ Platform, standard edition 8API specification* 2022).

3.1.2 Import a Class and Exception

There is another method to import the package's classes without importing the whole package. The classes can be imported from the package by simply replacing the asterisk sign (*) with the class name.

```
import javax.swing.JOptionPane;
```

Figure 4 Class of Package javax.swing

The JOptionPane class is from the Package javax.swing. To show a dialogue box that requests a value from the user or presents them with information, use the JOptionPane class (*Package javax.swing* 2022).

```
import javax.swing.table.DefaultTableModel;
```

Figure 5 Class of Package javax.swing.table

The DefaultTableModel class is from the Package javax.swing.table. The DefaultTableModel class is used to store the cell values objects in a Vector of Vectors (*Package javax.swing.table* 2022).

```
import java.time.LocalDate;

import java.time.LocalDateTime;
```

Figure 6 Classes of Package java.time

The classes mentioned above are from the Java.time package.The LocalDate class is used to provide date in the ISO-8601 calendar system without a timezone, for instance, 2022-11-06. On the other hand, the LocalDateTime class provide date in the ISO-8601 calendar system with timezone. For example, 2022-11-06T14:50:00 (*Package java.time* 2022).

```
import java.time.format.DateTimeFormatter;
```

Figure 7 Class of Package java.time.format

The Java.time.format package contains the DateTimeFormatter class. Using a formatter, this DateTimeFormatter class produced date-time objects that could be printed and processed (*Package java.time.format* 2022).

```
import data.ModifyUserData;
import data.UpdateData;
```

Figure 8 Classes of Package data

The classes above are from the self-created Package data. The user database's fields, including those for deleting users and login records, may be changed using the ModifyUserData class. As for the UpdateData class, this class is used to update the file data of the file chosen with the string array provided.

```
import model.User;
import model.CustomerFeedback;
import model.Booking;
```

Figure 9 Classes of Package model

The classes above are from the self-created Package model. These classes are used to specify the fields that will be utilised and provide access to the fields using the get set function.

```
import java.util.Scanner;

import java.util.Calendar;

import java.util.Arrays;
import java.util.ArrayList;
import java.util.Date;
```

Figure 10 Classes of Package java.util

The classes above are from the Package java.util. The purpose of the classes is describe in the table below.

Table: Classes of java.util (*Package java.util* 2022)

Class	Description
Scanner	A simple text scanner that extracts primitive types and strings using regular expressions.
Calendar	A method for converting between a certain time period and a collection of calendar data is provided by this class.
Arrays	This class has variety of array manipulation methods.
ArrayList	This class is the Implementation of the List interface using a resizable array.
Date	The millisecond-accurate representation of a specific time is provided by this class.

```
import java.util.logging.Level;
import java.util.logging.Logger;
```

Figure 11 Classes of Package java.util.logging

The classes above are from the Package java.util.logging. A range of common logging levels are defined by the Level class and may be used to manage logging output. With the Logger class, a Logger object is used to log messages for the Car Rental System (*Package java.util.logging* 2022).

```
import java.io.IOException;

import java.io.FileNotFoundException;
```

Figure 12 Classes of Package java.io

The exceptions above are from the Package java.io. The IOException indicates the occurrence of some type of I/O exception. The FileNotFoundException indicates that an unsuccessful attempt was made to open the file indicated by the supplied pathname (*Package java.io* 2022).

```
import java.text.ParseException;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
```

Figure 13 Classes of Package java.text

The classes and exception above are from the Package java.text. The purpose of the classes is described in the table below.

Table: Classes and Exception of Package java.text (*Package java.text* 2022)

Name	Type	Description
ParseException	Exception	This class indicates that a parsing error has been encountered unexpectedly.
DateFormat	Class	A language-independent date or time formatter, DateFormat is an abstract class for date/time formatting subclasses.
SimpleDateFormat	Class	SimpleDateFormat is a concrete class that allows for the locale-sensitive way of formatting and parsing dates.

```
import java.awt.HeadlessException;
```

Figure 14 Class of Package java.awt

The HeadlessException exception is from the Package awt. This exception will be thrown if keyboard, display or mouse dependent code is execute din a setting that does not support any of those devices.

3.2 Variables

```
//get input from Text field
String getname = txtName.getText();
String getUsername = txtUserName.getText();
String getic = txtIC.getText();
String getmobile = txtMobile.getText();
String getemail = txtEmail.getText();
String getpwd = txtPwd.getText();
String getConfirmPwd = txtConfirmPwd.getText();

int confirmClose = JOptionPane.showConfirmDialog
```

Figure 15 Variables

Variables is a data container for storing data values. Variables in Java are classified into many categories such as String, int and etc.

3.3 Control Structures

3.3.1 if Statement

```
if (dataValid) {

    //clean the form
    clean();

    //check username availability
    Boolean accountExist = ValidateUser.ExistingAdmin(user);
```

Figure 16 if Statement

If a specific condition is met, use if to execute a block of code. For example, if the condition of "dataValid" is true, it will run the clean function.

3.3.2 if-else Statement

```
if (!accountExist) {

    //store user
    StoreData.AdminRegistration(user, status:"
    JOptionPane.showMessageDialog( parentComponen

    //if anything happen
    clean();

    //go to home page, wait for account to be
    setVisible( b:false);
    new Home().setVisible( b:true);

} else {
    JOptionPane.showMessageDialog( parentComponen

    //go to login page
    setVisible( b:false);
    new AdminLogin().setVisible( b:true);
```

Figure 17 if-else Statement

If a certain condition is met, the if-else expression will execute a block of code. For example, if the condition of "accountExist" is not true, it will run else statement.

3.3.3 Nested if-else Statement

```
if (dataValid) {

    //clean the form
    clean();

    //check username availability
    Boolean accountExist = ValidateUser.ExistingAdmin(user);
    try {
        if (!accountExist) {
```

Figure 18 Nested if-else Statement

A nested if statement in Java is a set of if conditions that are nested inside each other. For example, if the condition of "dataValid" is true, it will run another if statement inside the code.

3.4 Looping Structure

3.5.1 for Loop

```
for (String eachUser : userInfo) {

    //split the user info
    String[] userData = eachUser.split( regex: "//");

    //Identify customer
    if (ic.equals(userData[3]) && userData[0].equals( anObject: "CUSTOMER")) {
        txtName.setText(userData[1]);
        txtUserName.setText(userData[2]);
        txtIC.setText(userData[3]);
        txtMobile.setText(userData[4]);
        txtEmail.setText(userData[5]);
    }
```

Figure 19 for Loop

The Java for loop is used to cycle a block of code repeatedly. For example, the for loop will read the data value inside the “userInfo” until the end.

3.5.2 Nested for Loop

```
//loop to get each booking from booking info arraylist
for (String eachBooking : bookingInfo) {

    //split the booking info
    String[] bookingData = eachBooking.split( regex: "//");

    //store booking info of that room
    if (id.equalsIgnoreCase(bookingData[0])) {

        if (bookingData[num].equalsIgnoreCase( anotherString: change)) {
            JOptionPane.showMessageDialog( parentComponent: null, "Booking

        } else {
            bookingData[num] = change;
            JOptionPane.showMessageDialog( parentComponent: null,  message: "

            StringBuilder bookingBd = new StringBuilder();
            for (int i = 0; i < bookingData.length; i++) {
```

Figure 20 Nested for Loop

The nested loop is used in Java to construct patterns such as full pyramids, partial pyramids, inverted pyramids, and so on.

3.5.3 While Loop

```
while ((line = reader.nextLine()) != null) {
    lineArr = line.split( regex: "//");

    //convert string to date
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern( pattern: "dd-MM-yyyy");
    String strDate = lineArr[11];
    LocalDate toDate = LocalDate.parse( text: strDate, formatter);
    System.out.println("todate" + toDate);
```

Figure 21 While Loop

The while loop is used to run over code until the specified Boolean condition is true.

3.6 Object-Oriented Programming Concept (OOPs Concept)

Object-oriented programming is the basis of Java programming, which is used to develop programmes using classes and objects. Another method to describe OOPs is data control for code access. With this approach, programmers define both the operations to be carried out on a data structure and its data type (Great Learning Team, 2022).

By strategically constructing Java programmes, OOps in Java aims to improve code readability and reusability. The core ideas of object-oriented programming are abstraction, encapsulation, inheritance, and polymorphism. These concepts aim to include actual objects in programming (Great Learning Team, 2022).

3.6.1 Object

```
Booking book = new Booking();
book.setIc( ic: passLbl.getText());
book.setName( name: txtName.getText());
book.setEmail( email: txtEmail.getText());
book.setContact( contact: txtPhone.getText());
book.setNationality( nationality: txtNationality.getText());
book.setNumberPlate( numberPlate: passNumPlateLbl.getText());
book.setCarBrand( carBrand: passBrandLbl.getText());
book.setModel( model: passModelLbl.getText());
book.setPickupLocation( pickupLocation: setLocationLbl.getText());
book.setReturnLocation( returnLocation: returnLocationComboBox.getSelectedItem().toString());
book.setPickUpDate( pickUpDate: passPickDtLbl.getText());
book.setReturnDate( returnDate: passReturnDtLbl.getText());
book.setPickUpTime( pickUpTime: passPickTimeLbl.getText());
book.setReturnTime( returnTime: passReturnTimeLbl.getText());
```

Figure 22 CustomerBookingPage.java

In Java, an object is created using a class. Since the Booking class has been created, the object “book” is also created with the Booking class. As shown in the image above, “book” object is used to set the value of booking fields using the set method created in the Booking class.

3.6.2 Class

```
public class Car {

    private String numberPlate;
    private String brand;
    private String model;
    private String rating;
    private String location;
    private String dailyRate;
    private String transmission;
    private String numberOfPax;
    private String status;

    public String getNumberPlate() {...3 lines }

    public void setNumberPlate(String numberPlate) {...3 lines }

    public String getBrand() {...3 lines }

    public void setBrand(String brand) {...3 lines }

    public String getModel() {...3 lines }

    public void setModel(String model) {...3 lines }
```

Figure 23 Car.java

This Car class is created with all the private variable defined and get set methods created. With this Car class, the object can be created so the car values can be get and set.

```
public class ValidateUser {

    public static Boolean ExistingCustomer(User user) {...49 lines }

    public static Boolean ExistingAdmin(User user) {...39 lines }

    public static Boolean CustomerLoginCredential(User user) {...50 lines }

    public static Boolean AdminLoginCredential(User user) {...58 lines }

    public static Boolean ExistingUsername(User user) {...47 lines }

    public static String[] SearchUser(String searchValue, String userType) throws FileNotFoundException {...47 lines }

}
```

Figure 24 ValidateUser.java

This ValidateUser class is created to validate existing users, authenticate the login credentials, check if username is already taken and search users in the database. This class contains methods to be called and return values. As shown in the figure, this ValidateUser class can return Boolean value and String array value according to the called method.

3.6.3 Encapsulation

One of the four fundamental OOP concepts is encapsulation. Encapsulation is a Java approach for integrating the code that manages variables and methods into a single piece. A class's variables are kept private from other classes using encapsulation. Additionally, only the methods specific to the class may access the class's variables. Encapsulation is hence sometimes known as "data hiding" (*Java - Encapsulation* 2022).

```
public class User {

    private String accountType;
    private String name;
    private String username;
    private String ic;
    private String phone;
    private String email;
    private String password;
    private String confirmPassword;
    private String status;

    public String getAccountType() {...3 lines }

    public void setAccountType(String accountType) {...3 lines }

    public String getName() {...3 lines }

    public void setName(String name) {...3 lines }

    public String getUsername() {...3 lines }

    public void setUsername(String username) {...3 lines }

    public String getIc() {...3 lines }

    public void setIc(String ic) {...3 lines }

    public String getPhone() {...3 lines }

    public void setPhone(String phone) {...3 lines }
```

Figure 25 User.java

The figure above shows how encapsulation can be achieved in Java. To achieve encapsulation, the private variables must be defined first. After private variables is defined, the public getters and public setters method is created in order to manipulate and retrieve the value of the class's private variables (*Java - Encapsulation* 2022). For example, getUsername() and setUsername () methods is created for the private String username.

```
User user = new User();
user.setUsername( username: getUsername);
user.setPassword( password: getPassword);
```

Figure 26 CustomerLogin.java

In this implemented code, the username of User class is accessed with the setUsername() method. This set method is used to change the value of the username variables.

```
public static void CustomerRegistration(User user) {
    try {
        FileWriter fw = new FileWriter( fileName: "user.txt",  append: true);
        BufferedWriter bw = new BufferedWriter( out: fw);
        PrintWriter pw;
        pw = new PrintWriter( out: bw);
        pw.println("CUSTOMER//" + user.getName() + "//" + user.getUsername() + "//" + user.getIc() + "//"
                + user.getPhone() + "//" + user.getEmail() + "//" + user.getPassword() + "//CUSTOMER");
        pw.close();
    } catch (IOException e) {
        JOptionPane.showMessageDialog( parentComponent: null,  message: e);
    }
}
```

Figure 27 StoreData.java

As for the getUsername() method, this get method is used to pass the username of the customer. This username of the User class can only be retrieved with the public getUsername() method of the User class. The figure above showed how the username is retrieved to store into the text file.

3.6.4 Generalization

"Generalization" is the process of transforming a subclass type into a superclass type while broadening the scope and making the subclass more generic. Alternative names for generalization include widening and up casting. Widening will be secure. This is due to the reason that the classes will be more general (*Generalization and specialization in Java* 2022).

```
public class Home extends javax.swing.JFrame {
```

Figure 28 Home.java

There is generalization as extension implemented in the system. As shown in the figure above, the Home class inherits all of the JFrame class's attributes and methods. The Home class may be extended with this "extends" keyword. Home class is the child class which is the subclass. The parent class, or superclass, is the JFrame class. All of the parent class's properties are included when a Home class object is created (*Oops: Generalization as extension and restriction using Java* 2018). After the JFrame class is extended, the Home class will be able to have a top-level window called Frame which has title and border (Oracle, 2020).

3.6.5 Constructor method

Constructor is similar with method in Java which are used everytime when the object of the class is created. There are three types of Java Constructors, which are No-Arg Constructors, Parameterized Constructor and Default Constructor (Programiz, n.d.).

No-Arg Constructor

```
public class SelectUserType extends javax.swing.JFrame {

    public SelectUserType() {
        initComponents();
    }
```

Figure 29 SelectUserType.java

Java constructor can come with parameter(s) or without parameter(s). No-Arg Constructor will be constructor that do not have any parameters. As shown in the figure above, the constructor SelectUserType() is created. This constructor dost not contain any parameter. As a result, SelectUserType() constructor is referred as a no-arg constructor. This constructor created is declared as public. With a public constructor, the objects can be created outside of the class (Programiz, n.d.).

Parameterized Constructor

```
public class Formatting {

    String pickupDate, returnDate, yesterday, returnCarDate, today;
    SimpleDateFormat formatter = new SimpleDateFormat( pattern: "dd-MM-yyyy");

    public Formatting(Booking book) {
        this.pickupDate = book.getPickUpDate();
        this.returnDate = book.getReturnDate();
        this.yesterday = CalcYtdDate();
        this.today = TdyDate();
    }

    public Formatting(Booking book, String returnCar) {
        this.pickupDate = book.getPickUpDate();
        this.returnDate = book.getReturnDate();
        this.returnCarDate = book.getReturnCarDate();
        this.today = TdyDate();
    }

    //calculate yesterday date
    private String CalcYtdDate() {...8 lines }

    //calculate yesterday date
    private String TdyDate() {...5 lines }

    //format string to date to do comparison
    public Date PickupDate() throws ParseException {...6 lines }

    public Date ReturnDate() throws ParseException {...6 lines }

    public Date YesterdayDate() throws ParseException {...6 lines }

    public Date ReturnCarDate() throws ParseException {...6 lines }

    public Date TodayDate() throws ParseException {...6 lines }
}
```

Figure 30 Formatting.java

Parameterized constructor is constructor that has parameter(s) (Programiz, n.d.). There are parameterized constructors for the Formatting class constructors. Formatting constructors has object or object and string. The passed parameters are helpful for obtaining formatted dates.

```
Formatting fmt = new Formatting(book);
Date pickupDate = fmt.PickupDate();
Date returnDate = fmt.ReturnDate();
Date yesterday = fmt.YesterdayDate();
Date today = fmt.TodayDate();
```

Figure 31 SearchCar.java

The object "fmt" is created using the Formatting class. The book object is passed into the constructor of Formatting as parameters. he formatted date of the pickup date, return date, yesterday's date, and today's date may be accessed with the help of this created "fmt" object. These dates are retrieved to check if the inputted pickup date and return date is valid or not.

```
Formatting fmt = new Formatting(book, returnCar: "returnCar");
Date pDate = fmt.PickupDate();
Date rDate = fmt.ReturnDate();
Date rCarDate = fmt.ReturnCarDate();
Date todayDate = fmt.TodayDate();
```

Figure 32 ReturnCar.java

The object "fmt" is created using the Formatting class. The book object and returnCar string is passed into the constructor of Formatting as parameters. The formatted dates of the pickup date, return date, return car date, and the current date may be accessed using the "fmt" object that was created. These dates are retrieved to do validation for the inputted return car date.

Default Constructor

```
public class Booking {

    private String bookID;
    private String ic;
    private String name;
    private String email;
    private String contact;
    private String nationality;
    private String numberPlate;
    private String carBrand;
    private String model;
    private String pickupLocation;
    private String returnLocation;
    private String pickUpDate;
    private String returnDate;
    private String returnCarDate;
    private String pickUpTime;
    private String returnTime;
    private String rentpurpose;
    private String remarks;
    private String paymentMethod;
    private String paymentStatus;
    private String status;
    private String rentalFee;
    private String payUponCollection;
    private String payOnline;
    private String fine;
    private String avgRating;
    private String comments;

    public String getBookID() {...3 lines }

    public void setBookID(String bookID) {...3 lines }

    public String getIc() {...3 lines }

    public void setIc(String ic) {...3 lines }
```

Figure 33 Booking.java

```
Booking book = new Booking();
```

Figure 34 CustomerBookingPage.java

When the program runs, the Java compiler automatically creates a no-arg constructor if there is no constructor created beforehand. In this case, this constructor is referred as default

constructor. The figure above shows the default constructor of the Booking class as there are no constructor created in the Booking class. The default constructors' default values are used to initialise each variable (Programiz, n.d.).

Constructor Overloading

```
public class SearchCar extends javax.swing.JFrame {

    public SearchCar() throws FileNotFoundException {
        initComponents();

        //hide table
        carScrollPane.setVisible( aFlag: false);
        carTbl.setVisible( aFlag: false);
    }

    public SearchCar(User user, String type) throws FileNotFoundException {
        initComponents();

        //hide table
        carScrollPane.setVisible( aFlag: false);
        carTbl.setVisible( aFlag: false);
        passTypeLbl.setText( text: type);
        if (type.equalsIgnoreCase( anotherString: "Admin")) {

            //get  and hide acc status
            passLbl.setText( text: user.getStatus());
            passLbl.setVisible( aFlag: false);

            backBtn.setVisible( aFlag: false);
        } else if (type.equalsIgnoreCase( anotherString: "Customer")) {
            //get  and hide ic
            passLbl.setText( text: user.getIc());
            passLbl.setVisible( aFlag: false);

            adminBackBtn.setVisible( aFlag: false);
        }

    }
```

Figure 35 SearchCar.java

As shown in the figure above, the SearchCar class has two constructors which are one constructor with no parameter and one constructor with object and string as parameters. These different constructors are called based on the parameters provided during object creation (Programiz, n.d.). If there is an object and string in the parameter, the second constructor will be called. The car table will be hidden, and the type of user is passed using the String, type. The text of the label will be set and button is hide based on the user type given.

3.6.6 Get and Set method

To further secure the code and protect data, getter and setter methods are used. Getters which are also known as accessors, will return values of various datatypes. Getter begins with the word “get” and followed by the variable name (GeeksforGeeks, 2022).

When it comes to setters, sometimes referred to as mutators, they are used to set or update the value. This changed the variable's value for the class. Setters have the word "set" at the beginning, then the variable name. Setting and getting values for a particular data type is made easier for programmers by the usage of getter and setter methods. For both the method of getter and setter, the variable's first letter must be capital (GeeksforGeeks, 2022).

```
public class Car {

    private String numberPlate;
    private String brand;
    private String model;
    private String rating;
    private String location;
    private String dailyRate;
    private String transmission;
    private String numberOfPax;
    private String status;

    public String getNumberPlate() {...3 lines }

    public void setNumberPlate(String numberPlate) {...3 lines }

    public String getBrand() {...3 lines }

    public void setBrand(String brand) {...3 lines }

    public String getModel() {...3 lines }

    public void setModel(String model) {...3 lines }

    public String getRating() {...3 lines }

    public void setRating(String rating) {...3 lines }

    public String getLocation() {...3 lines }

    public void setLocation(String location) {...3 lines }

    public String getDailyRate() {...3 lines }

    public void setDailyRate(String dailyRate) {...3 lines }
```

Figure 36 Car.java

As shown in the figure above, this Car class contains private variables defined in the class. Other than private variables, there are also getters created for all variables. Followed by the getters, all the setters are also created for all variables.

Setters

```
Car car = new Car();
car.setNumberPlate( numberPlate: getNumberPlate);
car.setBrand( brand: getBrand);
car.setModel( model: getModel);
car.setLocation( location: getLocation);
car.setTransmission( transmission: getTransmission);
car.setNumberOfPax( numberOfPax: getNumberOfPax);
car.setDailyRate( dailyRate: getDailyRate);
```

Figure 37 AdminManageCar.java

As shown in the figure above, all the input car details are updated to the variable in the Car class. The Car class's private variable may be modified by creating an object of the car car and utilising the set methods.

```
StoreData.CarRegistration(car);
```

Figure 38 AdminManageCar.java

The car object is then passed into the method of CarRegistration as parameter.

```
public static void CarRegistration(Car car) {
    try {
        FileWriter fw = new FileWriter( fileName: "car.txt",  append: true);
        BufferedWriter bw = new BufferedWriter( out: fw);
        PrintWriter pw;
        pw = new PrintWriter( out: bw);
        pw.println(car.getNumberPlate() + "//" + car.getBrand() + "//" + car.getModel() + "//-//" + car.getLocation() +
                "//" + car.getDailyRate() + "//" + car.getTransmission() + "//" + car.getNumberOfPax() + "//AVAILABLE");
        pw.close();
        JOptionPane.showMessageDialog( parentComponent: null, "Congratulations! " + car.getNumberPlate() + " is added into the car database.",
                title: "Car Added Successfully!",  messageType: JOptionPane.INFORMATION_MESSAGE);
    } catch (IOException e) {
        JOptionPane.showMessageDialog( parentComponent: null,  message: e);
    }
}
```

Figure 39 StoreData.java

When the car object is passed to this method, this method use the car object to get the variables of Car class. The private variables in the Car class is already updated previously using the setters. Using the getters, the updated value of car fields can be retrieved and stored into the database.

3.6.7 Normal method

```
private void retrieveCarTableData(String[] carInfo) throws FileNotFoundException {

    DefaultTableModel model = (DefaultTableModel) carTbl.getModel();

    //loop to get each car from car info arraylist
    for (String eachCar : carInfo) {
        String[] carData = eachCar.split( regex: "//"); //split the booking info
        String[] tableRow = {carData[0], carData[1], carData[2], carData[3],
            carData[4], carData[6], carData[7], carData[5], carData[8]};
        model.addRow( rowData: tableRow); //add to table
    }
}
```

Figure 40 SearchCar.java

This retrieveCarTableData method is created to get the data of the table row and add the rows into the car table. This method accepts string array parameter that contains all car information of each car. In this method, there are a DefaultTableModel default constructor. The TableModel that provides the data displayed by the Car table is obtained using the getModel method. With the for loop and split function, each field of the car is extracted. The addRow method helps to add the row of car data at the end of the car table (Oracle, 2022).

```
retrieveCarTableData( carInfo: availableCarDetails);
```

Figure 41 SearchCar.java

When the user enters valid pickup location, pickup date, pickup time, return date, and return time, the retrieveCarTableData function is called to display all the information for each car that are currently listed in the car table.

3.6.8 Exception Handling

Java's exception handling is a useful technique for managing runtime errors and sustaining the normal flow of the programme. The key advantage of exception handling is that it keeps the application's normal flow maintained. Exceptions need to be managed since the exceptions often disrupt the system's regular flow. There are a few keywords used to manage exceptions which are try, catch, multicatch, finally, throw, and throws (JavaTpoint, 2021). There are a few keywords implemented in this car rental system.

try-catch block

Java use try blocks to handle code that could cause an exception. The method needs to use a try block. If an exception occurs at a particular statement, the other statements in the try block won't execute. Therefore, it is advised not to keep code in a try block that won't throw an exception (JavaTpoint, 2021).

A finally block or a catch block must come after a Java try block. The Java catch block is used to handle the exception by stating the exception type within the parameter. The ideal approach is to declare the generated kind of exception. A catch block and can only be placed after the try block. Multiples catch blocks may be placed under a try block (JavaTpoint, 2021).

```
try {

    String[] newRecord = ModifyUserData.LogoutRecord();
    UpdateData.UpdateFile( modifiedList: newRecord,  fileName: "LoginRecord.txt");

    setVisible( b: false);
    new Home().setVisible( b: true);

} catch (FileNotFoundException ex) {
    JOptionPane.showMessageDialog( parentComponent: this,  message: ex);
}
```

Figure 42 SearchCar.java

As shown in the figure above, the try block surrounds the code. Try blocks are followed by catch blocks that capture exceptions of the FileNotFoundException class.. This exception is for

the method that opens the LoginRecord.txt file. If this file is unreachable, the FileNotFoundException will be thrown, and the remaining statements will not be run.

Multi-catch block

```
} catch (NumberFormatException | ParseException | HeadlessException | FileNotFoundException ex) {
    JOptionPane.showMessageDialog( parentComponent: this, message: "Please select a Return car date", title: "Warning!", messageType: JOptionPane.WARNING_MESSAGE);
}
```

Figure 43 ReturnCar.java

By using Multi-catch block, all exceptions can be handled under only one catch block. The exceptions are separate by "|" symbol as shown in the figure above (javaTpoint, 2021). NumberFormatException , ParseException, HeadlessException and FileNotFoundException are included in one catch block. If any of the exceptions is thrown, the error message will be shown.

throws Exception

```
public static Integer[] RentalSummary() throws FileNotFoundException {...38 lines }
public static Integer[] CarSummary() throws FileNotFoundException {...75 lines }
public static Double[] Sales() throws FileNotFoundException, ParseException, NumberFormatException {...54 lines }
```

Figure 44 GenerateReport.java

Declaring an exception in Java uses the throws keyword. Using this keyword will help to inform that an exception could happen in the method. Therefore, there are a few exception is thrown using the "throws" keyword.

3.6.9 Files and Input/Output

```
import java.io.*;
```

Figure 45 StoreData.java

The java.io package contains almost all of the classes that are possible required to perform input and output (I/O) in Java. These streams are an indication of an input source and an output destination. The stream in the java.io package is capable of handling several forms of data, including primitives, objects, localized characters, and others (Tutorials Point, n.d.).

Add A Row

```
public static void CustomerRegistration(User user) {
    try {
        FileWriter fw = new FileWriter( fileName: "user.txt",  append: true);
        BufferedWriter bw = new BufferedWriter( out: fw);
        PrintWriter pw;
        pw = new PrintWriter( out: bw);
        pw.println("CUSTOMER//" + user.getName() + "//" + user.getUsername() + "//" + user.getIc() + "//"
                + user.getPhone() + "//" + user.getEmail() + "//" + user.getPassword() + "//CUSTOMER");
        pw.close();
    } catch (IOException e) {
        JOptionPane.showMessageDialog( parentComponent: null,  message: e);
    }
}
```

Figure 46 StoreData.java

The code above shows how a row is added into the database with the help of FileWriter, BufferedWriter and PrintWriter classes. To open a file, the FileWriter object with a String of filename and Boolean value of append given in the constructor. As shown in the figure above, the filename given is "user.txt" which is the user database, and the Boolean value is true for append which means the file will be append with the new row of data. The output of the file will be received from the FileWriter object "fw" using the BufferedWriter class, which is used to generate a buffered character output buffer. The BufferedWriter class enables the efficient writing of single characters, arrays, and strings by buffering characters as the data in user text file are written to a character output stream. After the file output is gotten, a new PrintWriter is created. The OutputStreamWriter is also created with the constructor, which converts the file data into bytes using default character encoding. The IOException is thrown to handle the

exceptions whenever the file is unreachable. The row of customer data is printed into the text file using the println() method. Lastly, the close() method will close the stream (Oracle, 2022).

Update A Row of Data

```
public static String[] UpdateCarInfo(Car car) throws FileNotFoundException {

    //get number plate
    String carNumberPlate = car.getNumberPlate().toLowerCase().strip();

    // modifiedlist to store all updated cars
    ArrayList<String> updatedCarInfo
            = new ArrayList<>();

    String[] carInfo = RetrieveData.RetrieveData( fileName: "car.txt");

    //loop to get each car from arraylist
    for (String eachCar : carInfo) {

        //split the car info
        String[] carData = eachCar.split( regex: "//");

        //store car info
        if (carNumberPlate.equals( anObject: carData[0].toLowerCase().strip())) {
            carData[1] = car.getBrand();
            carData[2] = car.getModel();
            carData[4] = car.getLocation();
            carData[6] = car.getTransmission();
            carData[7] = car.getNumberOfPax();
            carData[5] = car.getDailyRate();
```

```
            StringBuilder carBd = new StringBuilder();
            for (int i = 0; i < carData.length; i++) {

                carBd.append(carData[i]);

                //last element no need seperator
                if (i == (carData.length - 1)) {
                    break;
                }
                carBd.append( str: "//");
            }

            String carDetails = carBd.toString();
            updatedCarInfo.add( e: carDetails);
        }

        updatedCarInfo.add( e: eachCar);

    }

    // convert car List to string array
    String[] modifiedCars
            = updatedCarInfo.toArray(String[]::new);

    return modifiedCars;
}
```

Figure 47 ModifyCarData.java

To update a row of data, the first thing to do will be creating a new string array which contains all the updated data of the text files. As shown above, the UpdateCarInfo method is used to return a string array of updated car data of the car text file. This method accepts the car object as parameter. The car object is used to retrieved all the updated car data from the Car class. An arraylist of updatedCarInfo is created advanced. A string array of carInfo is retrieved with the help of RetrieveData method. The RetrieveData function will assist in retrieving every piece of information from the text file and converting it to a string array. Using the given car number plate, the field of the car is changed if the car number plate matches the car number plate in the retrieved data. After the data of each car fields is changed, the fields will be appended together to form a row of car information. This can be done with the StringBuilder class and the append method. The arraylist which contains all the modified car details will then convert into a string array and returned with the method.

```
public static void UpdateFile(String[] modifiedList, String fileName) {
    try {
        //write the file
        FileWriter fw = new FileWriter(fileName,  append: false);
        BufferedWriter bw = new BufferedWriter( out: fw);
        PrintWriter pw;
        pw = new PrintWriter( out: bw);
        for(String eachRow : modifiedList)
        {
            pw.println( x: eachRow);
        }
        pw.close();
    }catch(IOException e){
        JOptionPane.showMessageDialog( parentComponent: null,  message: e);
    }
}
```

Figure 48 UpdateData.java

This UpdateFile method is similar with the method to add a row as shown above. At the FileWriter class, an object is created with the filename and Boolean value of append given in the parameter. Different with the CustomerRegistration method above, the Boolean value here is set as false, which means the data of the file will be write. Using the string array of modifiedList passed from the UpdateFile method parameter, the foreach loop and println method is used to print each row of data into the file. By using this way, the old data in the file is erased and are replaced with the new data in the string array passed from the UpdateFile method.

```
String updatedCarlist[] = ModifyCarData.UpdateCarInfo(car);
UpdateData.UpdateData( modifiedList: updatedCarlist,  fileName: "car.txt");
```

Figure 49 AdminManageCar.java

These two methods is used to update an individual car information as shown above.

Delete a row

```
public static String[] DeleteCar(Car car) throws FileNotFoundException {

    //get number plate
    String carNumberPlate = car.getNumberPlate().toLowerCase().strip();

    // modifiedlist to store all updated cars
    ArrayList<String> updatedCarInfo
            = new ArrayList<>();

    String[] carInfo = RetrieveData.RetrieveData( fileName: "car.txt");

    //loop to get each booking from booking details arraylist
    for (String eachCar : carInfo) {

        //split the car info
        String[] carData = eachCar.split( regex: "//");

        //store booking info of that room
        if (carNumberPlate.equals(carData[0] = carData[0].toLowerCase().strip())) {

            //continue without saving info
        } else {

            updatedCarInfo.add( e: eachCar);
        }

    }

    // convert booking List to string array
    String[] modifiedCars
            = updatedCarInfo.toArray(String[]::new);

    return modifiedCars;
}
```

Figure 50 ModifyCarData.java

The implemented above shows how a car is deleted. To delete a car from text file, the car should be removed from the string array that is going to store into the text file. As shown in the figure above, if the car number plate given matches the car number plate in the text file, the car information will not be added into the arraylist. The arraylist will then convert into a string array. As a result, a string array that contains all the cars except the deleted car will be returned.

```
String updatedCarlist[] = ModifyCarData.DeleteCar(car);
UpdateData.UpdateData( modifiedList: updatedCarlist,  fileName: "car.txt");
```

Figure 51 AdminManageCar.java

This is how a car will be deleted by using this DeleteCar method and the UpdateData method which will write the car file and store the car information of the string array.

Retrieve All Data

```
public static String[] RetrieveData(String fileName) throws FileNotFoundException {

    // arraylist to store all data
    ArrayList<String> list;
    list = new ArrayList<>();

    //open file to read
    FileInputStream fis = new FileInputStream( name: fileName);

    //scan file
    Scanner s = new Scanner( source: fis);

    // retrieve all rows from file
    while (s.hasNextLine()) {

        // add each line into arraylist
        list.add( e: s.nextLine());
    }

    //convert arrayList to string array
    String[] array;
    array = list.toArray(String[]::new);

    return array;
}
```

Figure 52 RetrieveData.java

This method uses the FileInputStream class the read data from the given filename. The hasNextLine function is then used to see whether the file has another line, after an object is created using the Scanner class. The nextLine method helps to retrieved the row of data in the text file. By using a while loop, all the lines in the file is retrieved and stored into the string array. This string array will return all the lines in the given text file.

4.0 Additional Features

4.1 Array

```
String[] leftCol = {"Car Number Plate", "Car Brand", "Pick up Date", "Pick up Time",
    "Return Date", "Return Time", "Pay Online", "Pay Upon Collection", "Total Rental Fee"};
String[] rightCol = {bookingDetails[6],bookingDetails[7],bookingDetails[11],bookingDetails[14],
    bookingDetails[12],bookingDetails[15],"RM "+bookingDetails[23],"RM "+bookingDetails[22],"RM "+bookingDetails[21]};
```

Figure 53 Receipt.java

In the implemented Java code, array is used to done many operation. In the figure above, there are two string arrays is declared. In Java, an array is declared by first stating its datatype and then its name. In this case, the string of string array elements is included in the curly braces "{}", separated with comma ",". As for the second string array, the bookingDetails string array elements is accessed by using the index number. The index number is placed between the square brackets "[]".

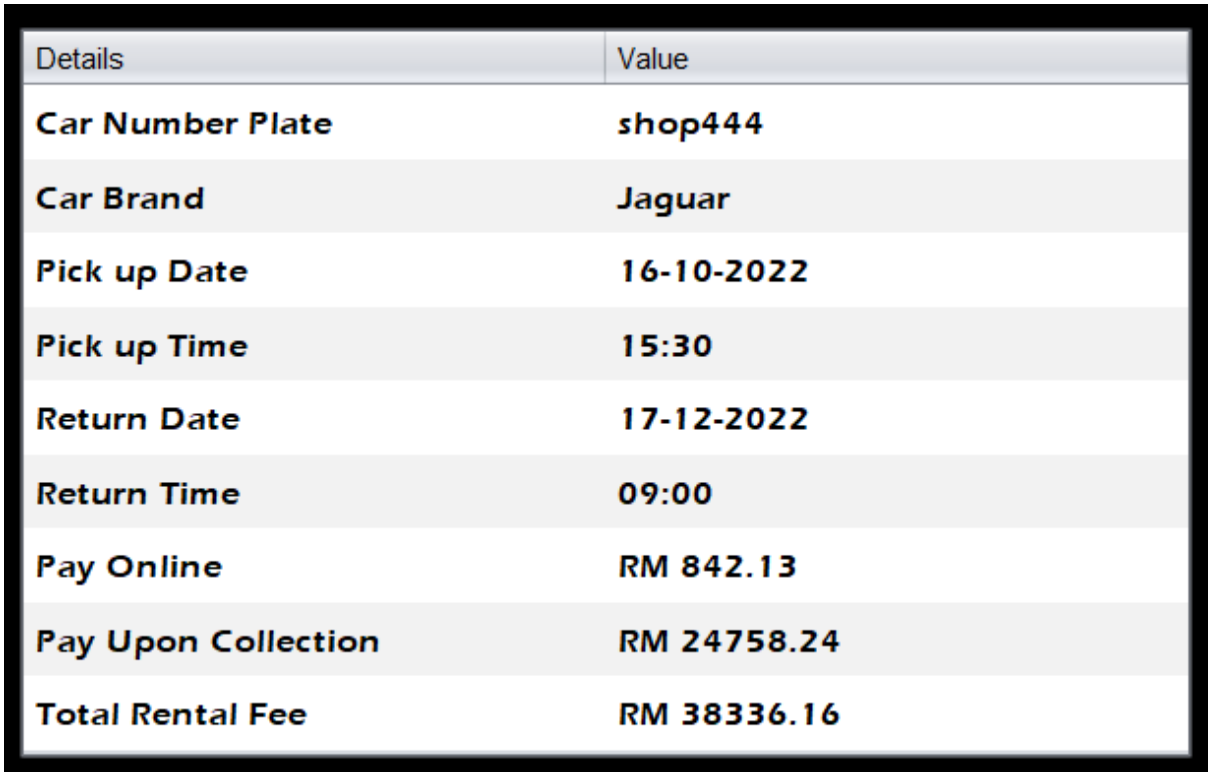
4.2 ArrayList

```
public static String[] Individual(String id, String fileName) throws FileNotFoundException, ParseException {

    String[] info = RetrieveData.RetrieveData(fileName);

    //create arraylist to store all details
    ArrayList<String> list
            = new ArrayList<>();

    //loop to get each record from arraylist
    for (String each : info) {

        //split info
        String[] details = each.split( regex: "//");

        if(id.equalsIgnoreCase(details[0])){

            //add all fields to the list individuall
            list.addAll( c: Arrays.asList( a: details));
        }

    }

    // convert list to string array
    String[] indvDetails
            = list.toArray(String[]::new);

    return indvDetails;

}
```

Figure 54 RetrieveData.java

The figure above shows how the arraylist is declared. The data type of the arraylist is stated between "<>". As shown in the figure above, the list arraylist datatype is String. By using the addAll method, all the elements of the list will be added into the list arraylist. The list arraylist is then converted into the string array with the toArray mehod and returned by the Individual method.

4.3 JTable



Details	Value
Car Number Plate	shop444
Car Brand	Jaguar
Pick up Date	16-10-2022
Pick up Time	15:30
Return Date	17-12-2022
Return Time	09:00
Pay Online	RM 842.13
Pay Upon Collection	RM 24758.24
Total Rental Fee	RM 38336.16

Figure 55 Receipt.java

As shown in the figure above, JTable helps to show the data in a table.

```
//table
DefaultTableModel model = (DefaultTableModel) rentalInfoTbl.getModel();

String[] leftCol = {"Car Number Plate", "Car Brand", "Pick up Date", "Pick up Time",
    "Return Date", "Return Time", "Pay Online", "Pay Upon Collection", "Total Rental Fee"};
String[] rightCol = {bookingDetails[6],bookingDetails[7],bookingDetails[11],bookingDetails[14],
    bookingDetails[12],bookingDetails[15],"RM "+bookingDetails[23],"RM "+bookingDetails[22],"RM "+bookingDetails[21]};

for (int i = 0; i < leftCol.length; i++) {
    String[] tableRow = {leftCol[i], rightCol[i]};
    model.addRow( rowData: tableRow); //add to table
}
```

Figure 56 Receipt.java

The data is displayed in the rental info table with the help of the DefaultTable class constructor and object. Then, the addRow method is used to add each row of data into the rental info table.

4.4 JCalendar

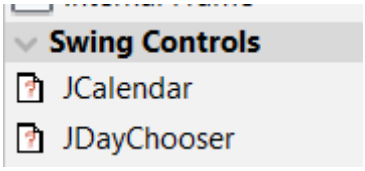


Figure 57 Swing Controls

The JCalendar is downloaded and installed into the NetBeans IDE to allow user to be able to select a date from the calendar GUI (toedter.com, n.d.).

Figure 58 SearchCar.java

This is an example of the JDateChooser, which allows user to select the day, month and year.

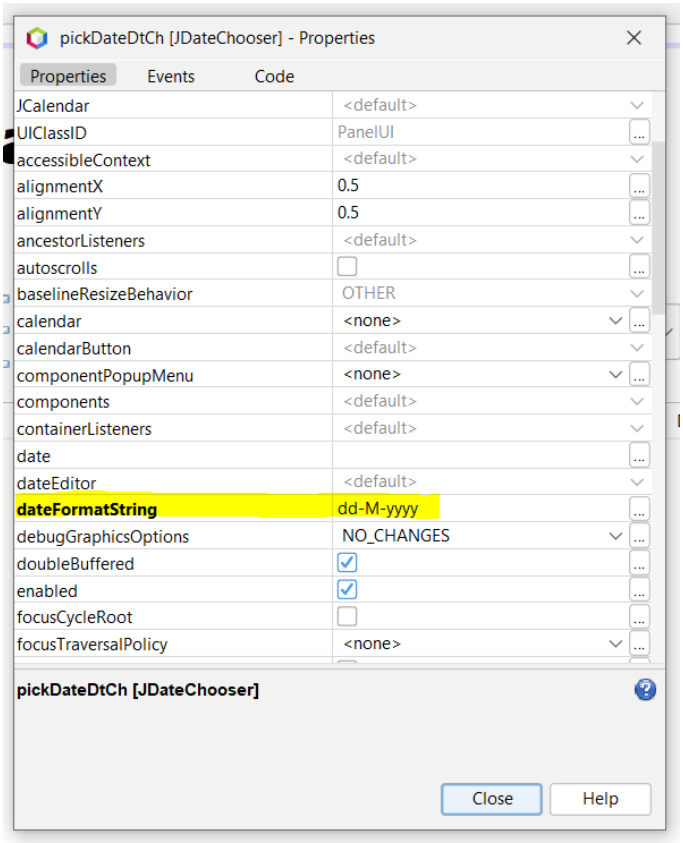


Figure 59 SearchCar.java

The format of the date can be set. As shown in the figure above, the format is set to “dd-M-yyyy” which returns date such as “08-11-2022”.

```
String getPickUpDate = formatter.format( date: pickDateDtCh.getDate());
String getPickUpTime = pickupTmCoBox.getSelectedItem().toString();
String getReturnDate = formatter.format( date: returnDateDtCh.getDate());
```

Figure 60 SearchCar.java

The date selected by user can be retrieved using the getDate method. The getDate method return in the Date datatype. The date can be format into String datatype with the help of the SimpleDateFormat class.

5.0 Sample Output Screens

5.1 Home

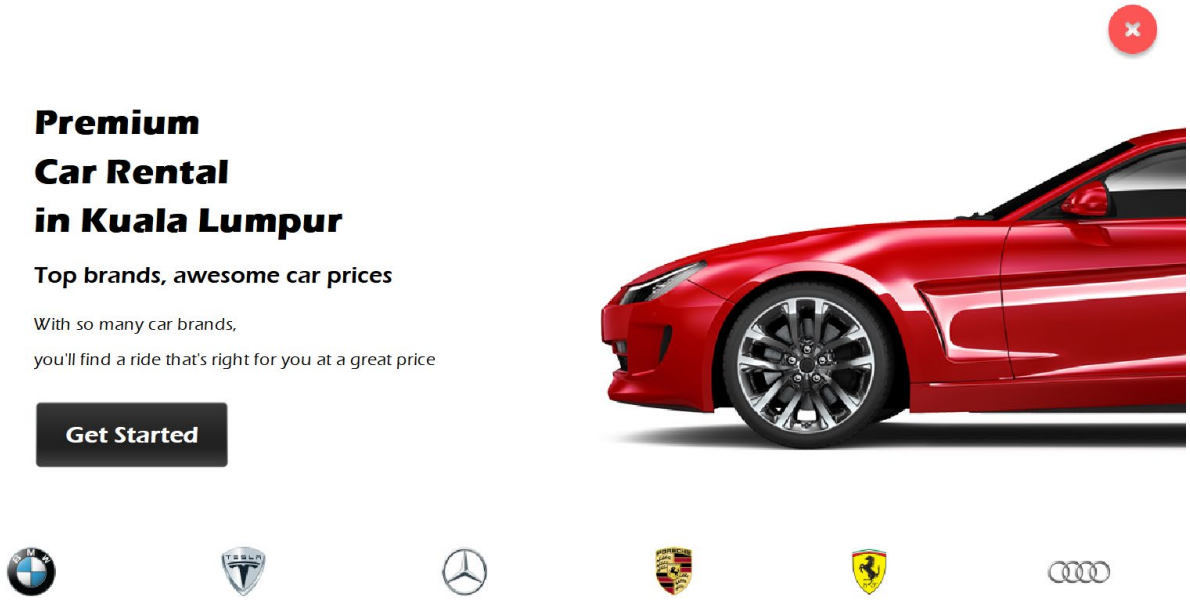


Figure 61 Home.java

This is the Welcome page for Premium Car Rental in Kuala Lumpur. This page displays some of the car brands in this system. The "Get Started" option allows users to begin enjoying the Premium Car Rental System.

5.2 Select User type

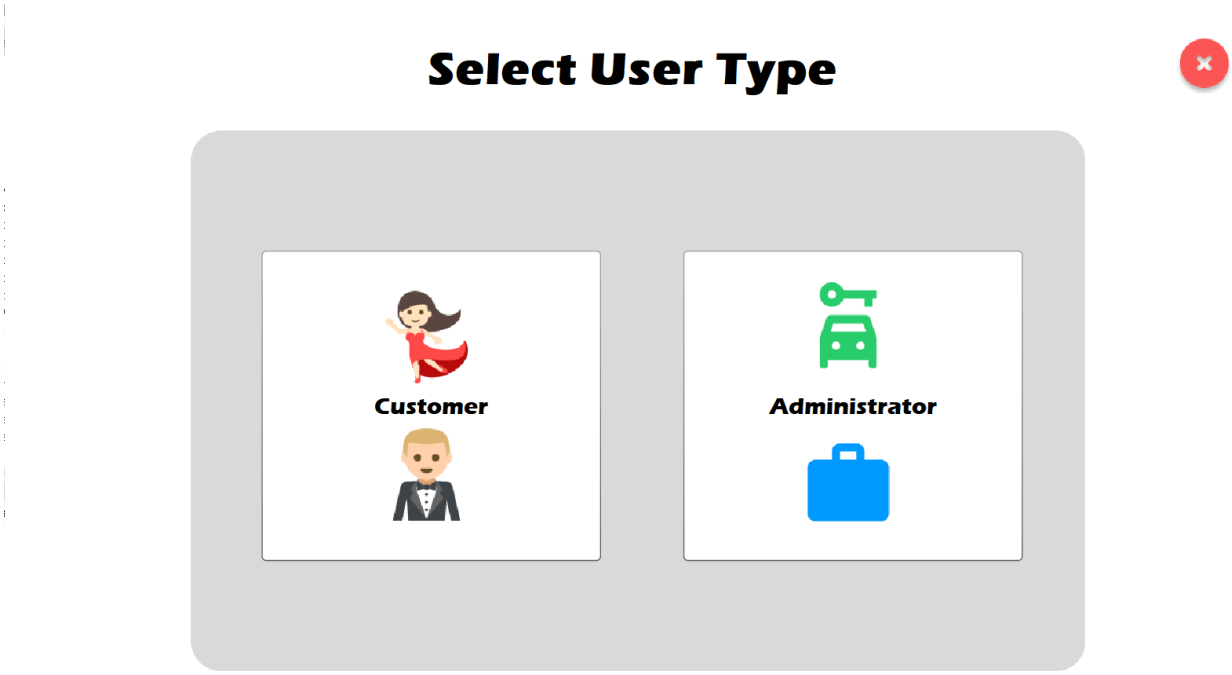


Figure 62 Select User type

This page allows users to choose between two user types such as customer and administrator. Customers may begin using the system by clicking the "Customer" button. Furthermore, the "Administrator" button is used by administrators to supervise the Premium Car Rental System.

5.3 Customer

5.3.1 Customer Login

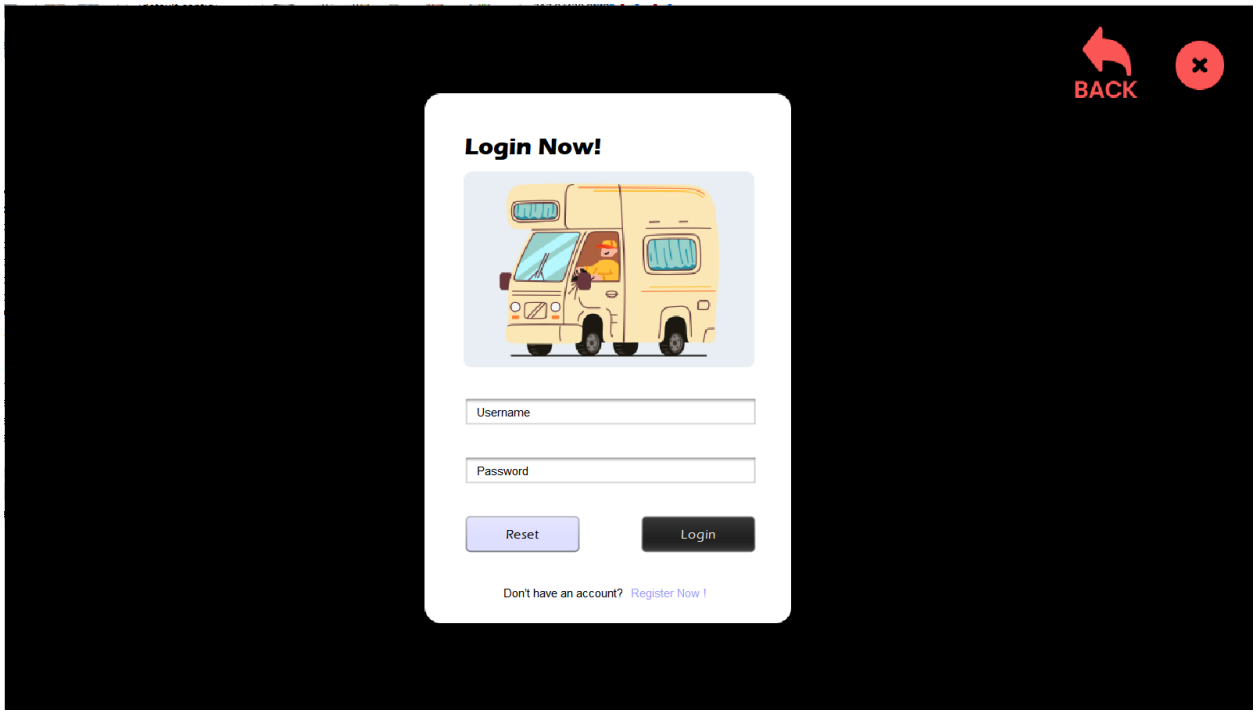


Figure 63 Customer Login

This is the Login page. Customers need to provide a valid login and password in order to access the system. The "BACK" button may be used to go back to the previous page. By clicking the "Reset" button, customers may erase their login details. To access the system and verify their username and password, customers must click the "Login" button. Customers will be sent to the "Home page" if the valid username and password are entered.

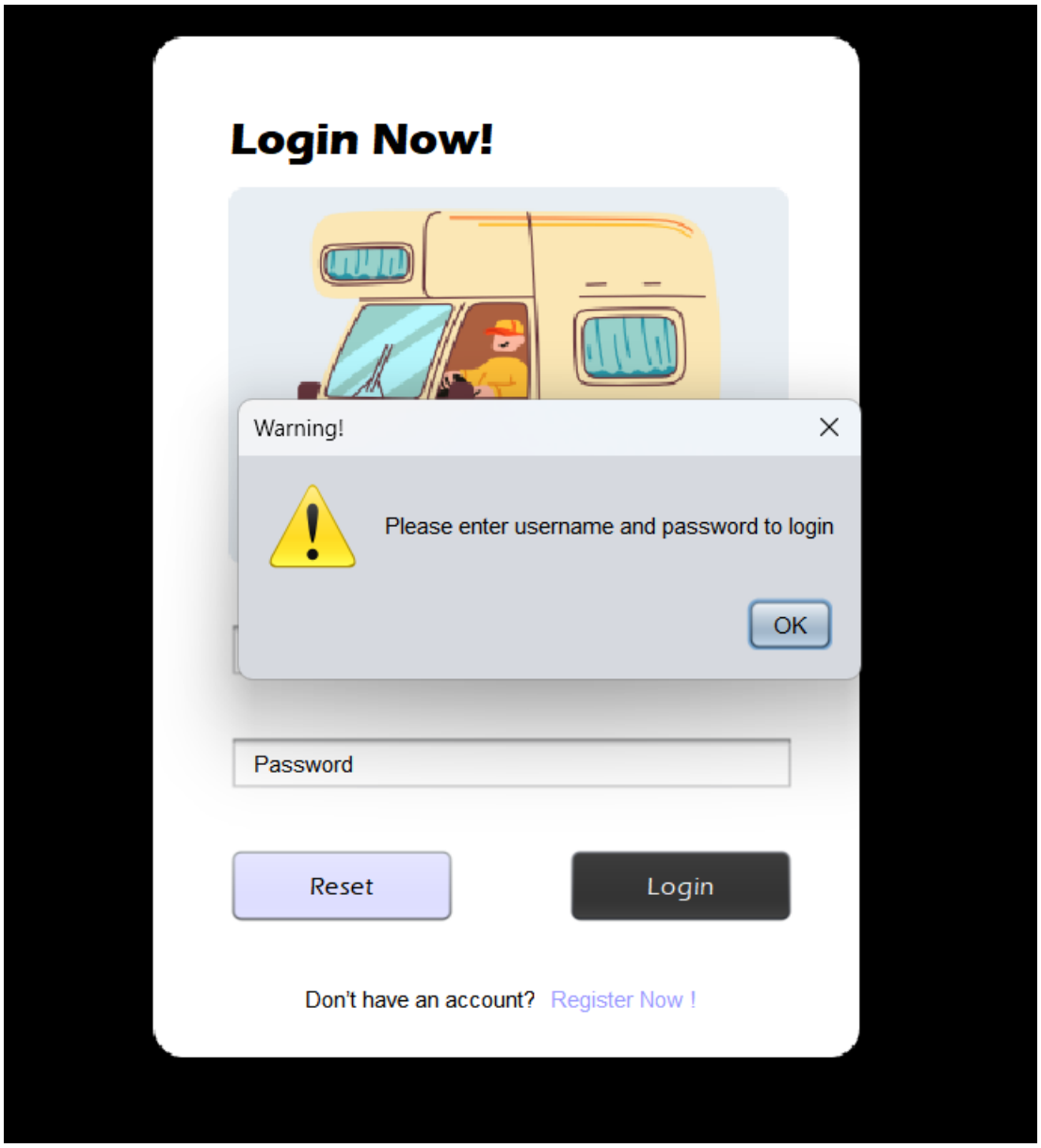


Figure 64 Customer Login

If the login information is incomplete, this notice will show and requesting that consumers complete all of the login information.

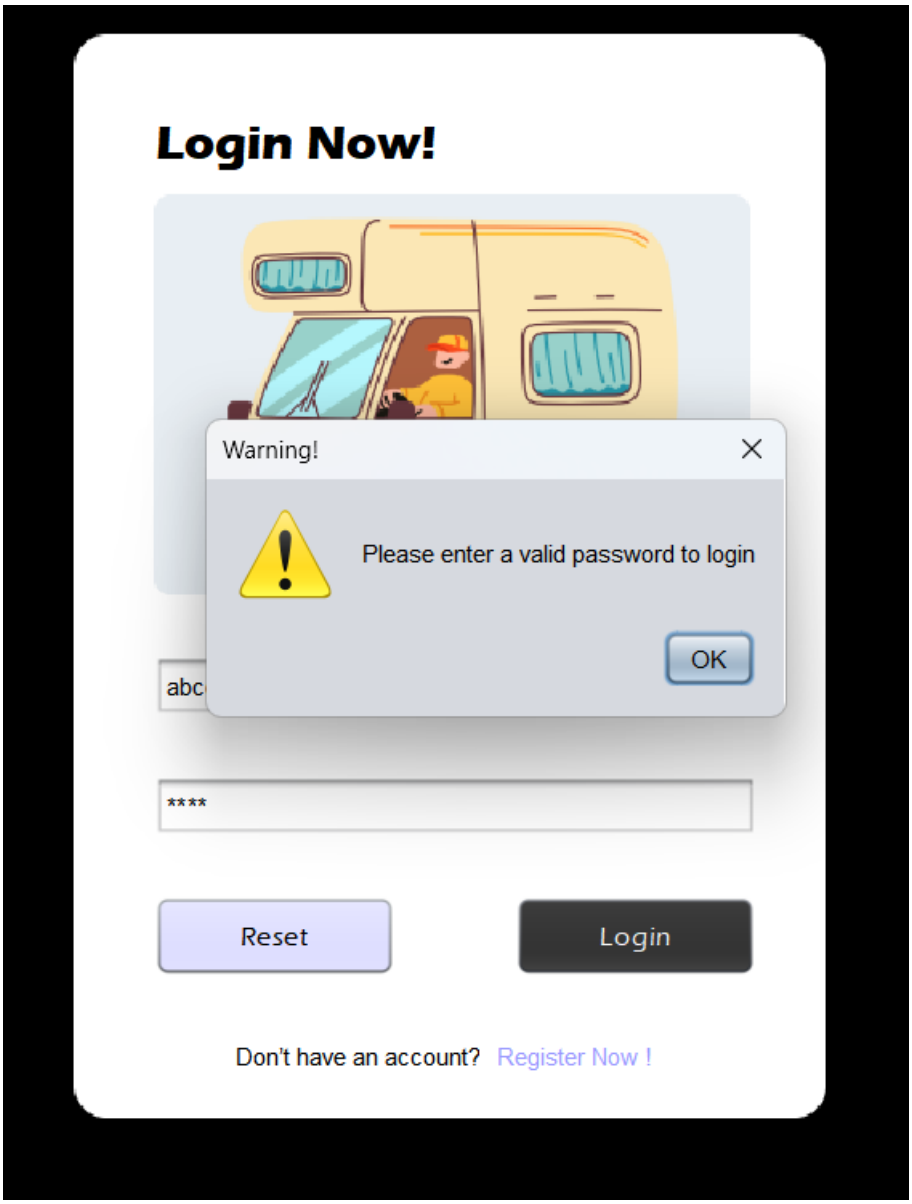


Figure 65 Customer Login

If customers forget to enter their password, this message will prompt them to do so.

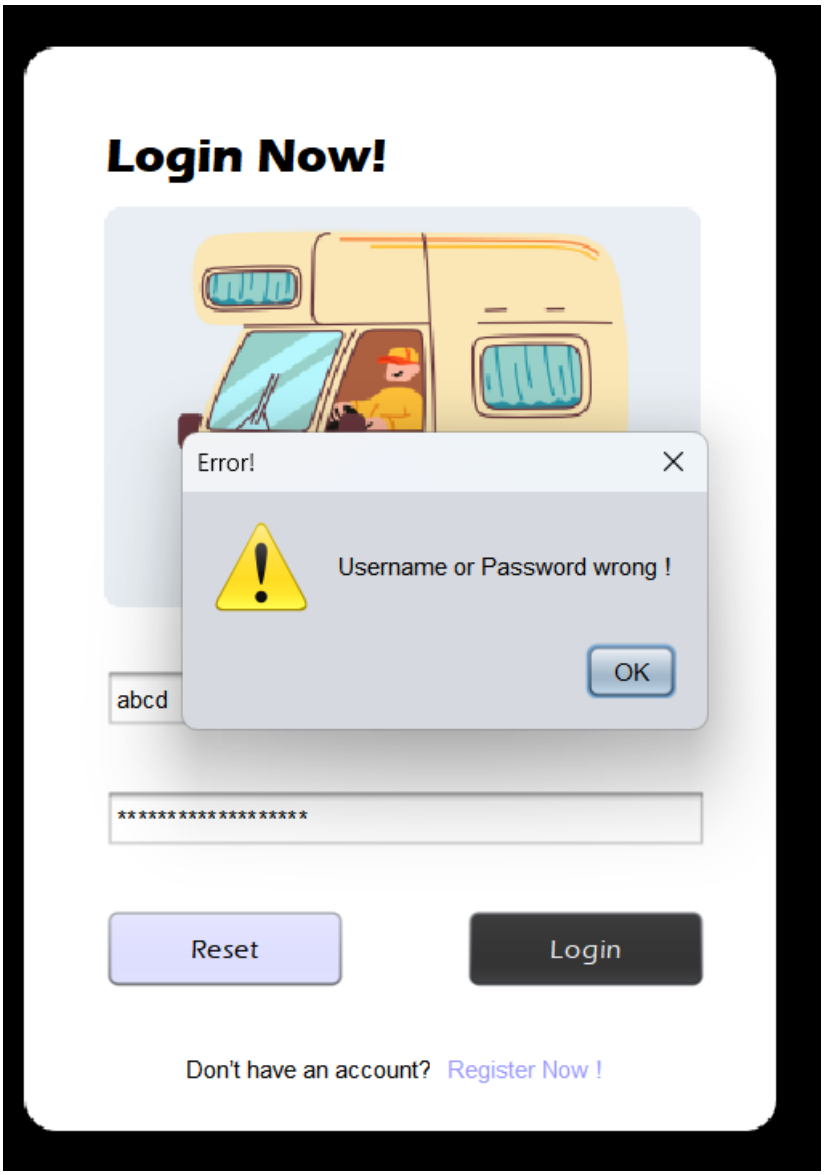


Figure 66 Customer Login

This message will notify the user and prompt them to input their username and password again if the entered username or password does not match.

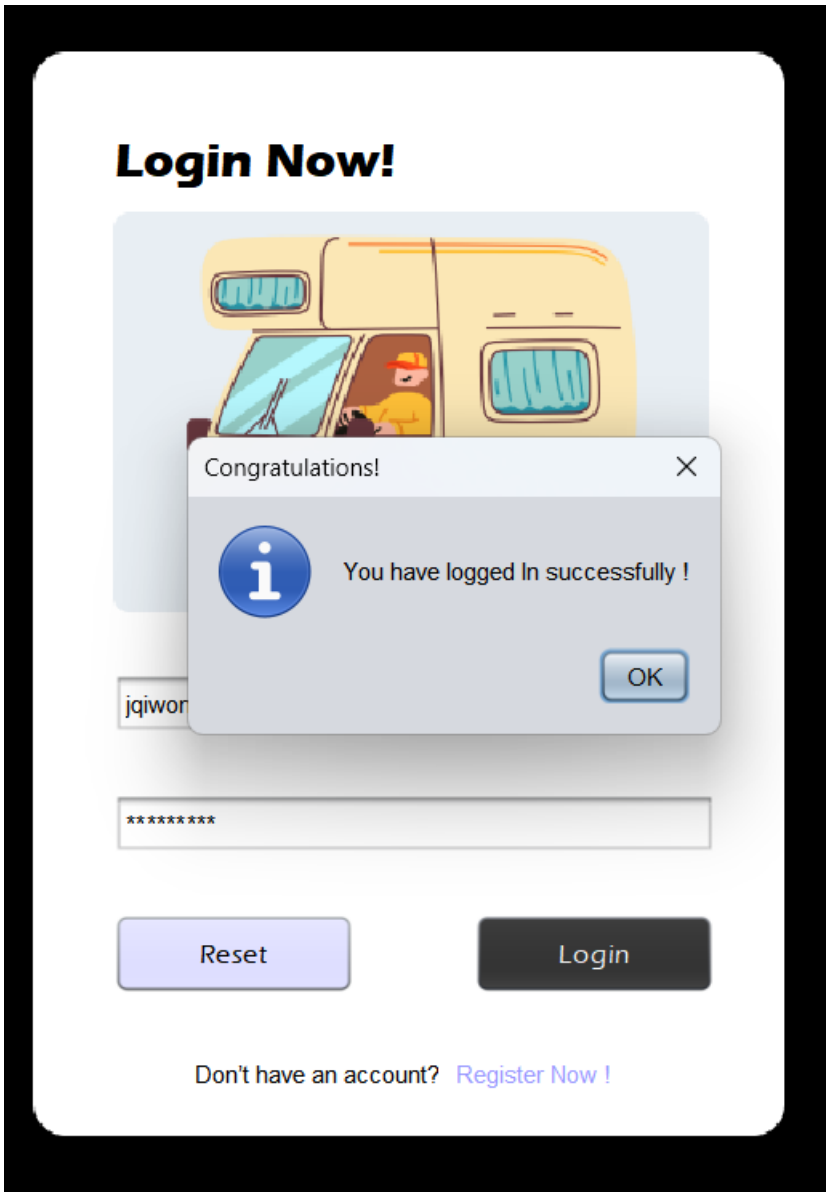


Figure 67 Customer Login

This message will appear if customers login or password is valid, click "OK" to proceed to the “Customer Menu” page.

5.3.2 Customer Registration

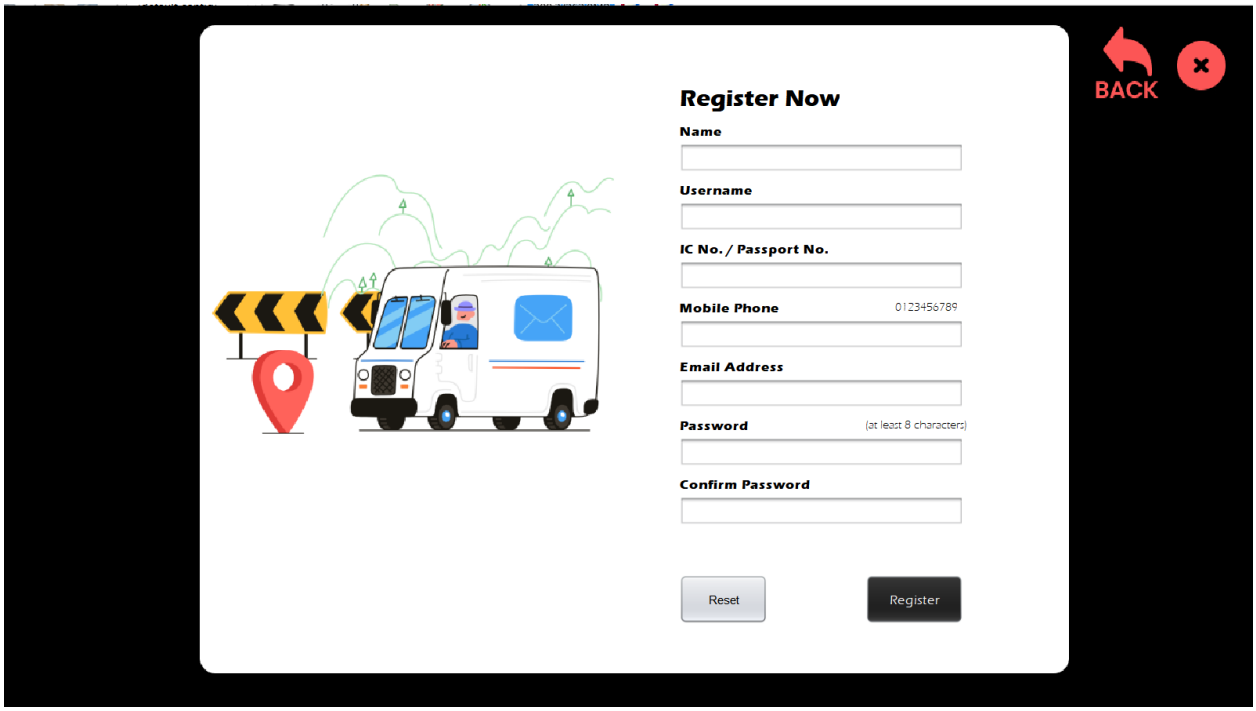


Figure 68 Customer Registration

This is the Registration page, where customers may create a new account by entering their personal information. Customers will be able to reset everything in the form by pressing the "Reset" button. Customers may return to the "Login page" by pressing the "Back" button. After entering all their personal information, customers must click the "Register" button to authenticate their information, which is saved in the user.txt file.

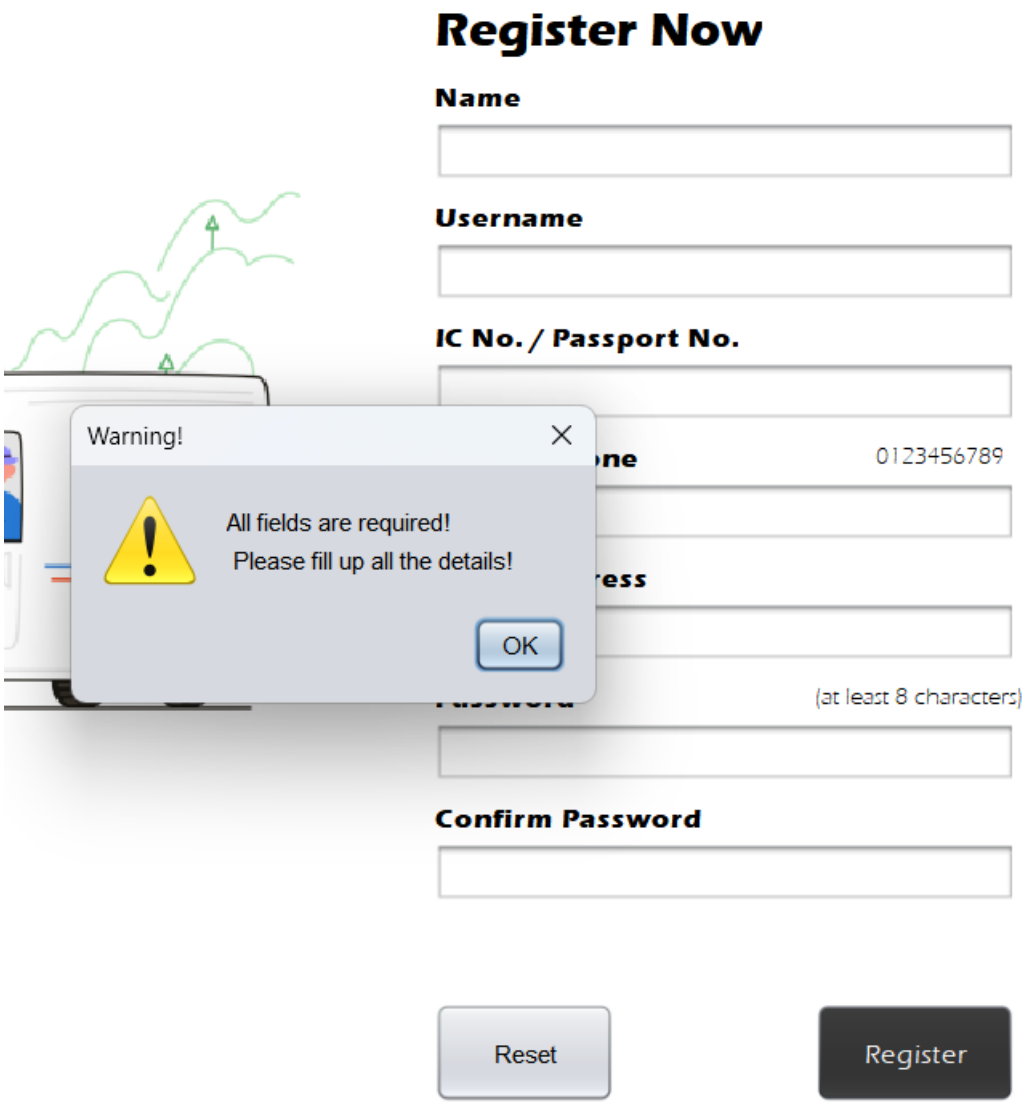


Figure 69 Customer Registration

This warning notice will show if the personal information on the Registration form is incomplete. This notification instructs customers to enter all information.

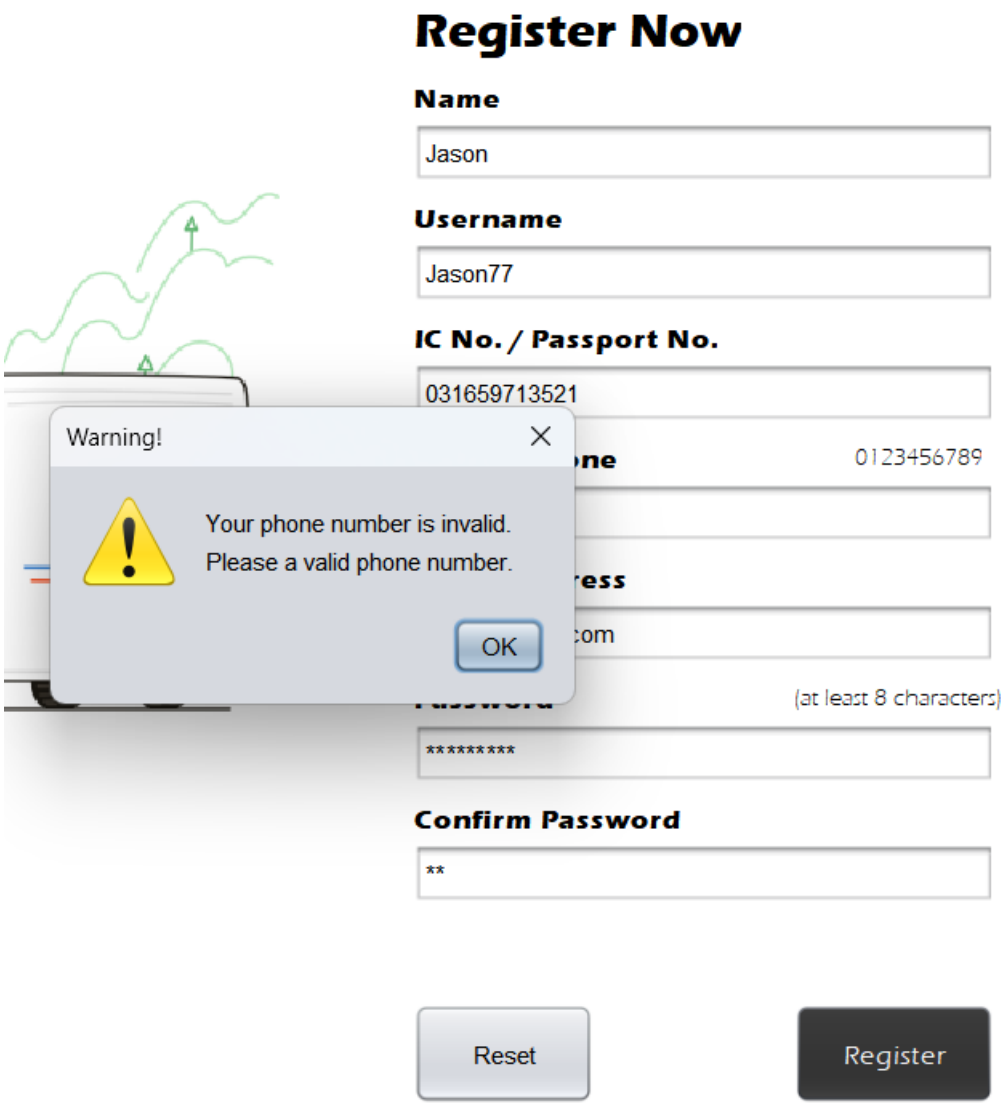


Figure 70 Customer Registration

This warning notice will show if the contact number in the registration form does not contain precisely 10 digits. This notice requests that customer provide a proper contact number to the system.

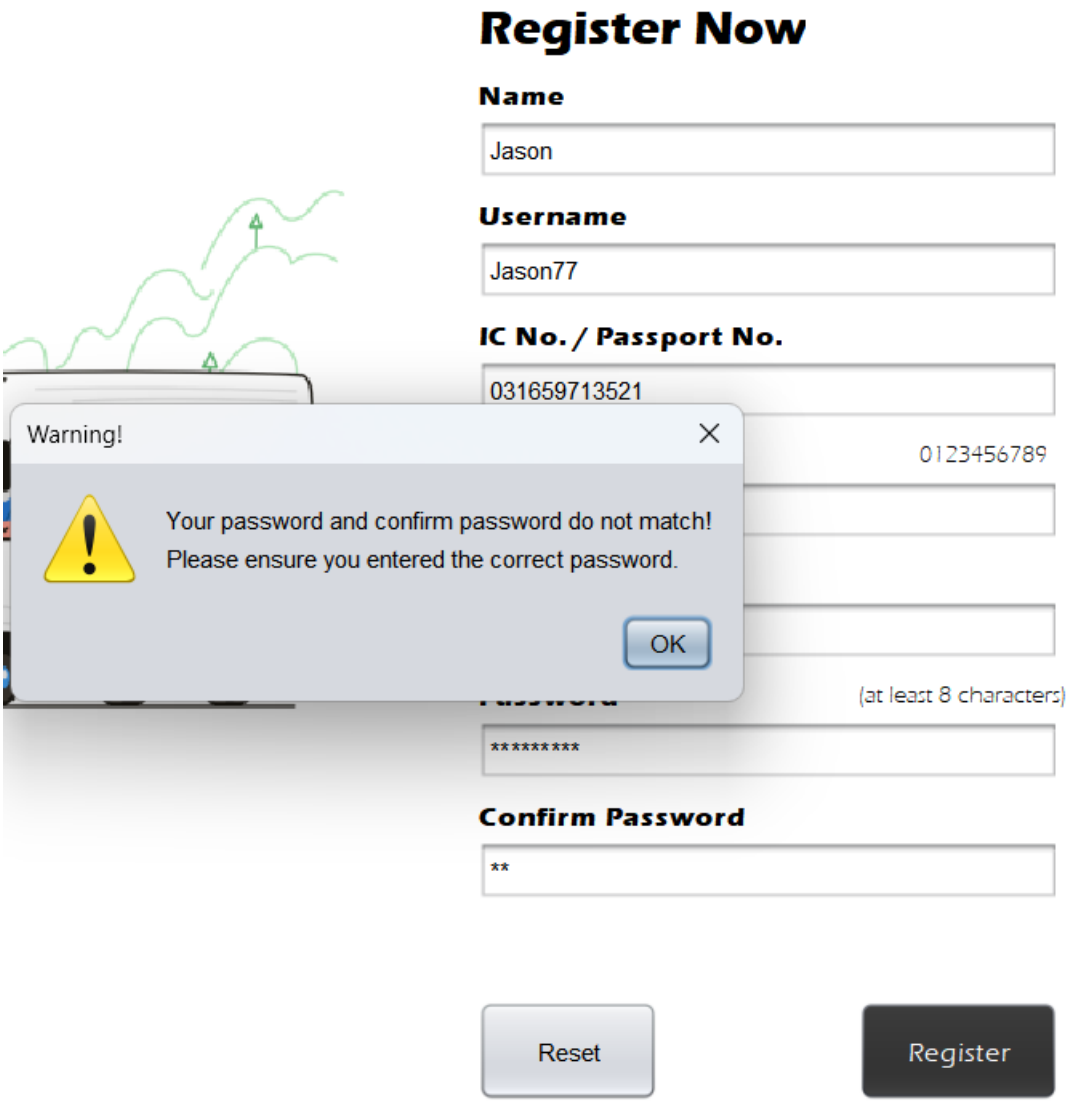


Figure 71 Customer Registration

If the password and confirm password entered on the registration form do not match, this warning message will appear. This notification requests that customers give a valid password in order to prevent typing errors.

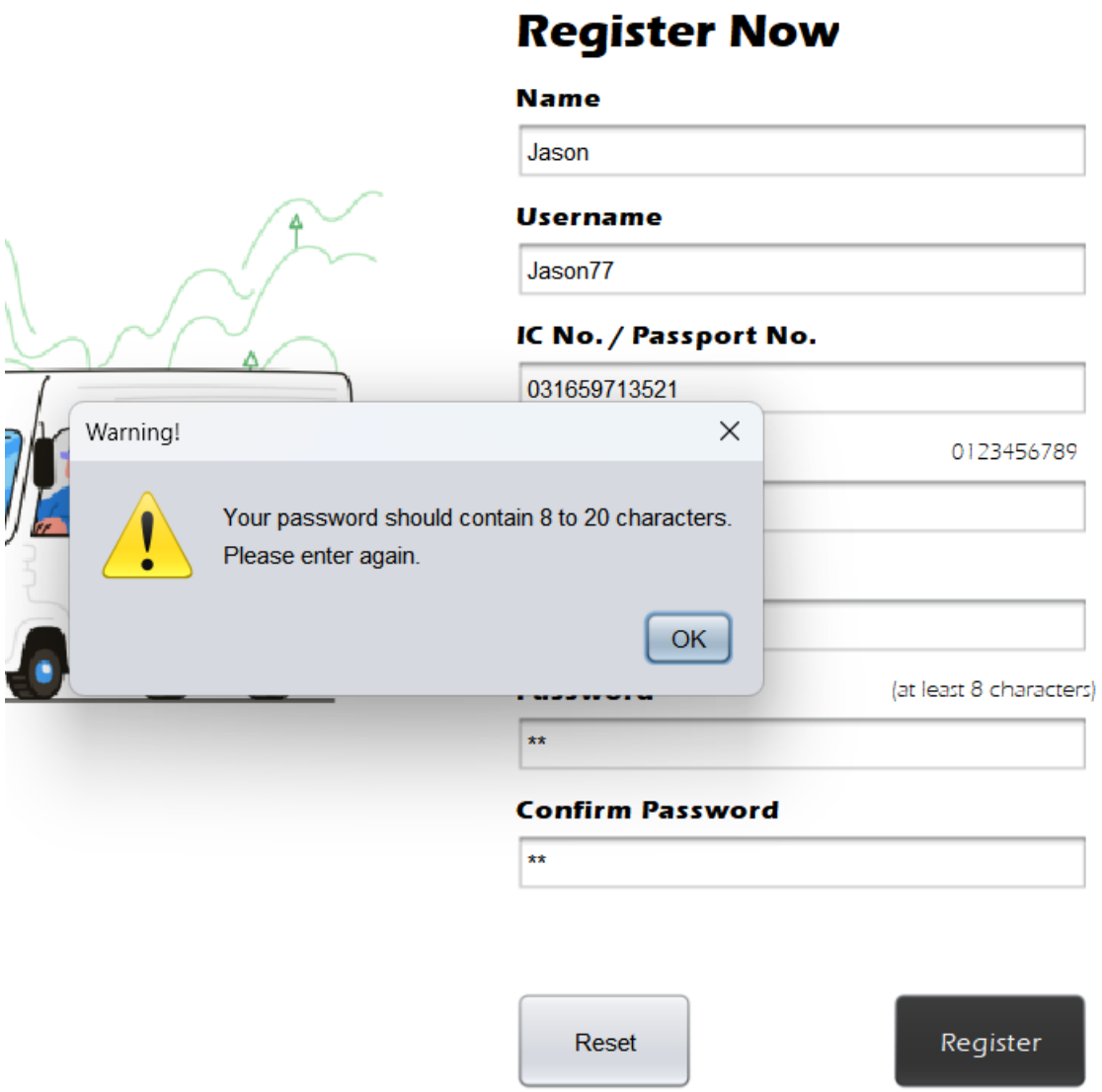


Figure 72 Customer Registration

If the password entered on the registration form does not include between 8 and 20 characters, a warning message will appear. This notice requests that customers give a valid password.

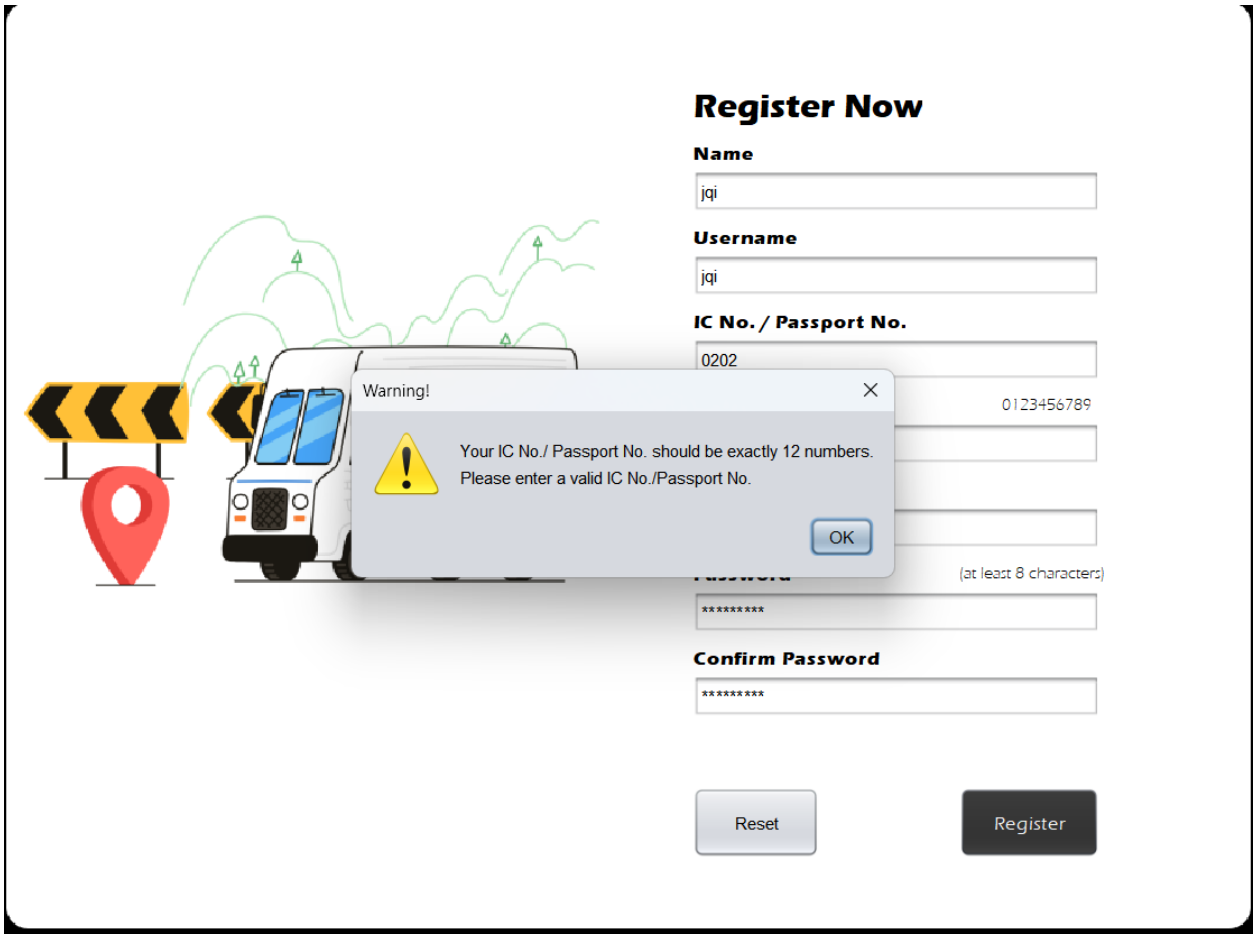


Figure 73 Customer Registration

If the passport number or IC number entered on the registration form does not exactly contain 12 digits, a warning message will be shown. This notice requests that customers provide a valid IC number or passport number into the system.

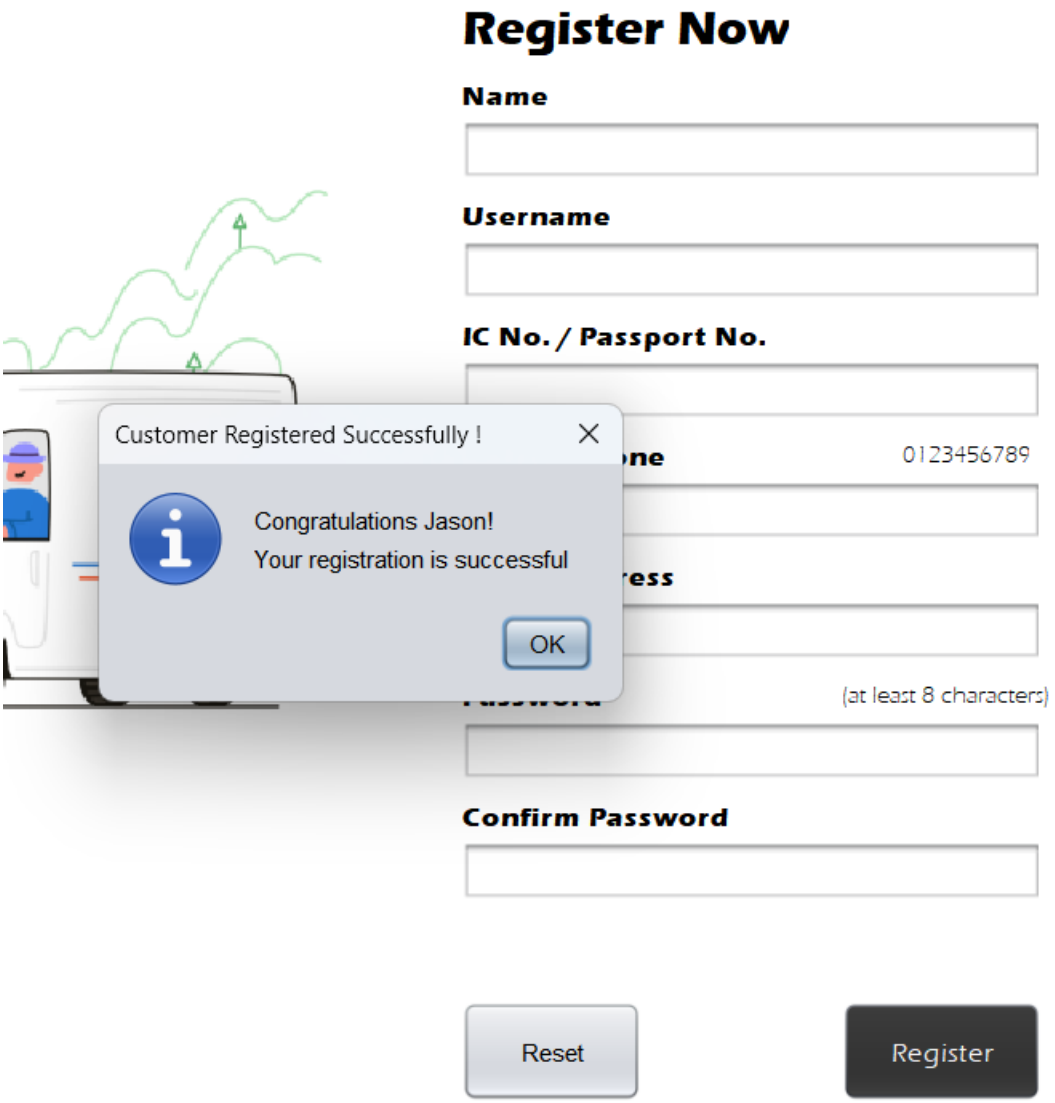


Figure 74 Customer Registration

This message will appear if customers information is valid, click "Register" to proceed to the "Login page."

```
CUSTOMER//asdfas//addcusotmer//111112111111//0128003983//asd@gmail.com//asdfasdf//CUSTOMER
CUSTOMER//asdfas//addc1sotmer//111112111211//0128003983//asd@gmail.com//asdfasdf//CUSTOMER
CUSTOMER//Jason//Jason77//031659713521//0132659879//jason@gmail.com//123456789//CUSTOMER
```

Figure 75 Customer Registration

The registered details will save into the user.txt file.

5.3.3 **Customer Menu**

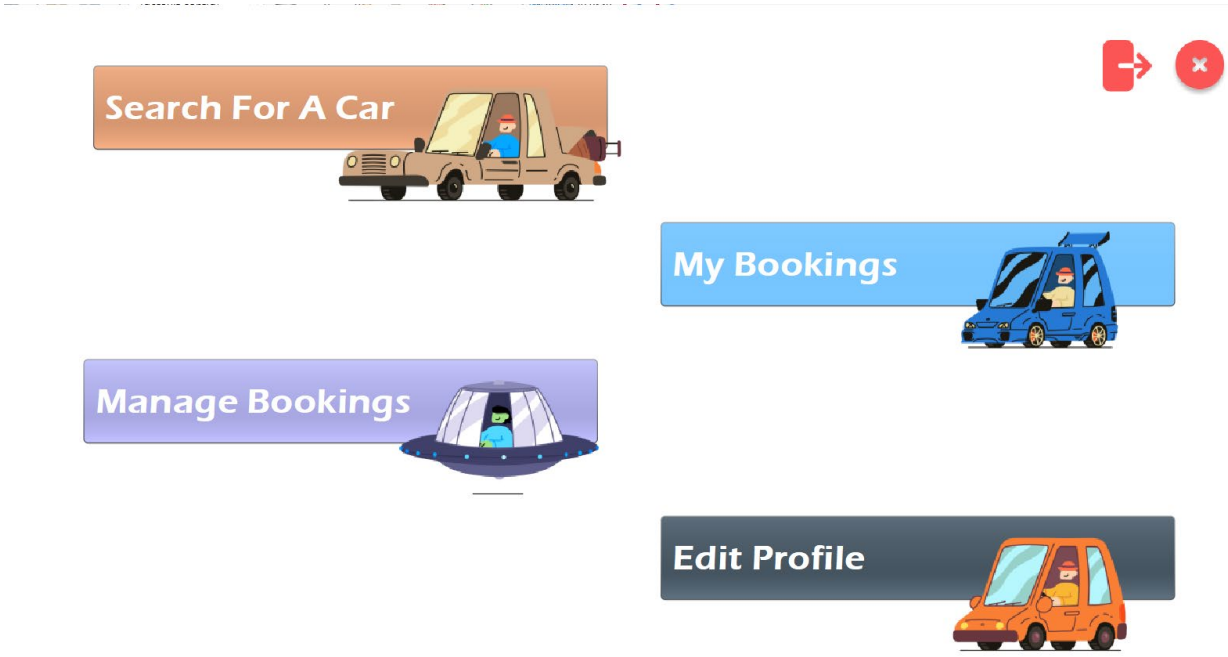


Figure 76 Customer Menu

The system's main menu may be seen on this page, which is titled "Customer Menu." The choices for customers are Edit Profile, My Bookings, Manage Bookings, and Search For A Car. Customers get easy accessibility to these features.

5.3.4 Search For A Car

Figure 77 Search For A Car

Customers may use this page to search for a car by entering pickup time, return time, pickup location, pickup date and return date.

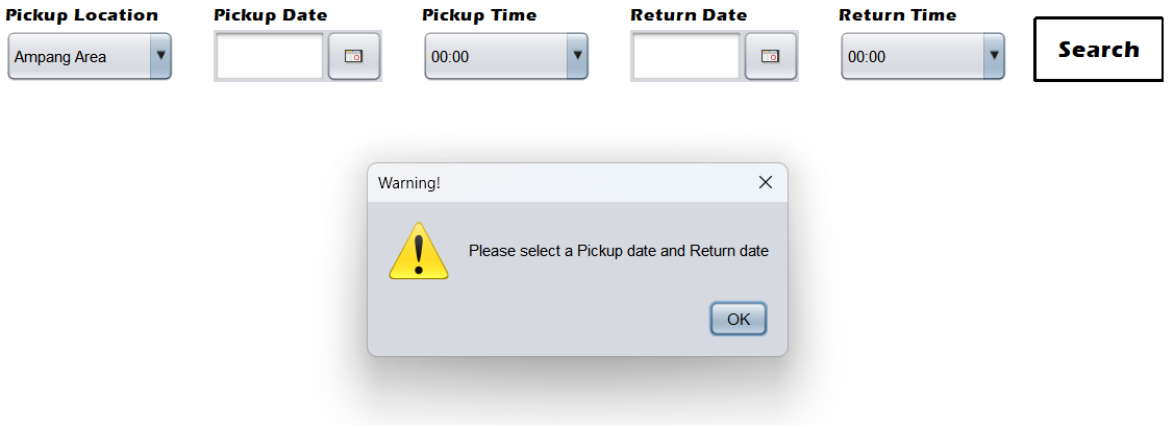


Figure 78 Search For A Car

This warning message would pop up, if customers forgot to select the Pickup Date or Return Date.

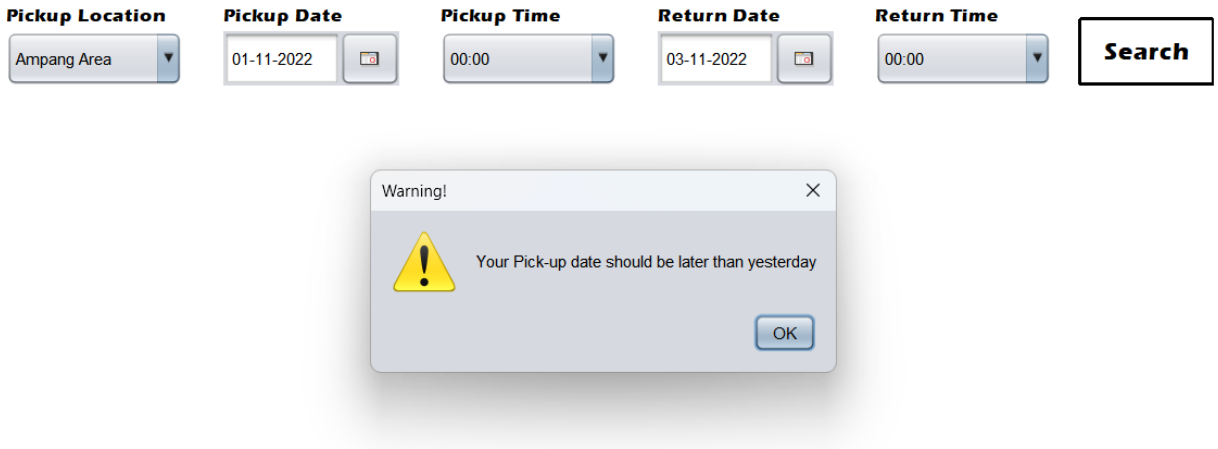


Figure 79 Search For A Car

This message will pop up if the Pickup Date is before today's Date.

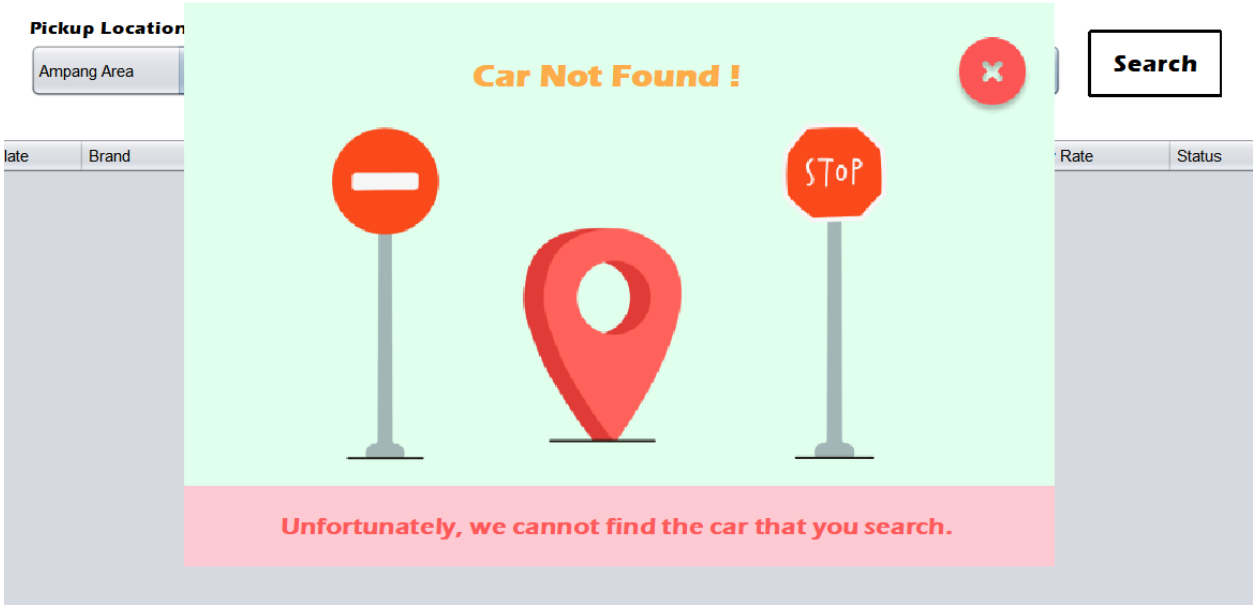


Figure 80 Search For A Car

This message will pop up if there are no available car between the date.

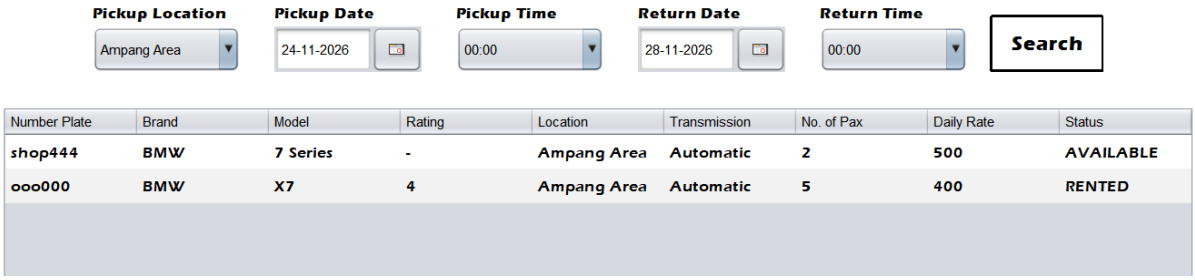


Figure 81 Search For A Car

After select all the option, it will show up a table and car details.

5.3.5 Customer Make Booking

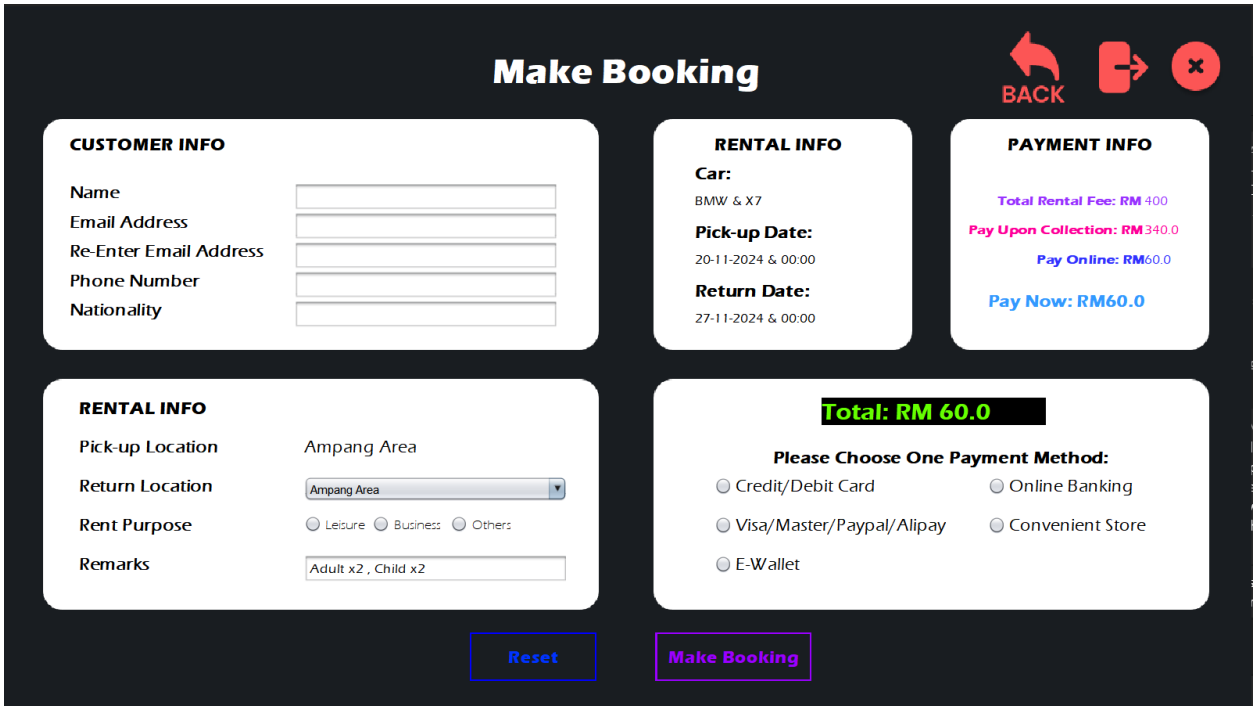


Figure 82 Search For A Car

This is the booking page where customers can enter their information such as name, email address, phone number, nationality. Besides, customer can select Return Location, Rent Purpose, Remarks and Payment Method. Before press the “Make Booking” button, must check the Rental Info and Payment Info. After checking, click the “Make booking” to book the car.

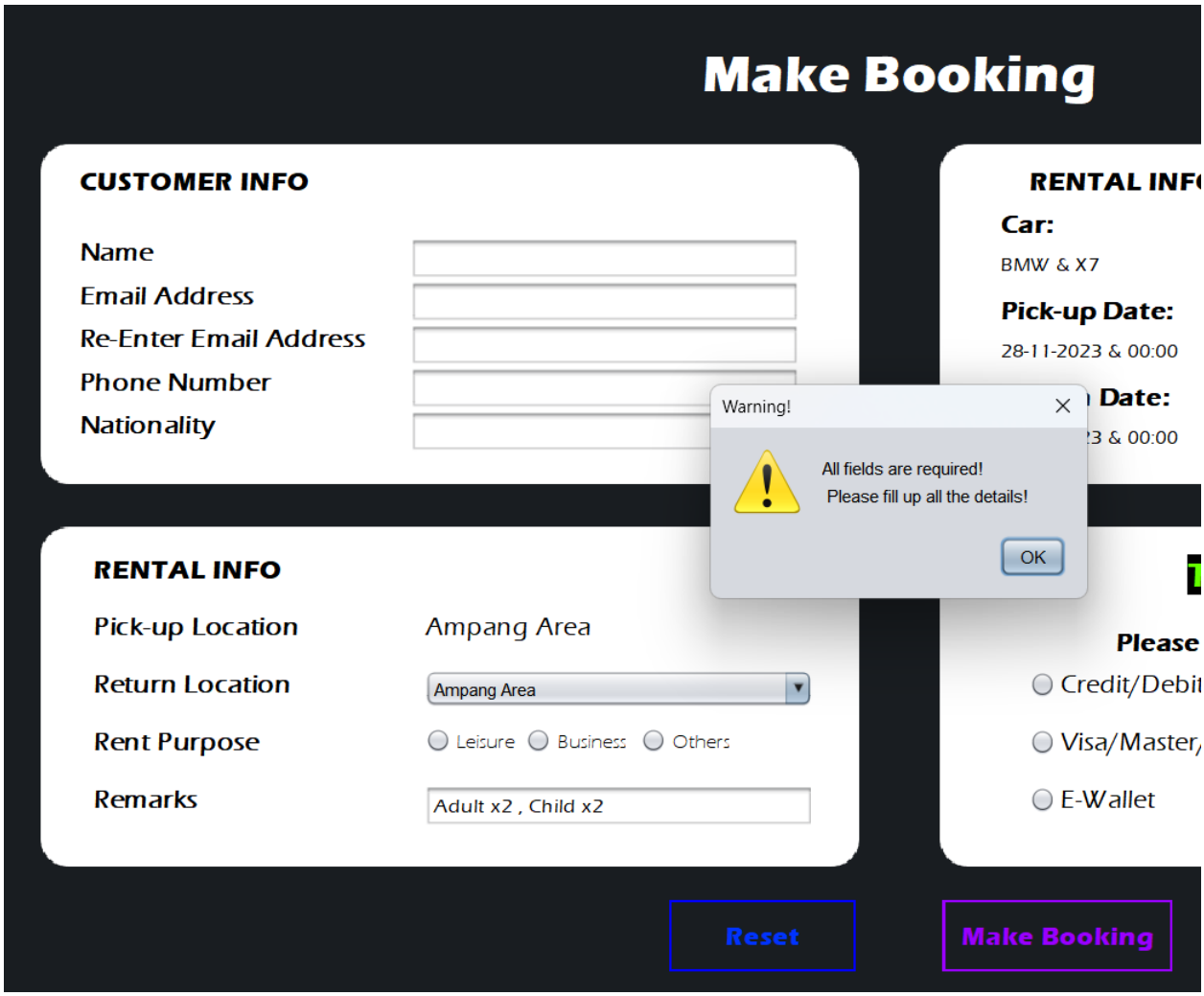


Figure 83 Search For A Car

This warning message will show up if customers didn't key in the information.

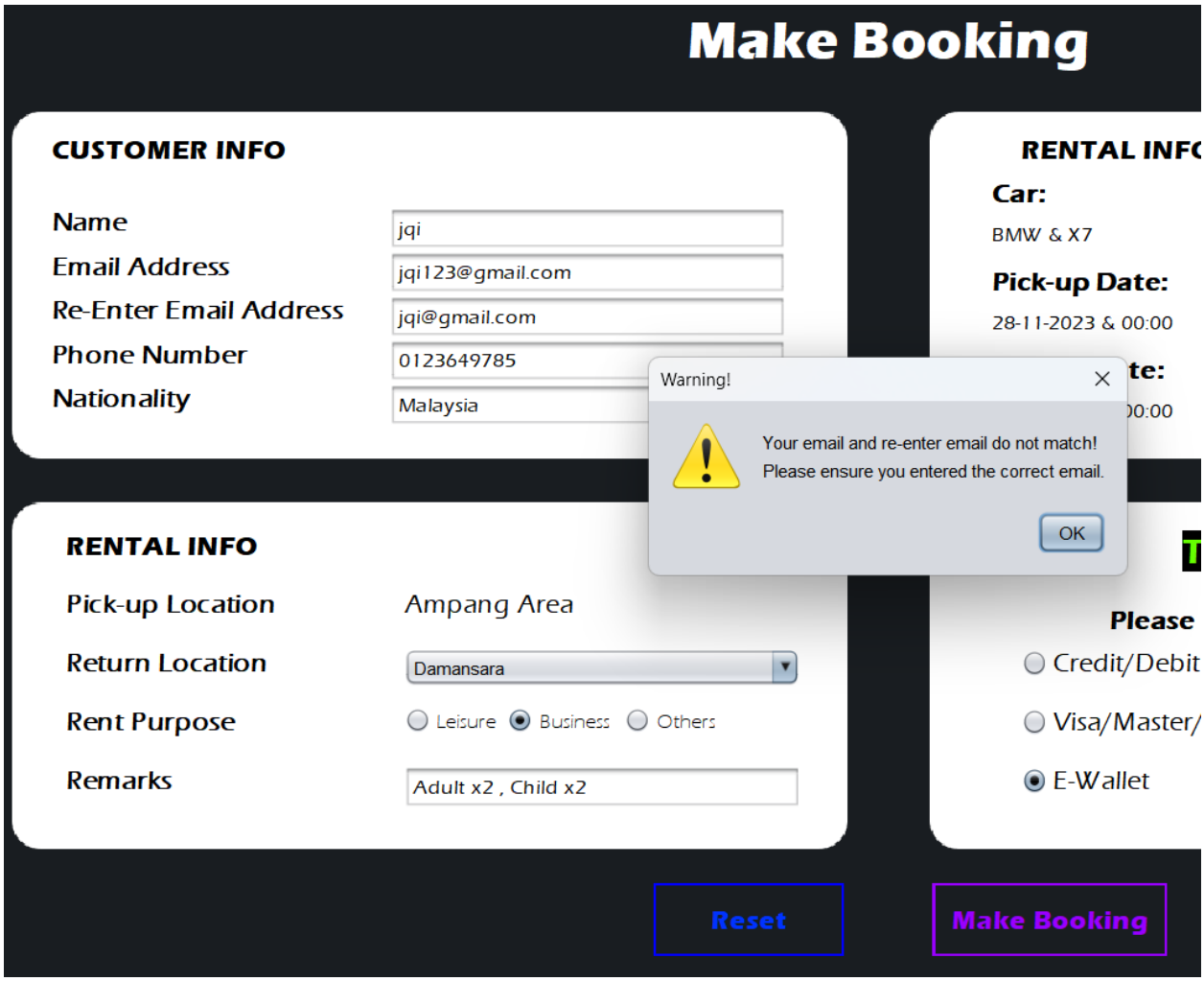


Figure 84 Search For A Car

This warning message will show up if the Email Address didn't match to the Re-Enter Email Address field.

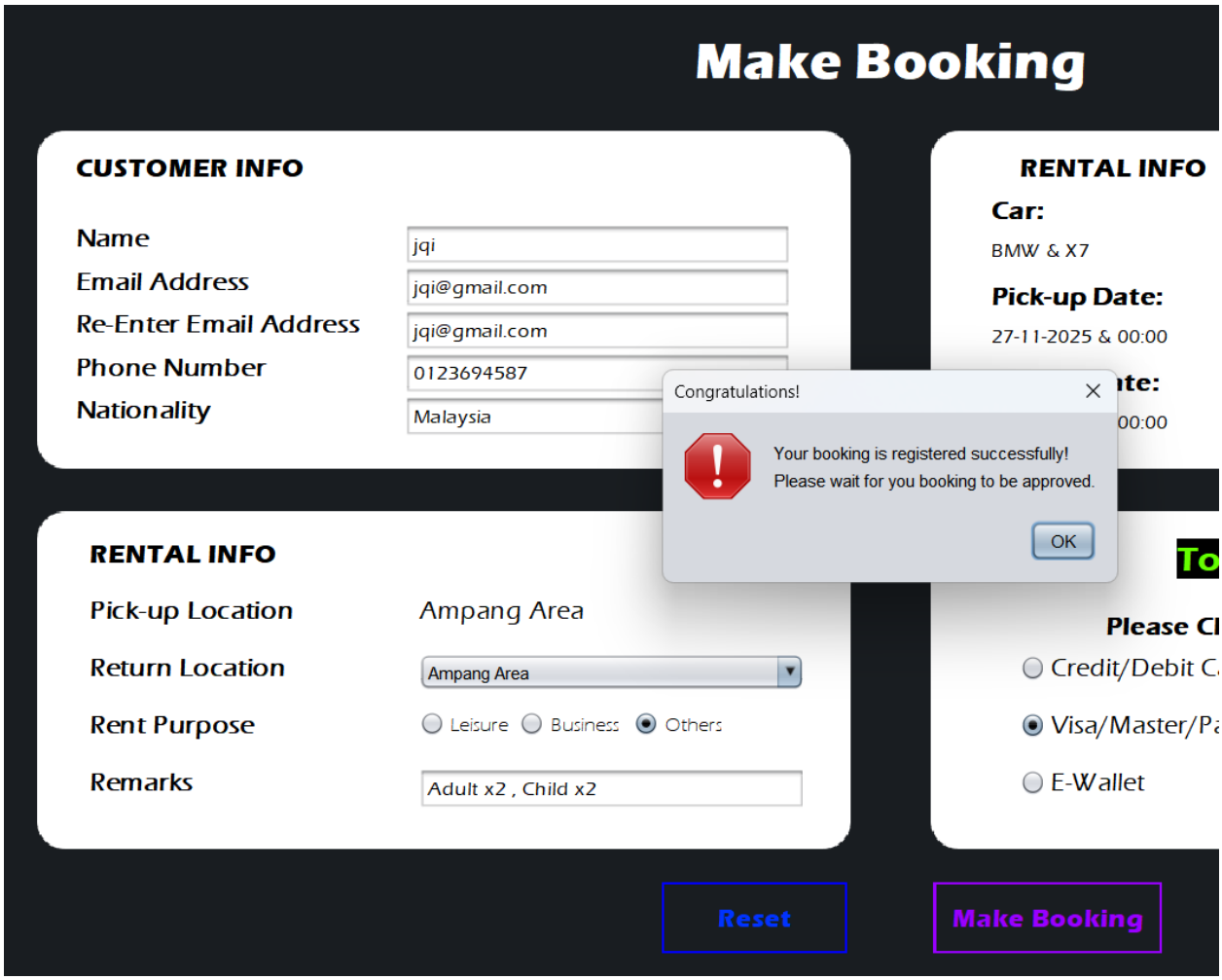


Figure 85 Search For A Car

After customers press the “Make booking” button. The system will show “Your booking is registered successfully” and customers need to wait for admin to approve the booking.

```
0//913023120401//Fancie Tasseler//ftasseler0@uiuc.edu//3733275855//Brazil//aapl666//Jeep//Grand Cher
1//913040120401//Fancie Tasseler//ftasseler0@uiuc.edu//3733275855//Brazil//aapl666//Jeep//Grand Cher
2//492969304142//Almeta Burner//aburner1@chronoengine.com//8155700657//Brazil//ooo000//Toyota//Land
3//718489117132//Baldric Clubley//aclubley2@creativecommons.org//4146623243//Philippines//sq66//Olds
4//243952851706//Addison Labes//alabes3@i2i.jp//2786390604//Philippines//pins222//Audi//A8//KLCC//Da
5//697435917316//Eb Grayland//egrayland4@omniture.com//7046537258//Russia//aapl666//Ford//Crown Vict
6//948675668951//Daveen Tapp//dtapp5@cbslocal.com//3016031978//Azerbaijan//shop444//Jaguar//S-Type//
7//710306971534//Sophie Teo//lscully6@joomla.org//8599542623//Mongolia//pins222//BMW//7 Series//Pand
8//686901604184//Sony Marple//lmarple7@plala.or.jp//9649244762//France//ooo000//Lamborghini//Diablo/
9//291498602721//Mandie Whitely//mwhitely8@weibo.com//8292280123//Russia//isrg333//Mitsubishi//Diama
10//761074099274//Dory Edgett//dedgett9@macromedia.com//9867756430//Japan//pins222//Chevrolet//Corve
11//320644094244//Ravi Smallacombe//rsmallacombea@blogtalkradio.com//6767573420//Indonesia//pins222/
12//330420851536//Evyn Haughan//ehaughanb@businesswire.com//2981832301//China//amd555//Ford//E350//P
13//845970795849//Jillie McDill//jmcdillc@networksolutions.com//1972787938//Sweden//ooo000//Volvo//C
14//692175385845//Domenico Conwell//dconwelld@oakley.com//6623359832//Indonesia//etsy9999//Audi//rio
15//517437103339//Noble Radki//nradkie@economist.com//6296199363//China//aapl666//Chrysler//New York
16//128457960013//Andris Talby//atalbyf@foxnews.com//4303776414//Peru//aapl666//BMW//600//Subang Jay
17//578618437965//Karlotta Plover//kploverg@chicagotribune.com//8416768056//France//shop444//Mitsubi
18//858333795167//Cyrus Musla//cmuslah@tinypic.com//2809077993//Poland//aaa001//Audi//R8//Sepang Are
19//968510014830//Gwenette Hazleton//ghazletoni@elpais.com//9514126990//Finland//sq66//Chevrolet//Ta
20//704216528019//Latashia Latham//llathamj@constantcontact.com//4667075510//Sweden//pins222//GMC//R
21//307569702475//Emmerich Mathers//emathersk@reverbnation.com//4958485584//France//shop444//Mercury
22//273202270422//Maud Phythean//mphytheanl@ox.ac.uk//6499037724//Nicaragua//aaa001//Chrysler//Cirru
23//590368859831//Sergei Cominello//scominellom@nasa.gov//7603162061//Russia//shop444//Volkswagen//J
24//141084798956//Wainwright Tremble//wtremblen@princeton.edu//4546433112//Czech Republic//ooo000//G
25//734073727084//Sela Brayn//sbrayno@cnet.com//1608828603//France//tsl888//Fairthorpe//Rockette//Sh
26//768988582119//Brett Lias//bliasp@domainmarket.com//8153694042//Russia//sq66//Lincoln//MKZ//Ampan
27//020612140368//Ford Edscer//fedscerq@narod.ru//5094026840//China//isrg333//Chrysler//Town & Count
28//020612140368//Zachery herro//zarminr@joomla.org//7111445342//Brazil//amd555//Ford//F350//Sepang
29//020612140368//Linda Darry//linkles@chronoengine.com//6534779777//Malawi//shop444//GMC//Yukon//Pu
30//020612140368//Kayley Isles//kislest@shutterfly.com//1234567890//Ukraine//etsy9999//Ford//E250//P
31//020612140368//Ferry Tail//aas@gmail.com//0123749323//Malaysian//tsl888//Mercedes-Benz//S-Class//
32//111111111111//Haydie//asdf@gmail.com//0123456789//Malaysia//etsy9999//Audi//E-tron//Kepong Area/
```

Figure 86 Search For A Car

After make booking, all the booking details will store into the booking.txt file.

5.3.6 My Booking

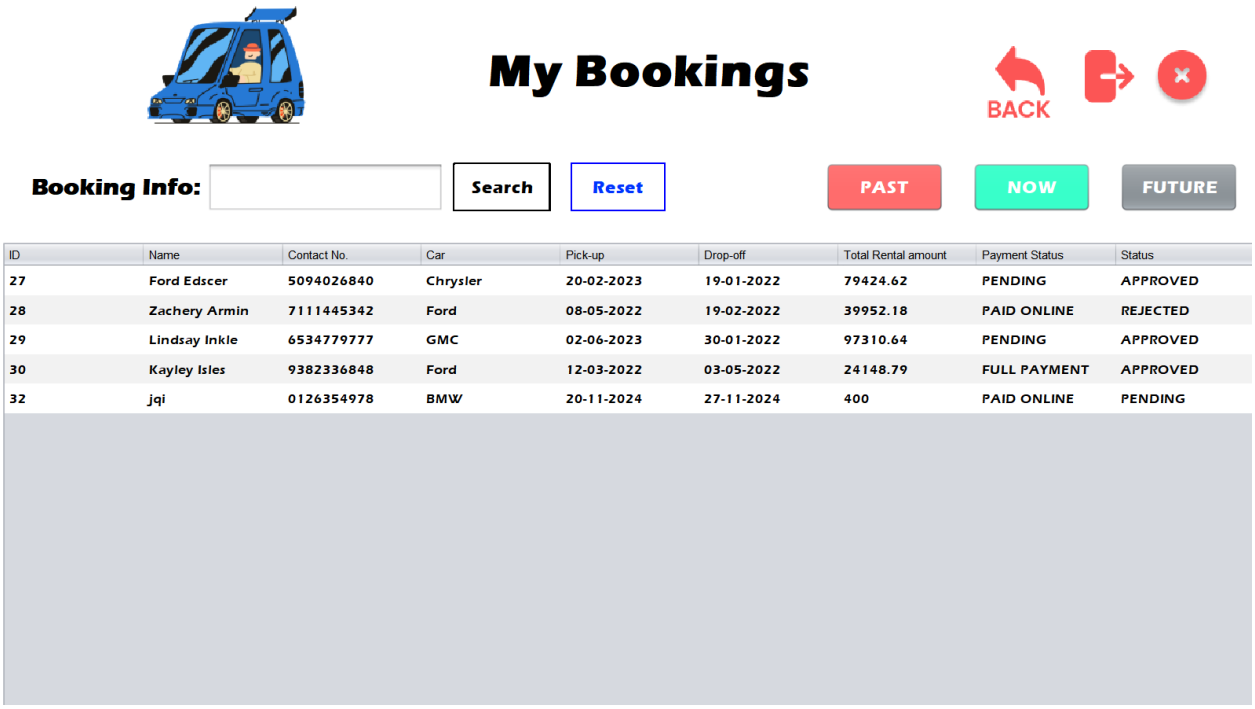


Figure 87 My Booking

This is “My Bookings” page where for customers to check all their bookings. Furthermore, customers able to click the booking which inside the table to check the booking details, give some feedback and check the receipt.

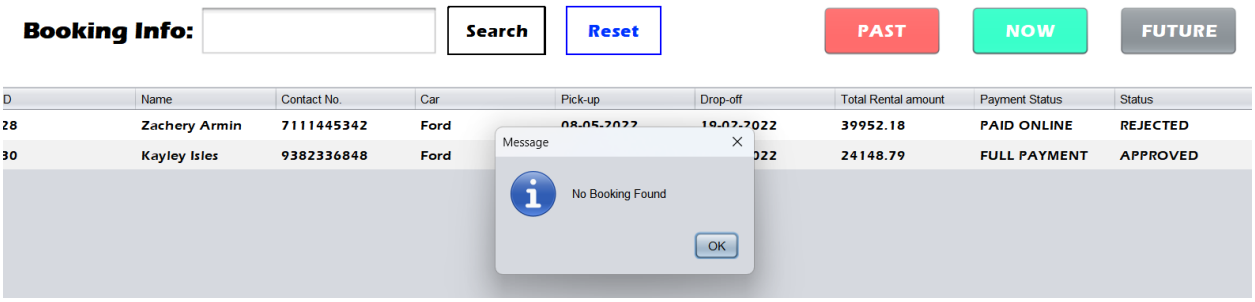


Figure 88 My Booking

This message will pop up if there are no booking found for the customer by pressing the search button to search their booking. Customers also can check all the booking by clicking the “Search” button without key in the booking info.

PAST

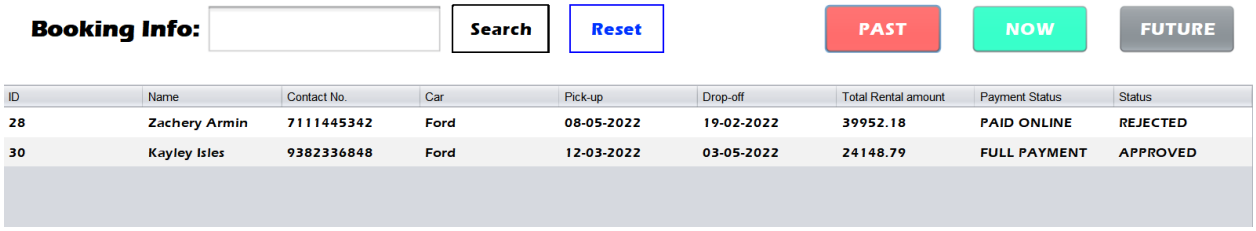


Figure 89 My Booking

Customer can click on the "PAST" button to sort the date and system will show the bookings which bookings are past.

NOW

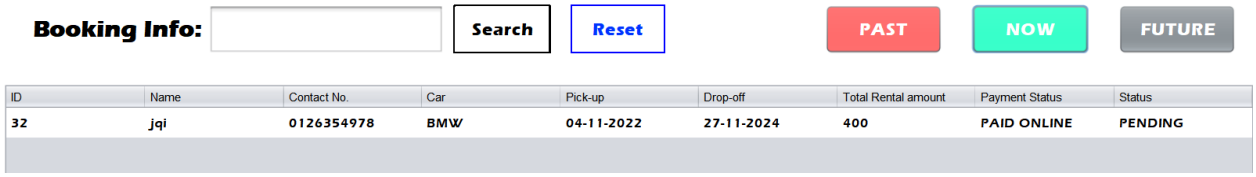


Figure 90 My Booking

Customer can click on the "NOW" button to sort the date and system will show bookings for today.

FUTURE

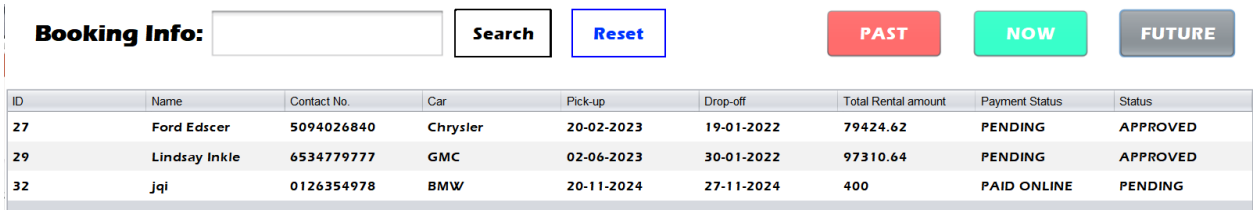


Figure 91 My Booking

Customer can click on the "FUTURE" button to sort the date and system will show the bookings which bookings are coming soon.

Figure 92 My Booking

Customer can click on each of their booking from “My Booking” page to check the booking details. Moreover, there have two button such as “Feedback” and “Receipt” button for customers to check the receipt and submit their feedback.

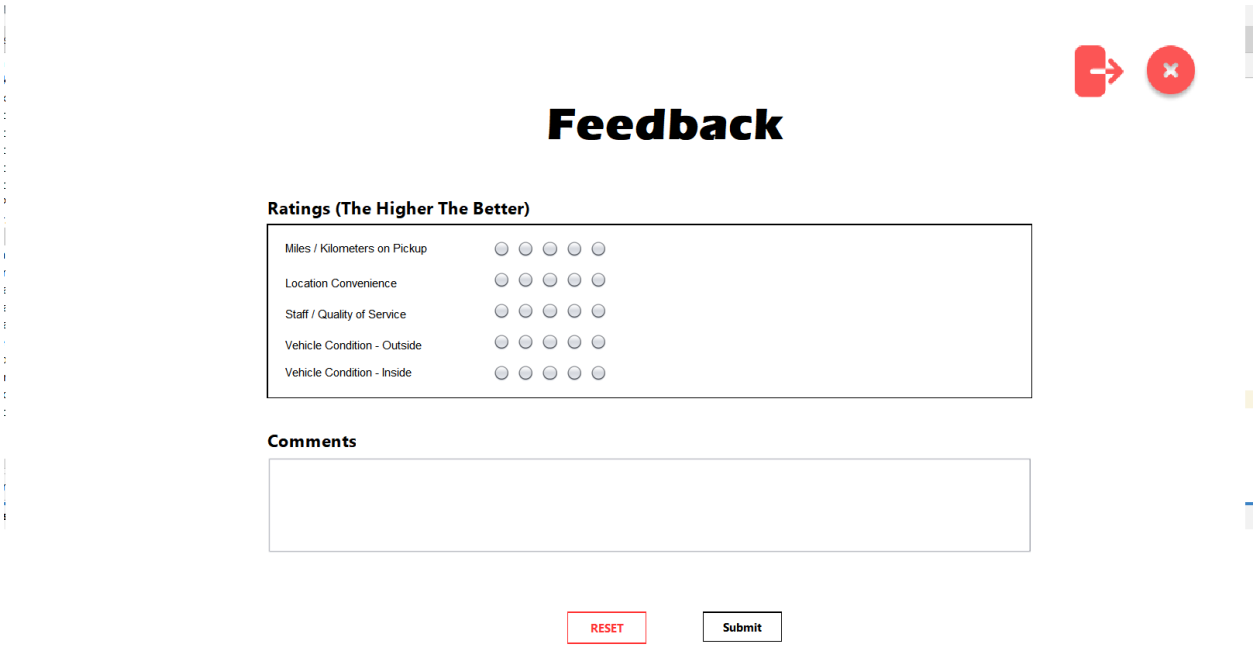


Figure 93 My Booking

This is “Feedback” page for customers to give rating and comments for their booking. In the Rating column, there has “Miles or kilometre on Pickup”, “Location Convenience”, “Staff or Quality of Service”, “Vehicle Conditions on Outside” and “Vehicle Conditions on Inside” and there has one text field for customer to enter their comments.

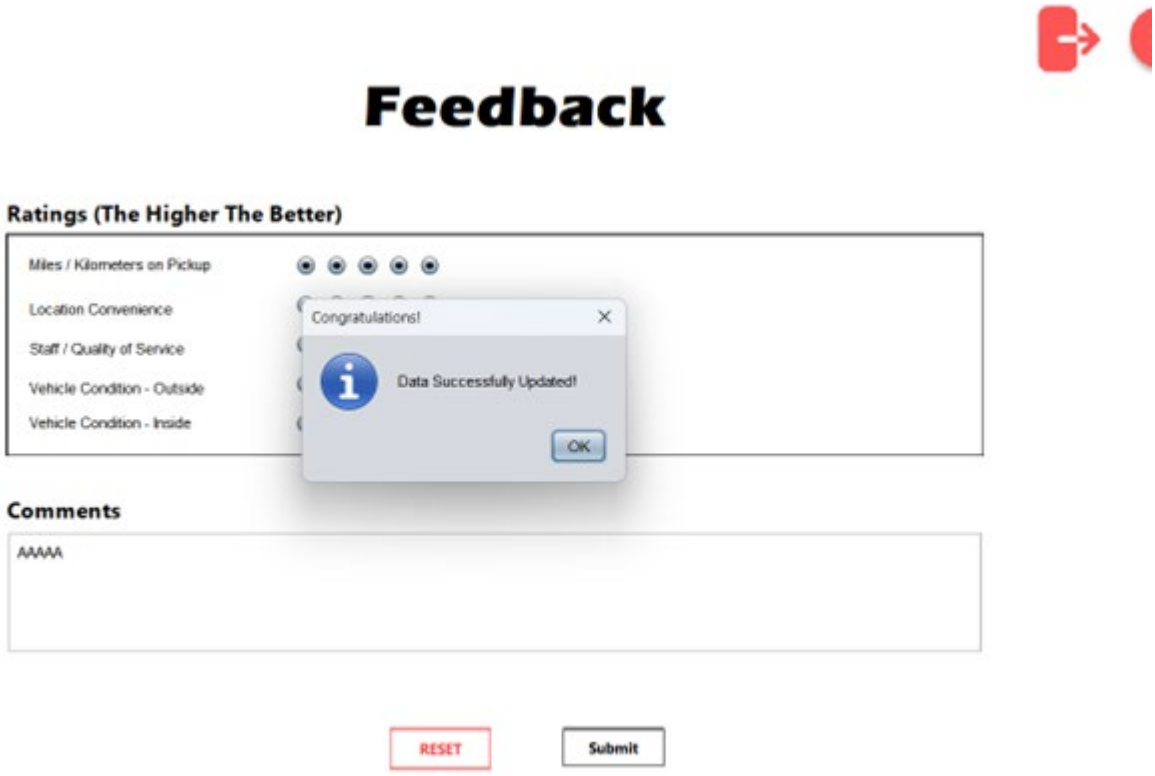


Figure 94 My Booking

After enter all the information, click the “Submit” button to submit their feedback or click the “RESET” button to clear all the feedback.

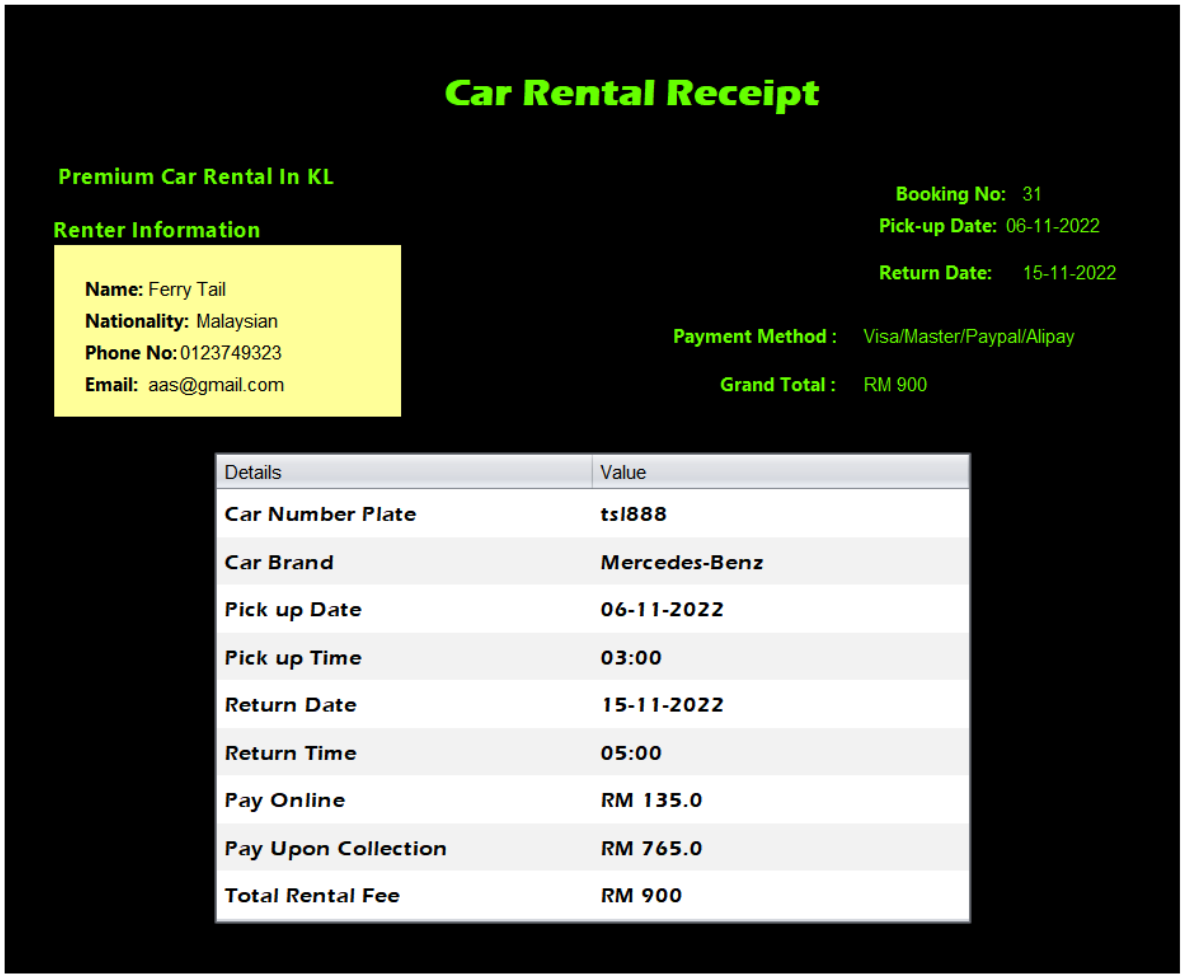


Figure 95 My Booking

This is the “Car Rental Receipt” page where customers can view their booking details.

5.3.7 Manage Booking

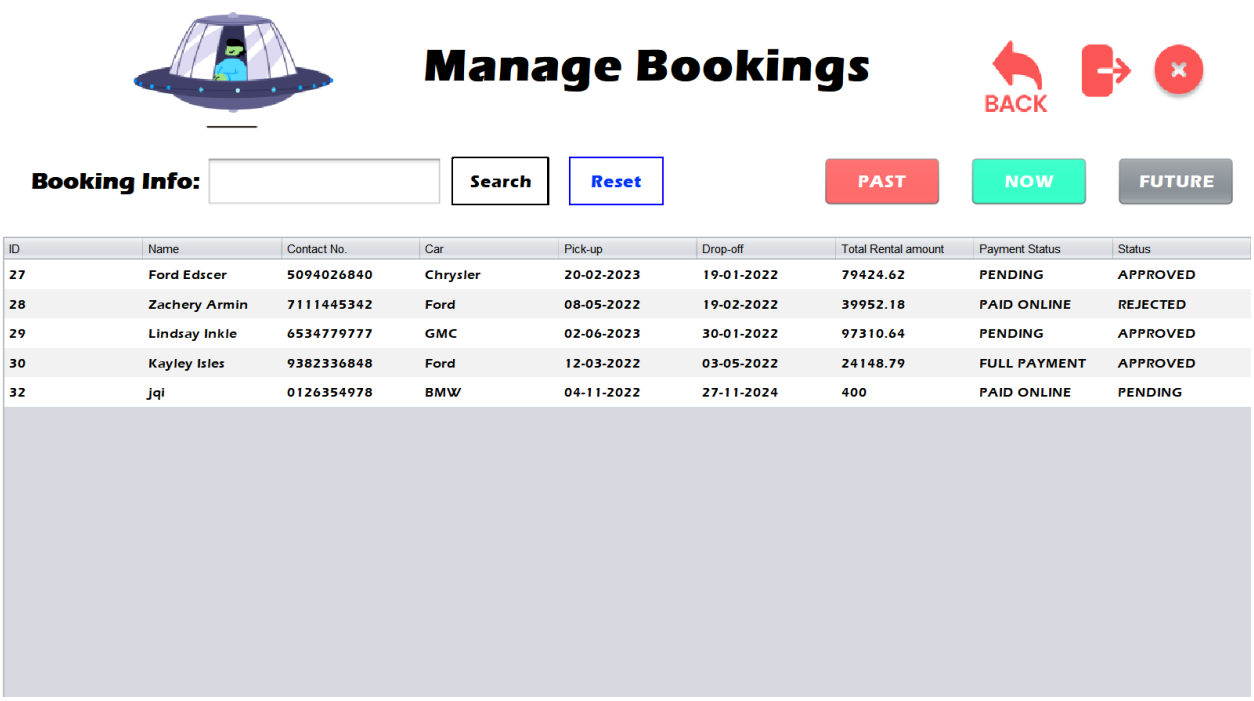


Figure 96 Manage Booking

This is “Manage Bookings” page which is for customers to manage their bookings by selecting the booking.

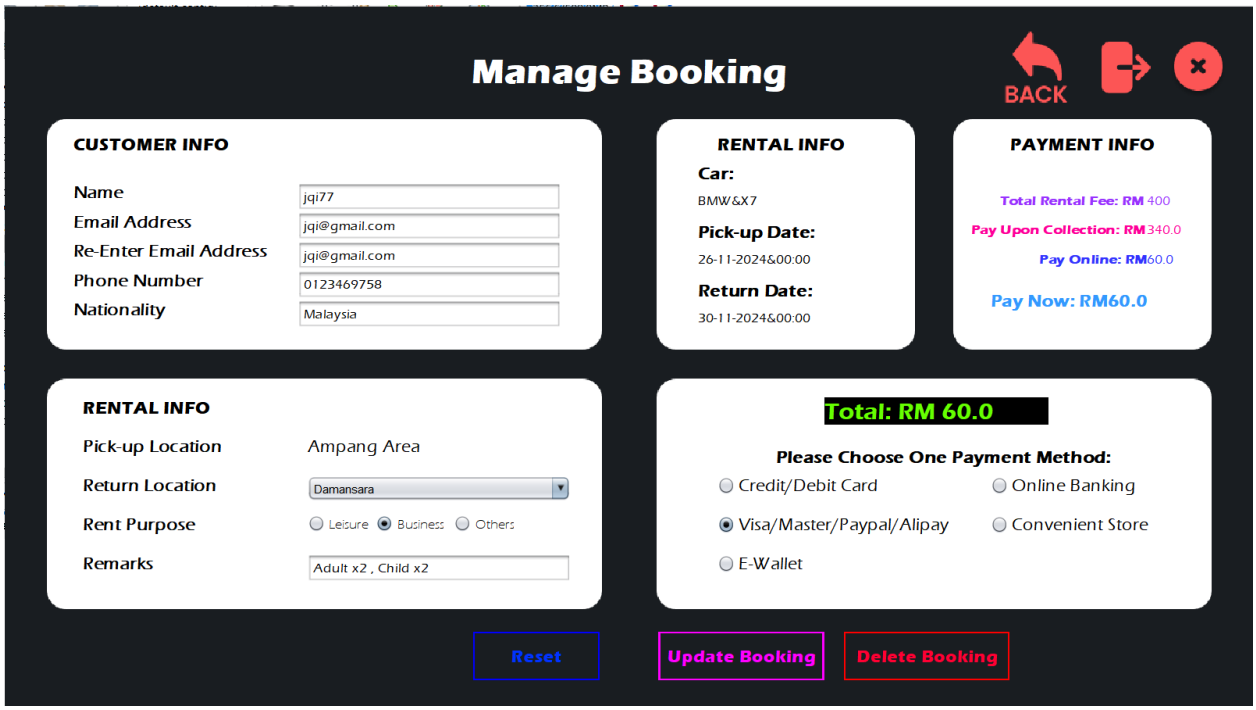


Figure 97 Manage Booking

After selected a booking, the system will redirect customers to “Update Booking” page for them to update booking or delete the booking.

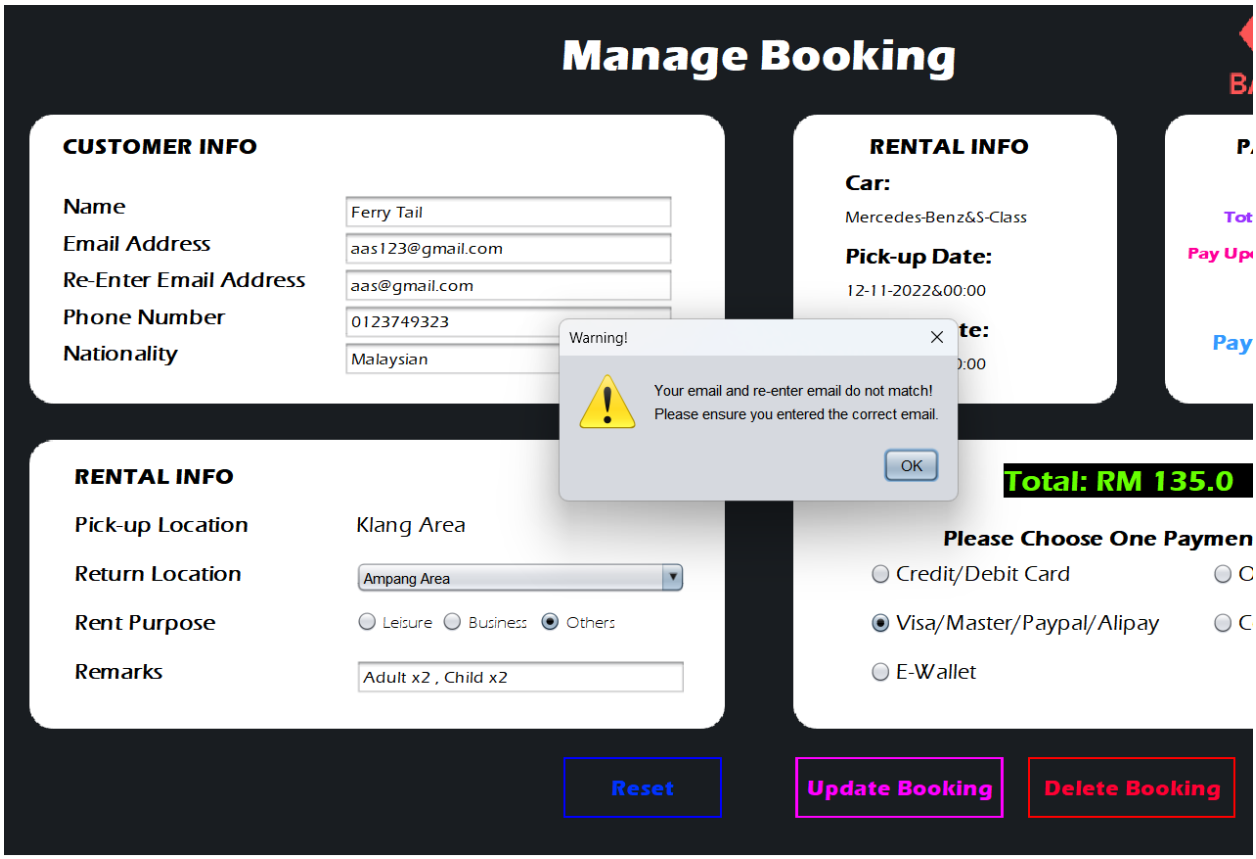


Figure 98 Manage Booking

This warning message will show if the Email Address and Re-Enter Email Address used in the Manage Booking page do not match. This notification requests that customers give a valid Email Address in order to prevent typing errors.

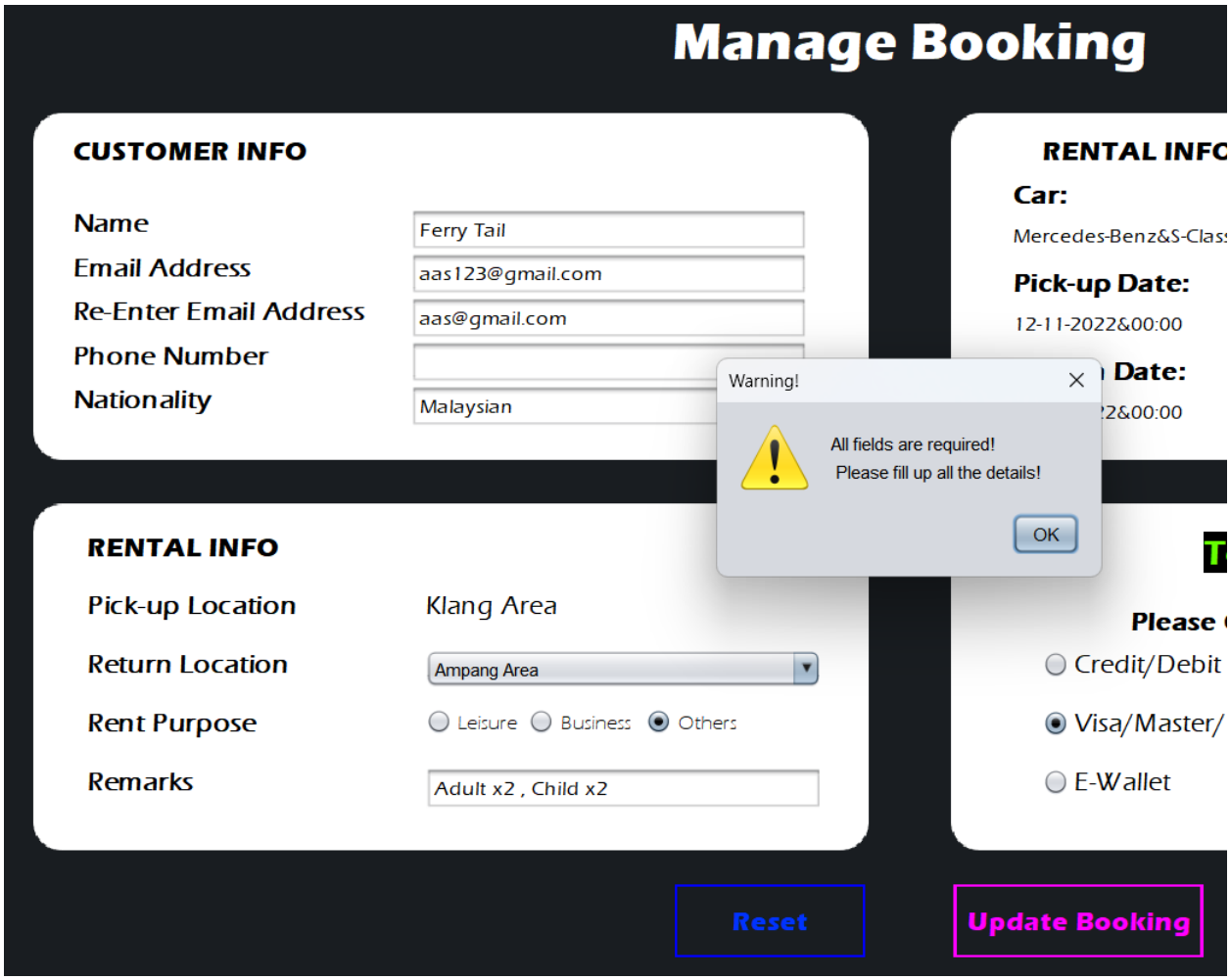


Figure 99 Manage Booking

This warning notice will show if the customer information on the Manage Booking page is incomplete. This notification instructs customers to enter all information.

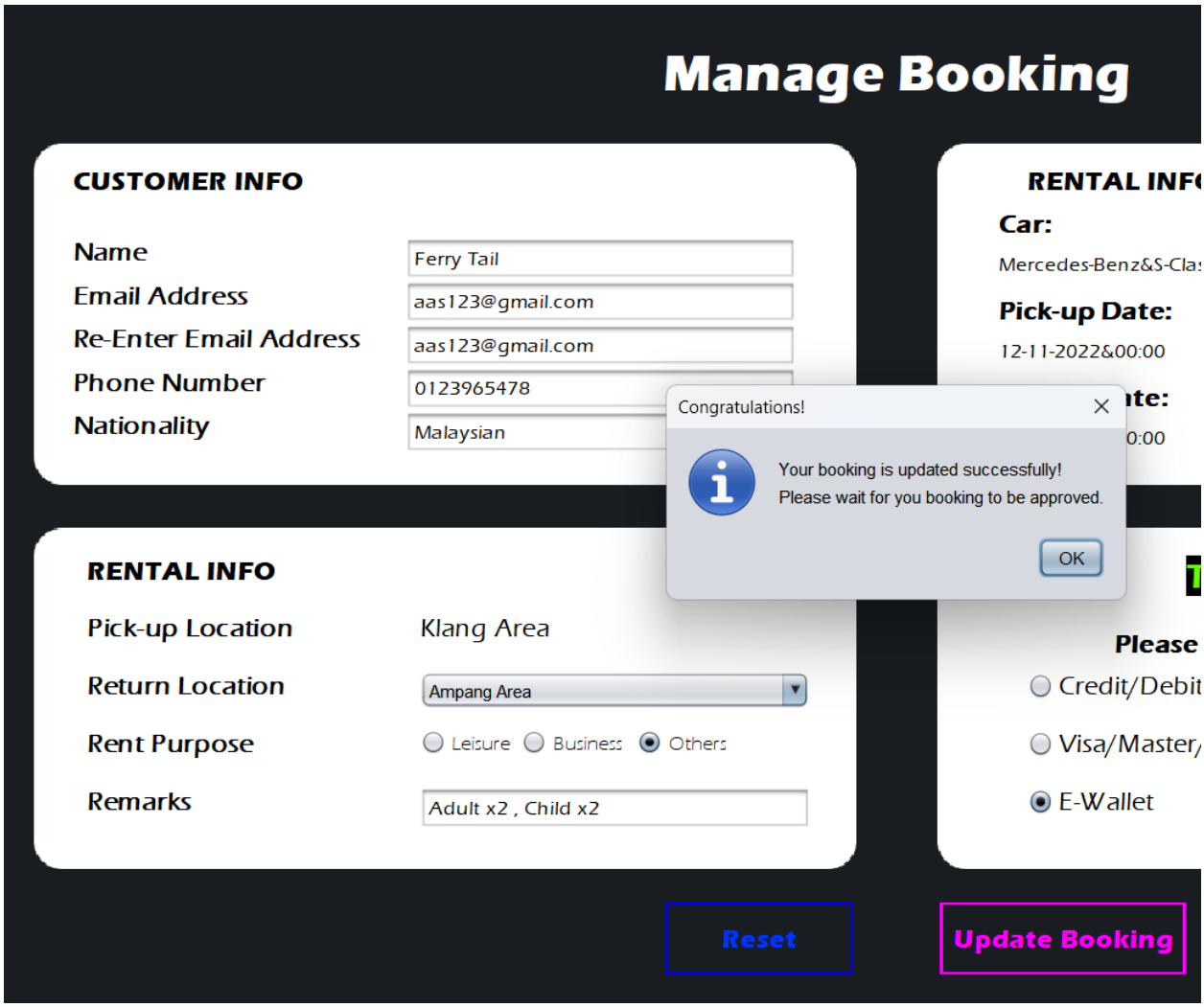


Figure 100 Manage Booking

This message will show when customers input all the valid information and click "Update Booking" button. The updated booking need to wait for admin to approve.

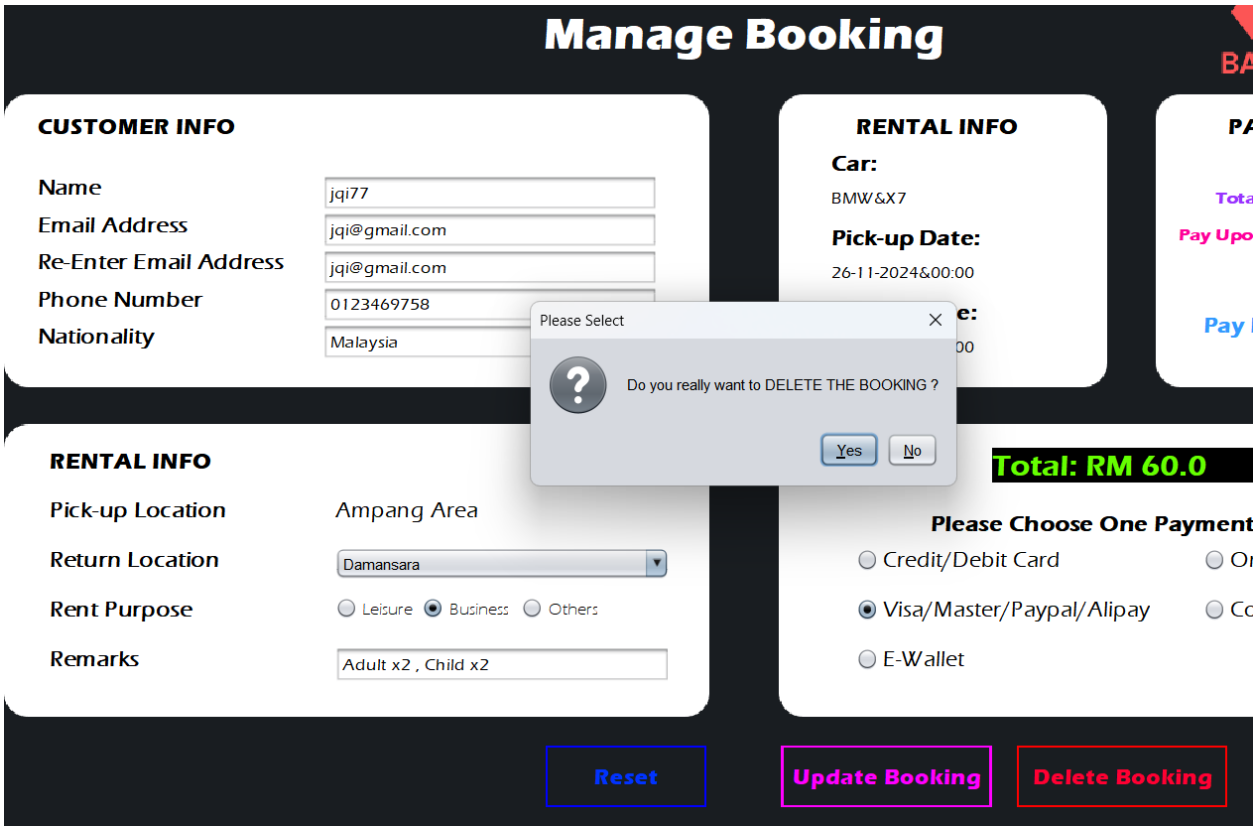


Figure 101 Manage Booking

This message will show for customers to confirm their decision if customer clicked on “Delete Booking” button. Press “Yes” to delete the customer from text file otherwise click “No” to cancel the process.

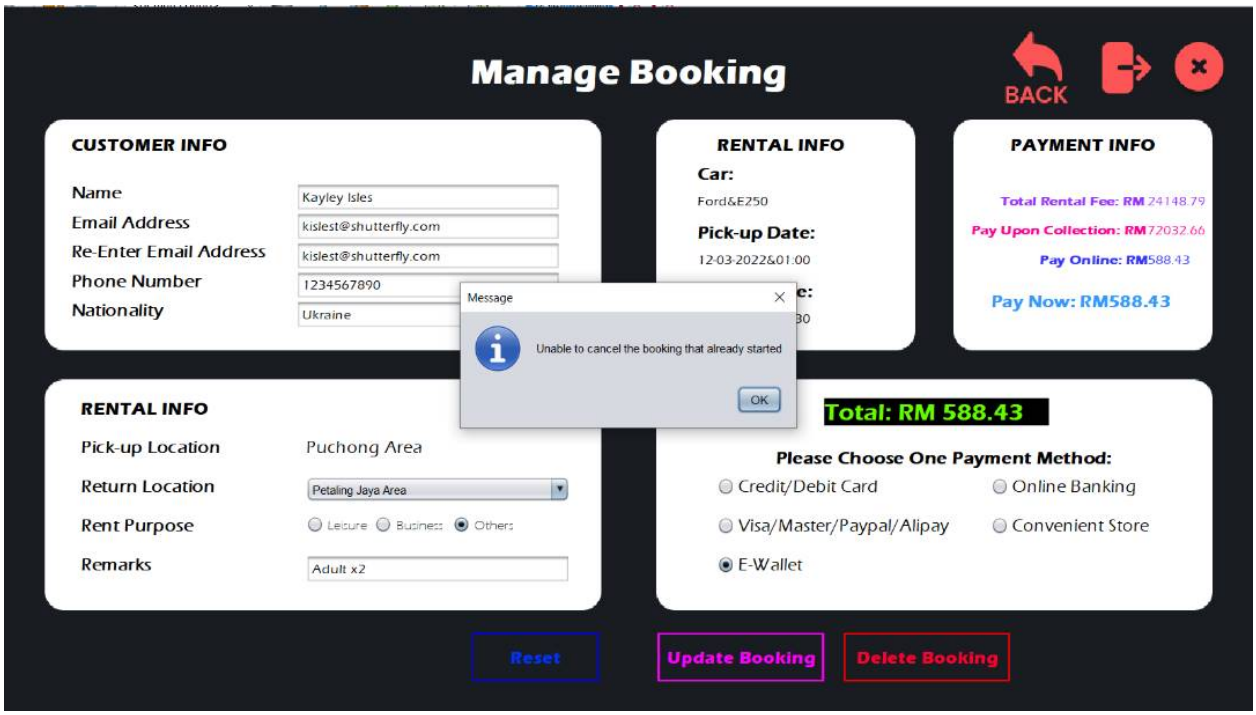


Figure 102 Manage Booking

This message will show if customer want to delete the booking, but the booking already started.

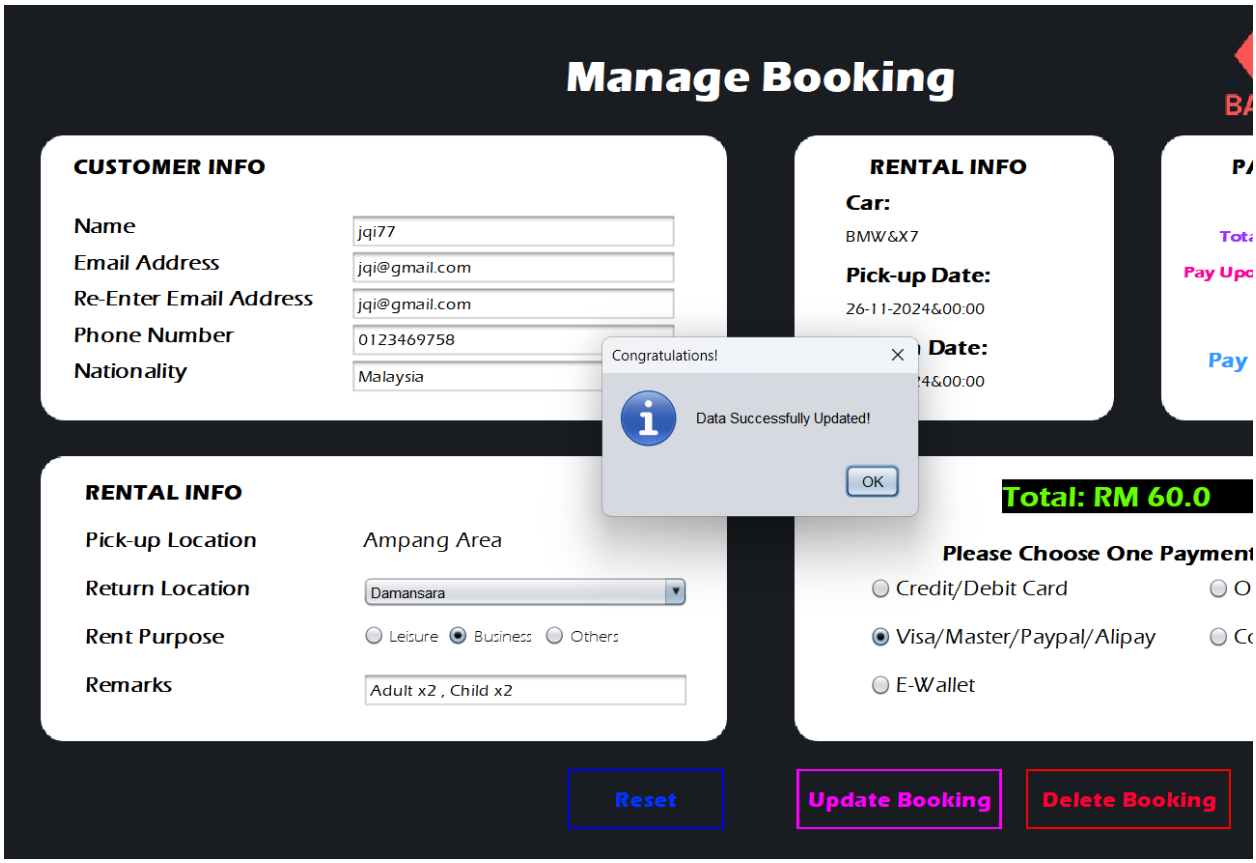


Figure 103 Manage Booking

This message will show if the booking deleted successfully.

5.3.8 Edit Profile

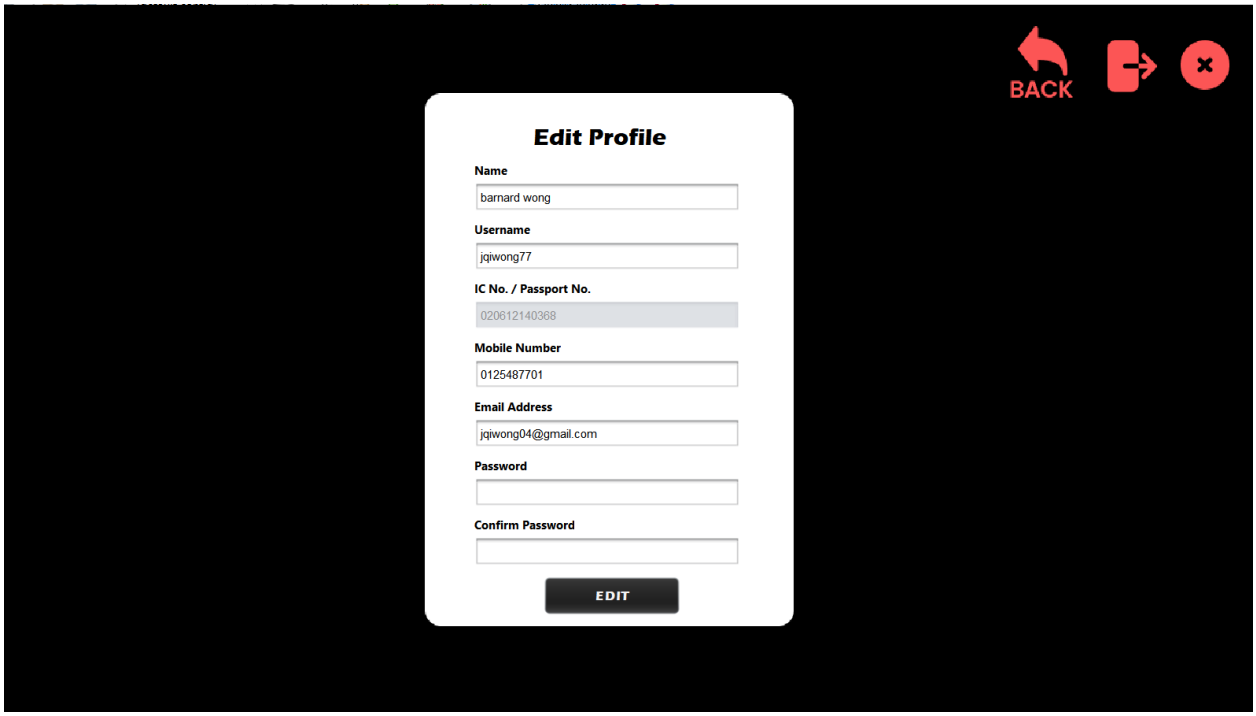


Figure 104 Edit Profile

Customers may modify their personal information on this "Edit Profile" page, with the exception of their IC number.

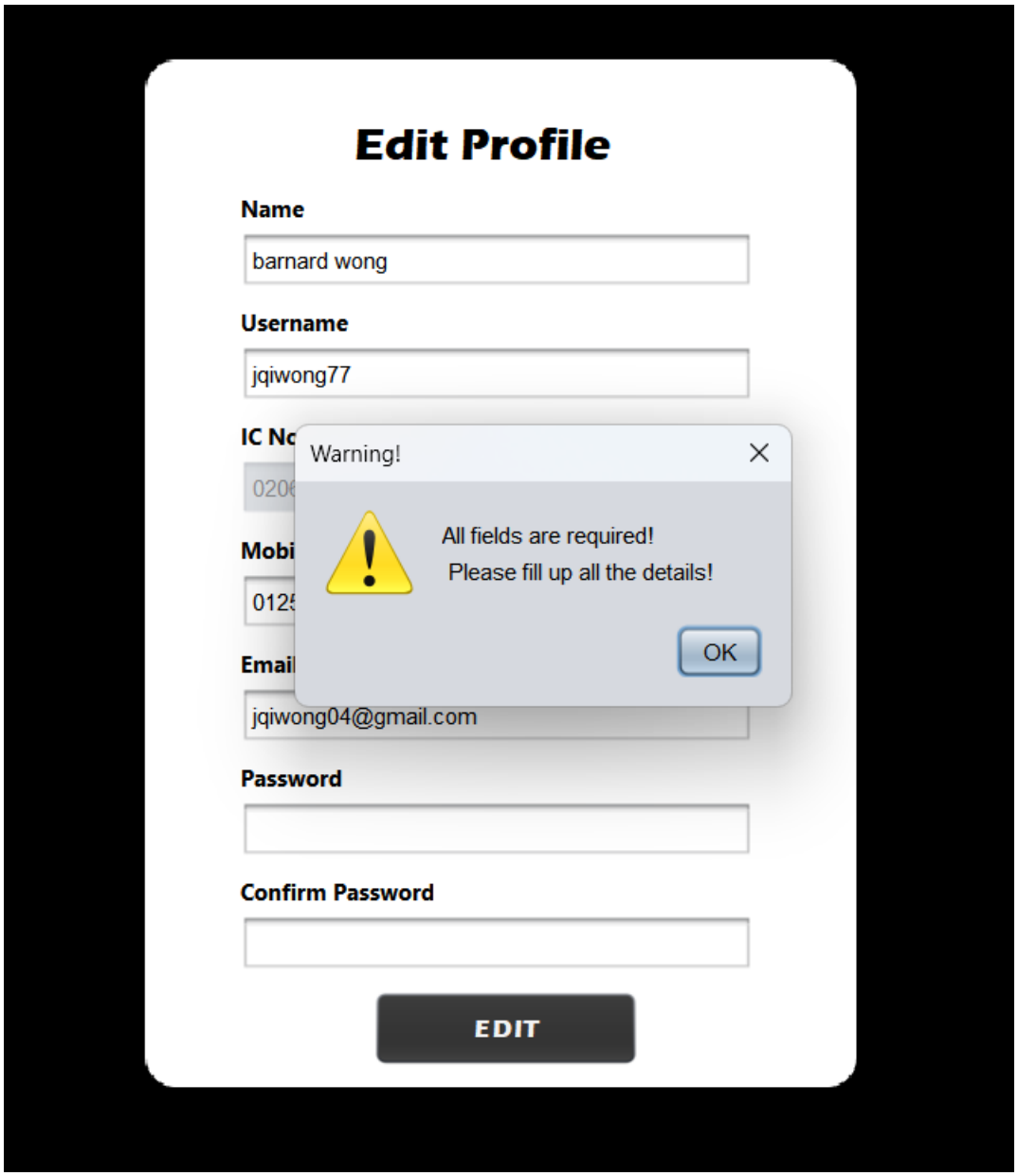


Figure 105 Edit Profile

This warning notice will show if the personal information on the Edit Profile page is incomplete. This notification instructs customers to enter all information.

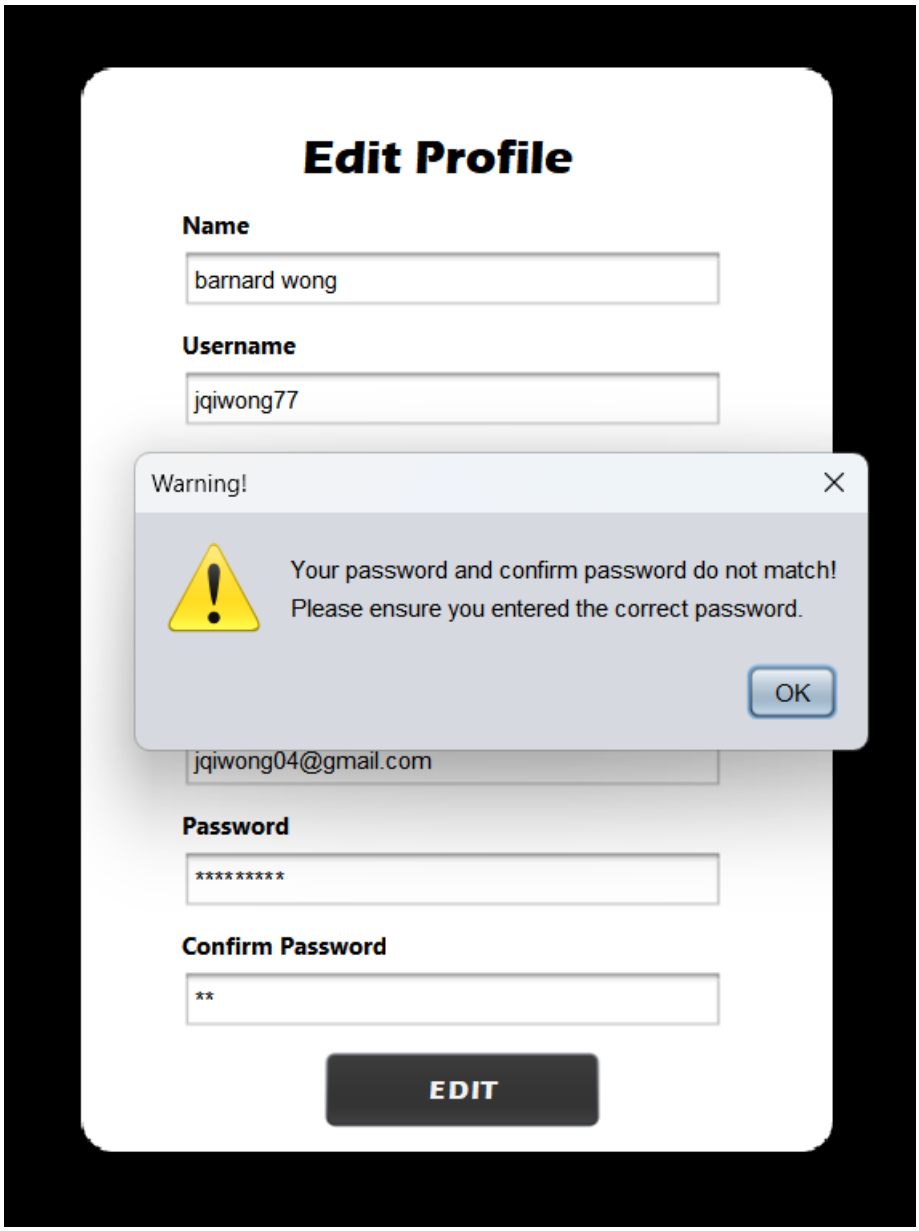


Figure 106 Edit Profile

This warning message will show if the password and confirm password used in the Edit Profile page do not match. This notification requests that customers give a valid password in order to prevent typing errors.

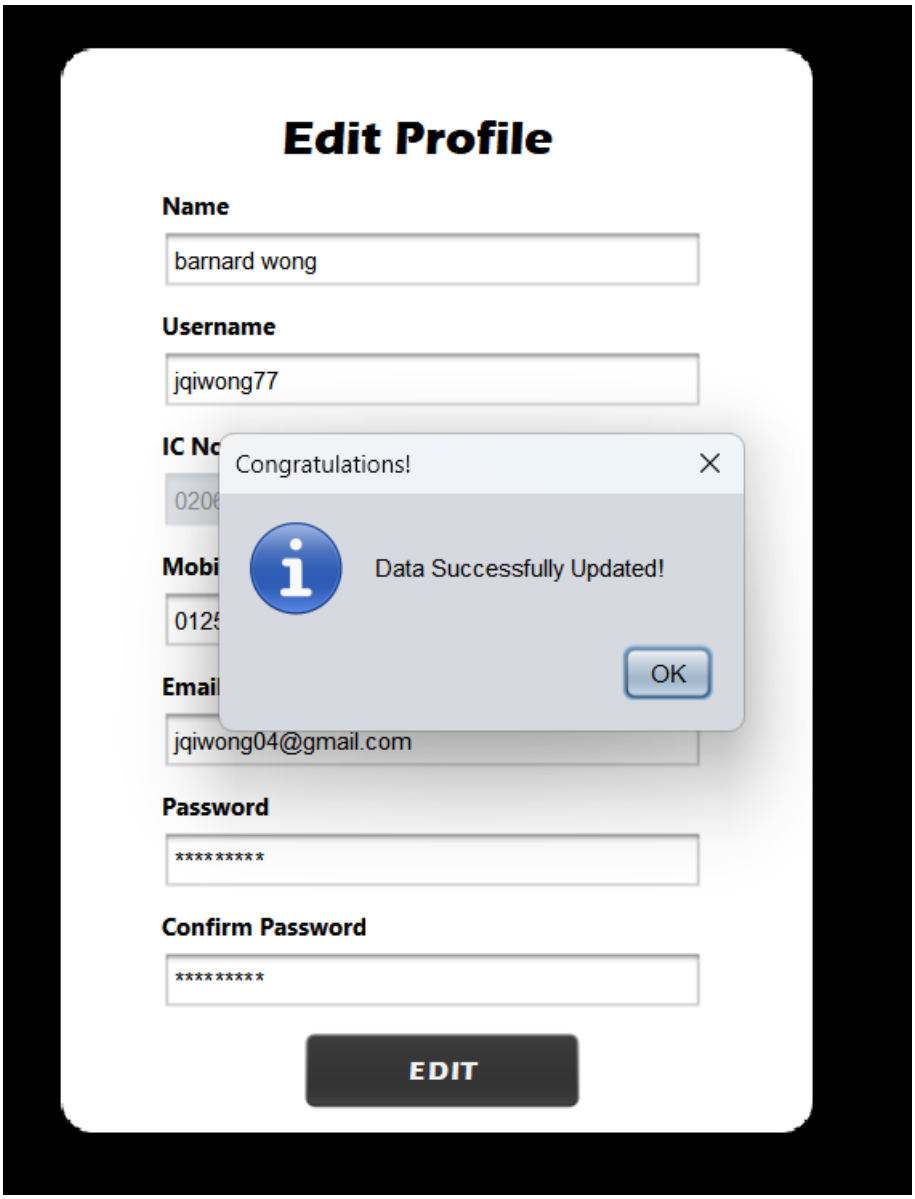


Figure 107 Edit Profile

After entering all the valid information, the system will update the information and show this message by pressing the “EDIT” button.

5.4 System Admin

5.4.1 Admin Login

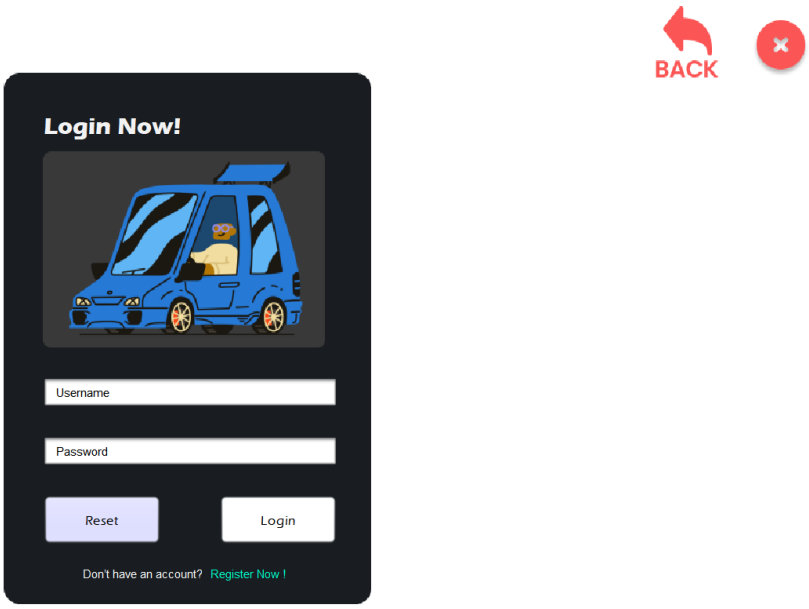


Figure 108 Admin Login

This page is for signing in. Admin needs to enter a valid login and password in order to access the system. The "BACK" button may be used to go back to the previous page. By clicking the "Reset" button, admin can erase their login details. To access the system and validate their username and password, admin must click the "Login" button. Admin will be sent to the “Admin Menu” page if the valid username and password are entered.

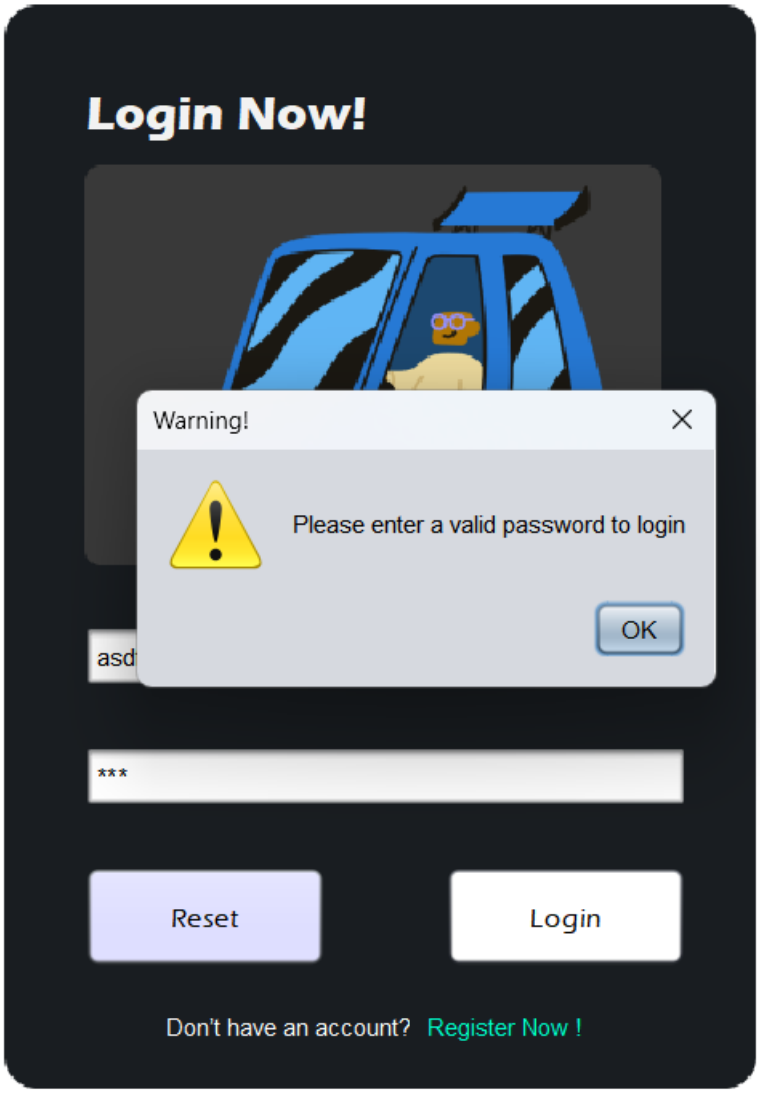


Figure 109 Admin Login

If admin forget to enter their password, this message will prompt them to do so.

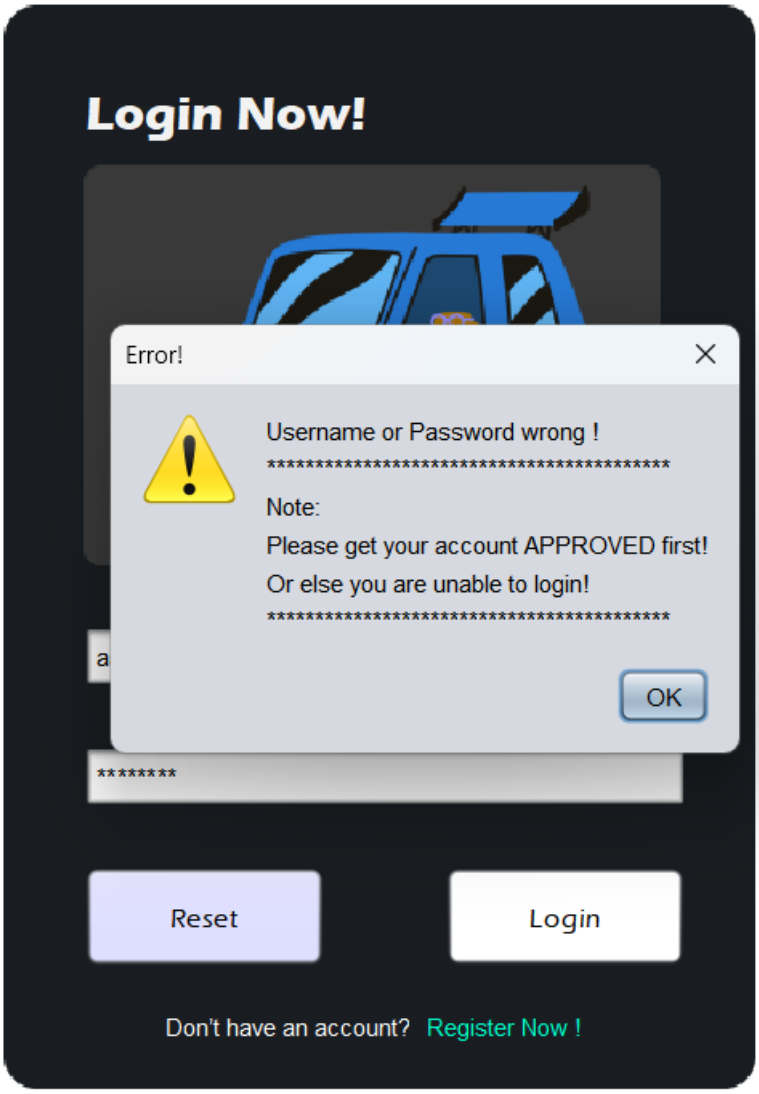


Figure 110 Admin Login

This message will show if the username or password invalid or the admin account not yet approved.

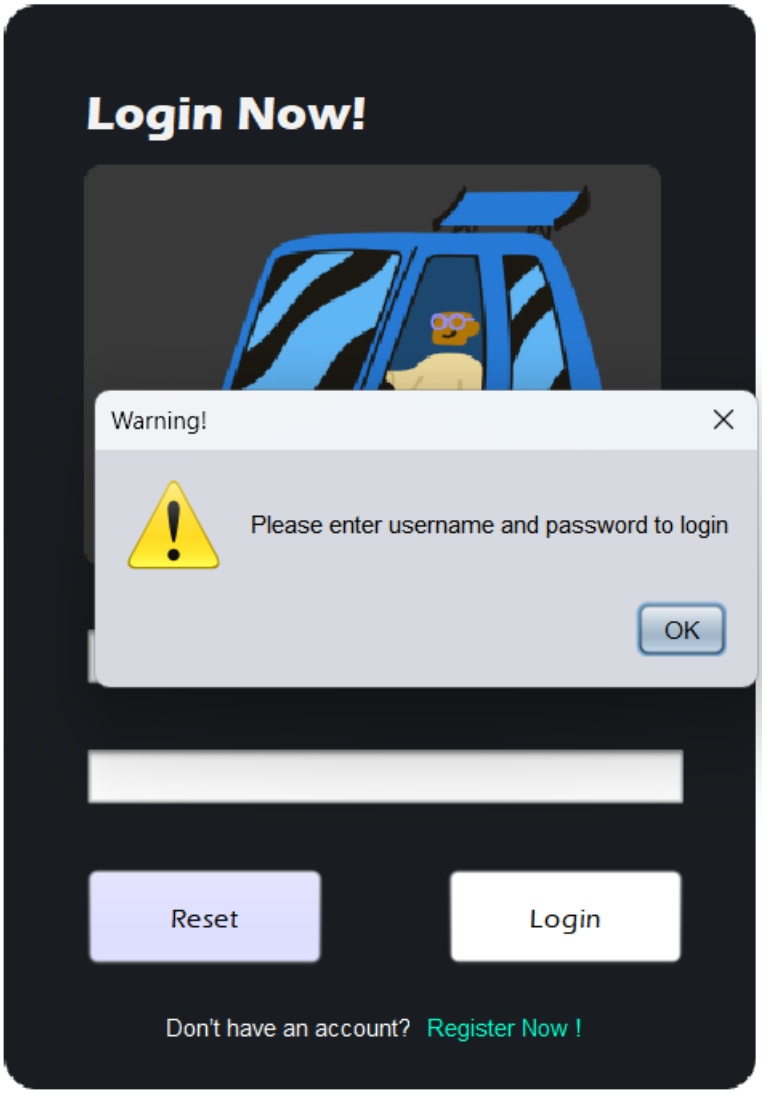


Figure 111 Admin Login

If the login information is incomplete, this notice will show and requesting that admin complete all of the login information.

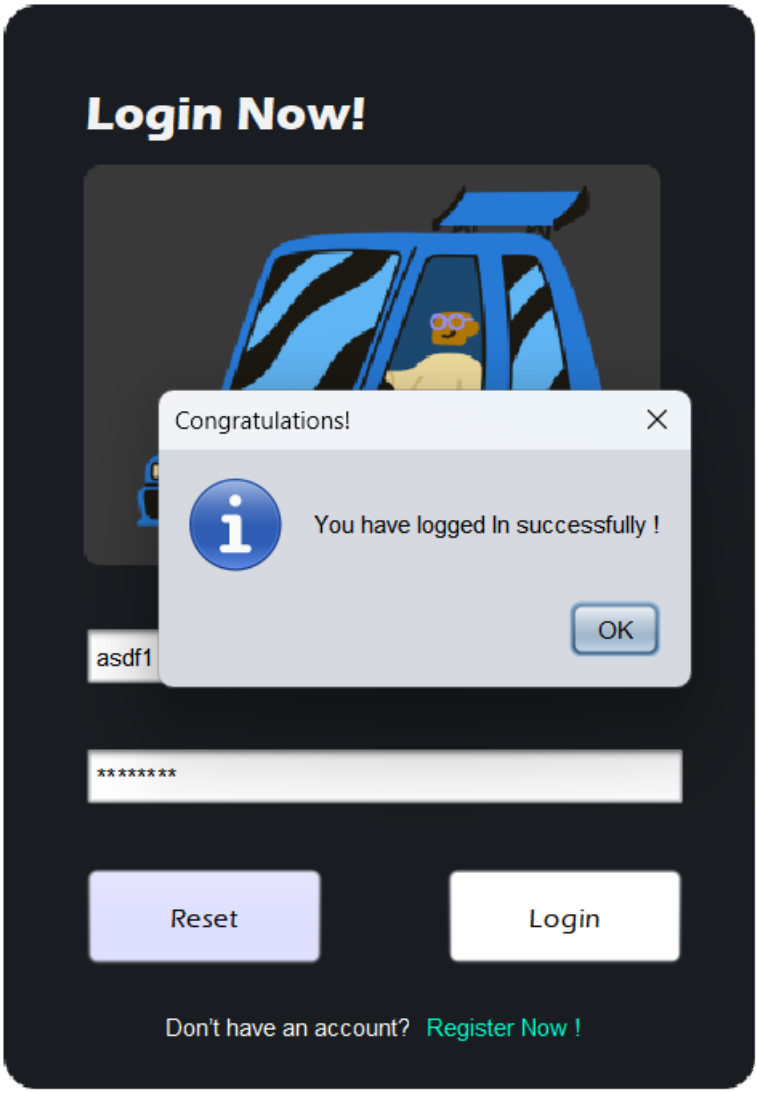


Figure 112 Admin Login

This message will appear if customers login or password is valid, click "OK" to proceed to the "Admin Menu" page.

5.4.2 Admin Registration

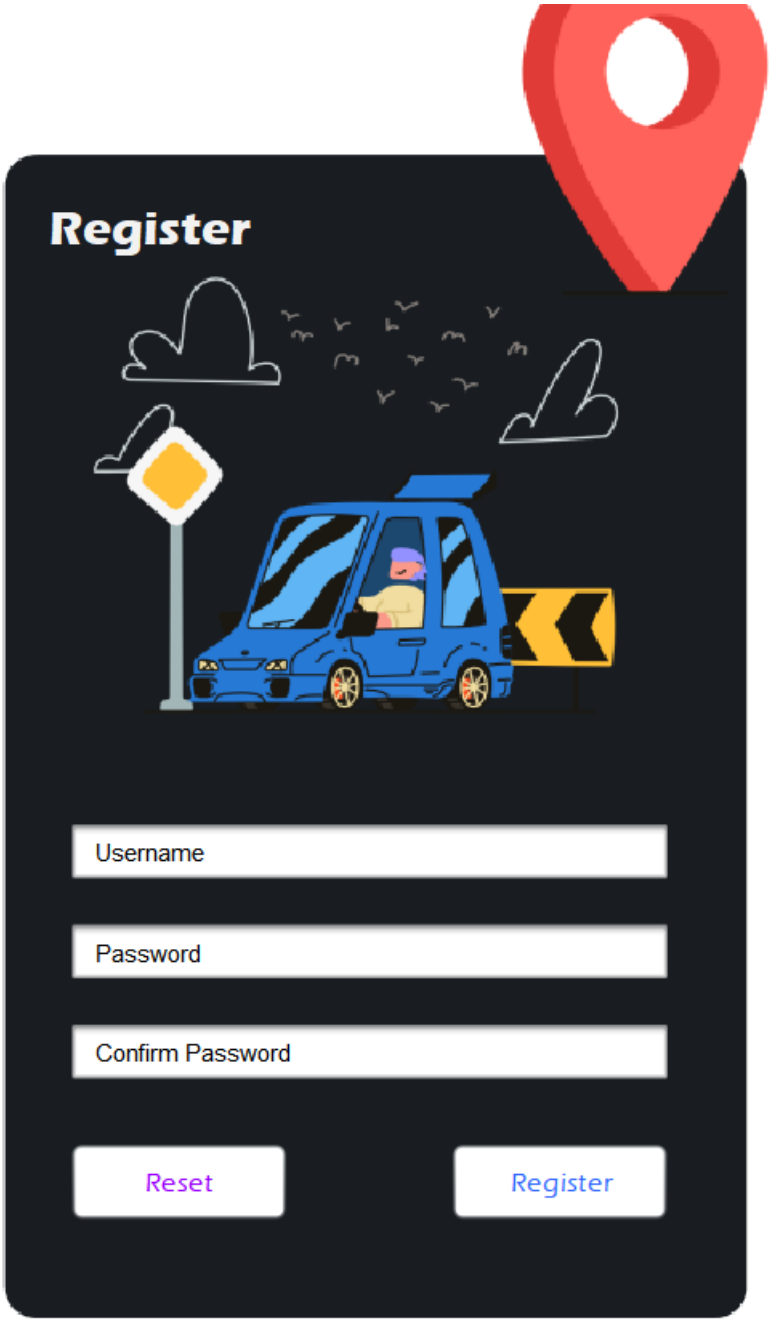


Figure 113 Admin Registration

This is the Registration page, where admin may create a new account by entering their username and password. Admin will be able to reset everything in the form by pressing the "Reset" button. Admin may return to the "Login page" by pressing the "Back" button. After entering all their information, admin must click the "Register" button to authenticate their

information, which is saved in the user.txt file. After registration, the new admin account need to wait for approved by default admin.

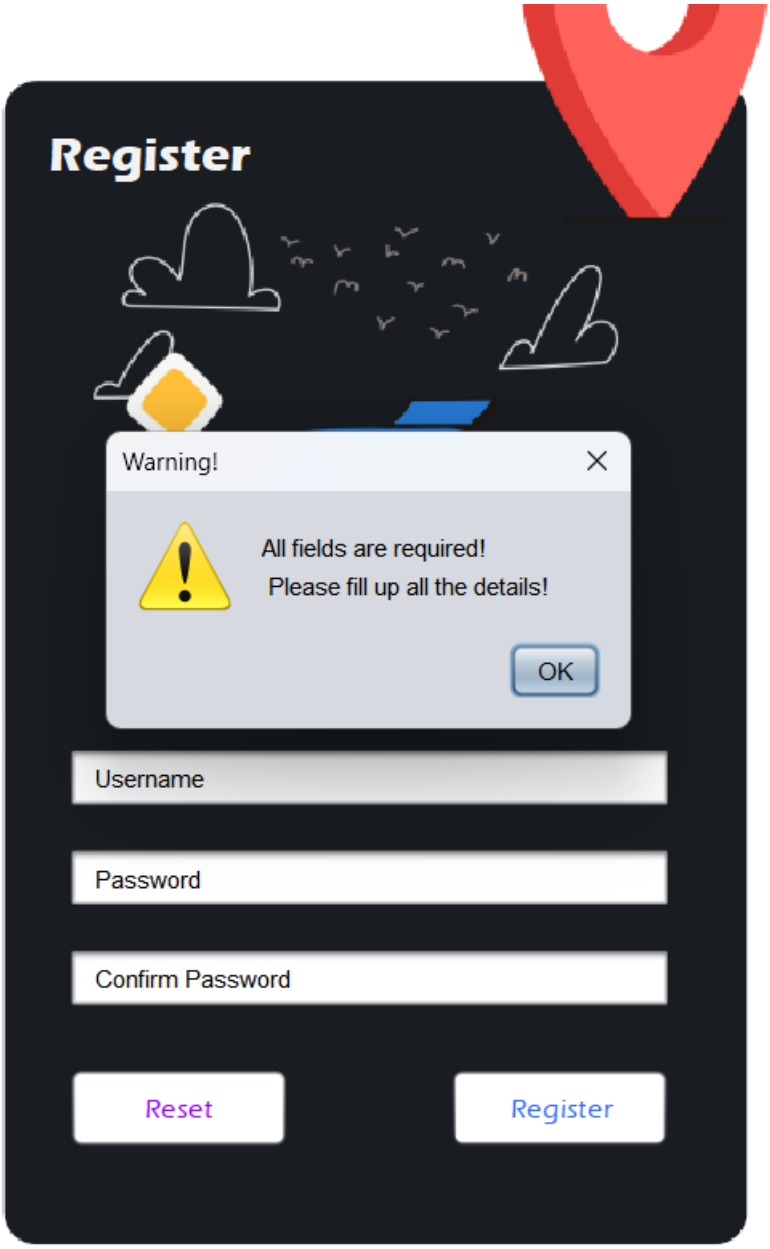


Figure 114 Admin Registration

This warning notice will show if the information on the Registration form is incomplete. This notification instructs admin to enter all information.

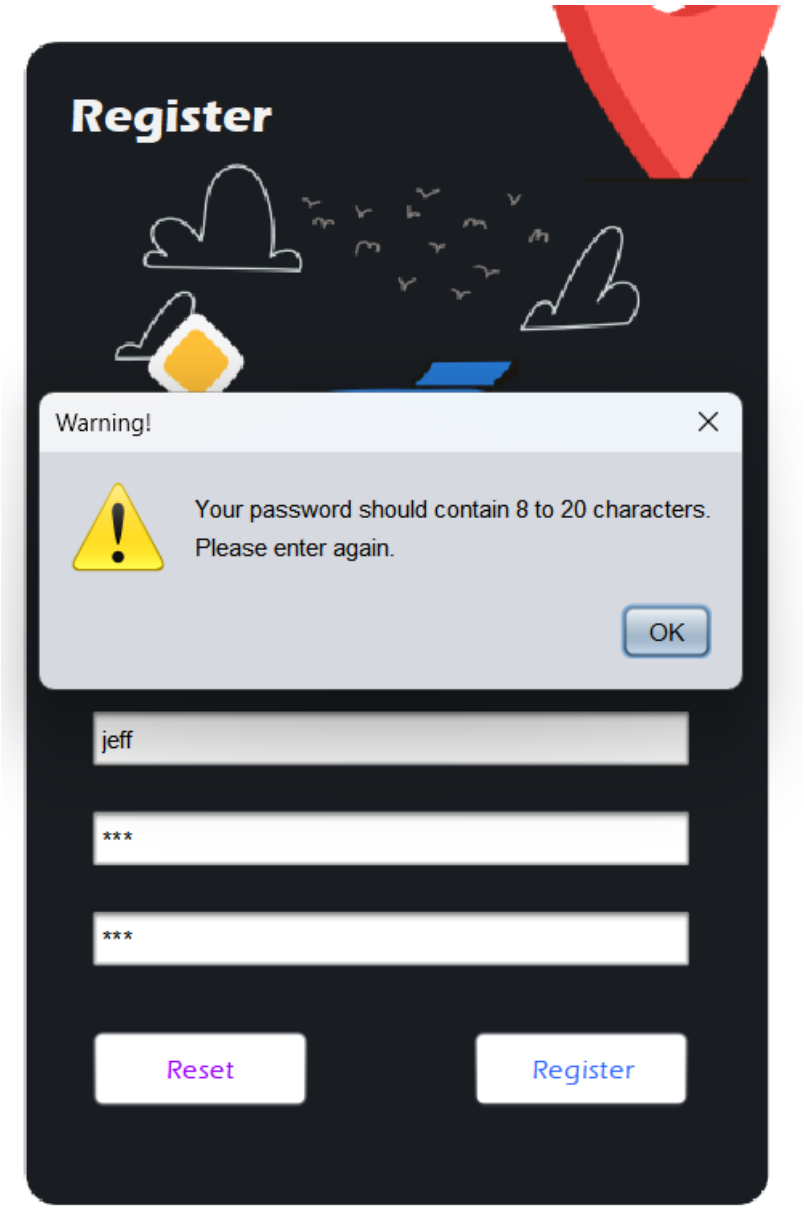


Figure 115 Admin Registration

If the password entered on the registration form does not include between 8 and 20 characters, a warning message will appear. This message demands a valid password from the admin.

Figure 116 Admin Registration

If the password and confirm password entered on the registration form do not match, this warning message will appear. To avoid typing mistakes, this notice asks the admin to provide a valid password.

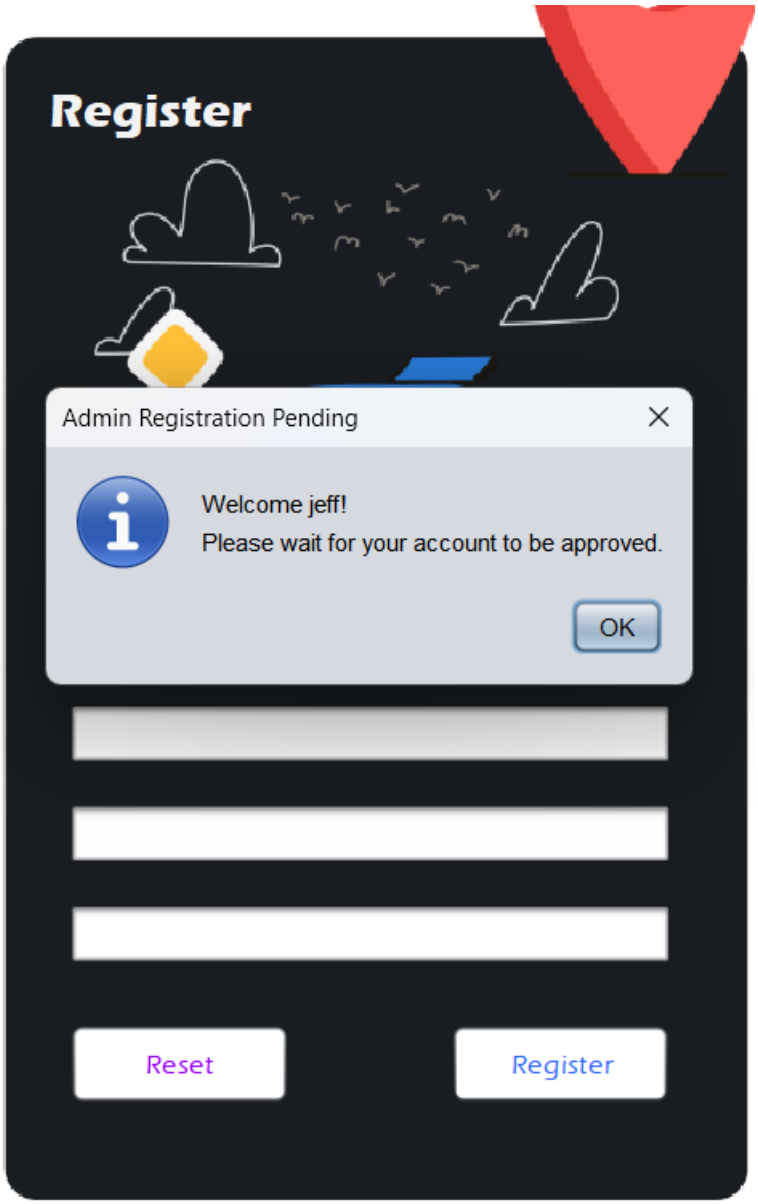


Figure 117 Admin Registration

This message will appear if admin information is valid, click "Register" to proceed to the "Login page."

5.4.3 Admin Menu

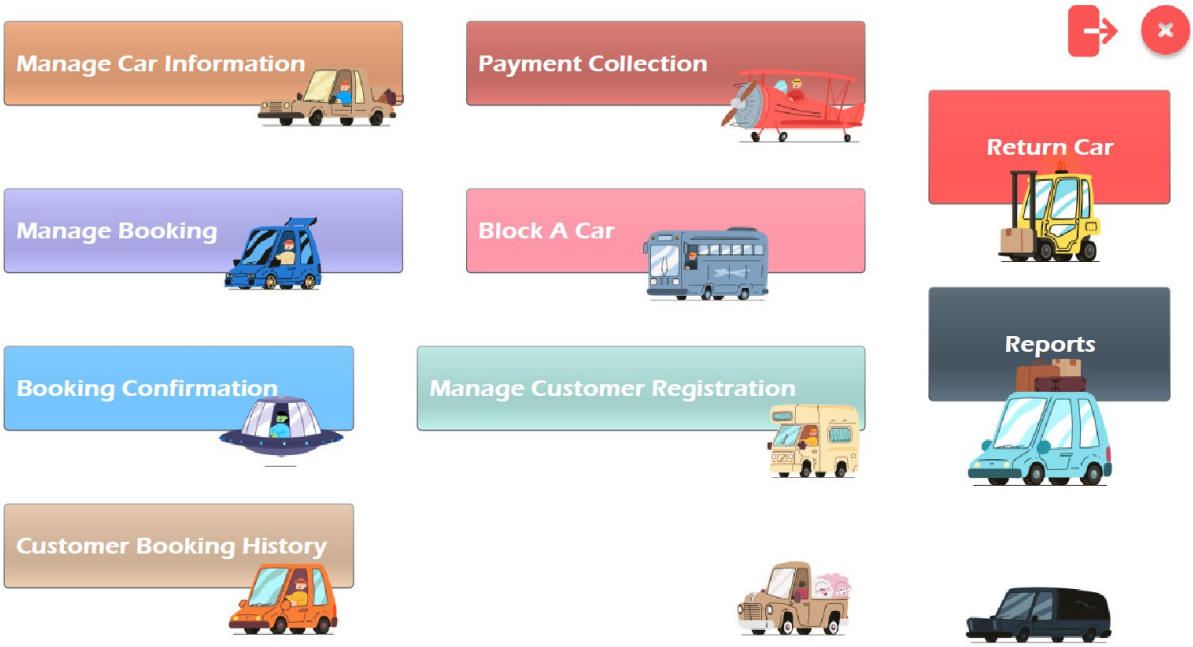


Figure 118 Admin Menu

The system's main menu may be found on the page titled "Admin Menu." Options are available to Manage Car Information, Manage Booking, Booking Confirmation, Customer Booking History, Payment Collection, Block A Car, Manage Customer Registration, Return Car, Reports. These features are directly accessible by system admin.

5.4.4 Manage Car Info

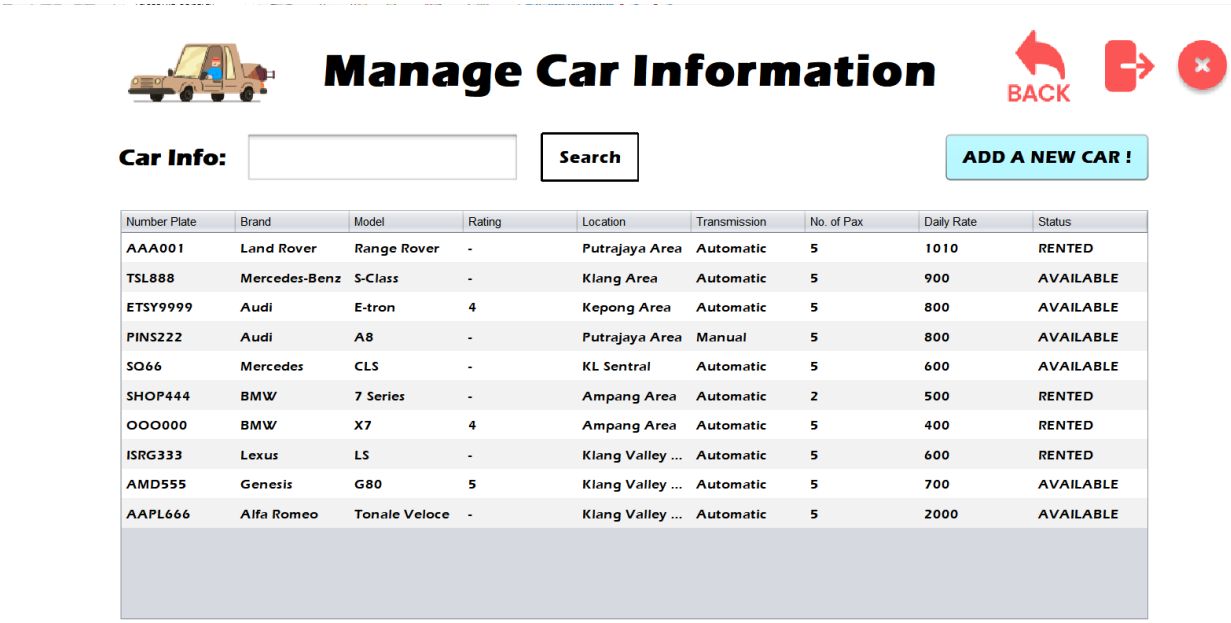


Figure 119 Manage Car Info

This page is for admin to manage car information where can add a new car, update, and delete the car.

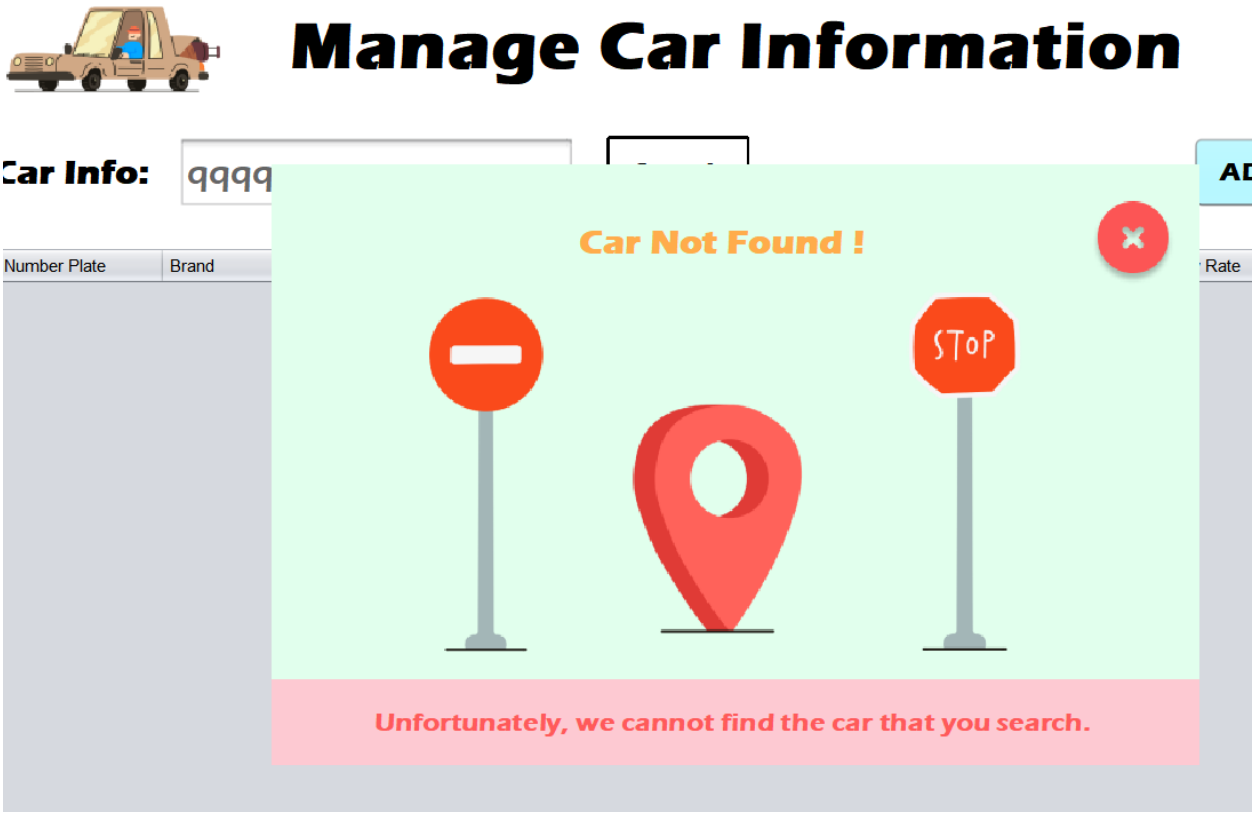


Figure 120 Manage Car Info

This message will show if there are no car found according to the search.

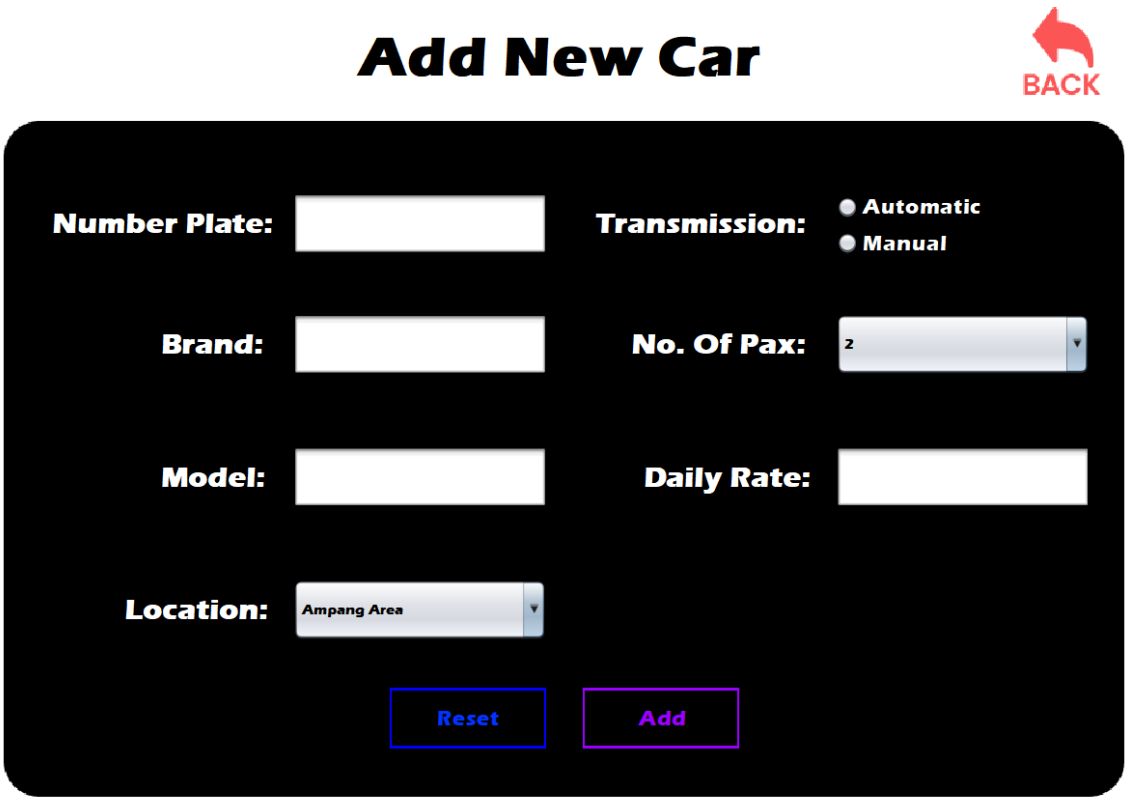


Figure 121 Manage Car Info

This page is for admin to add a new car by enter all the info and click the “ADD” button.

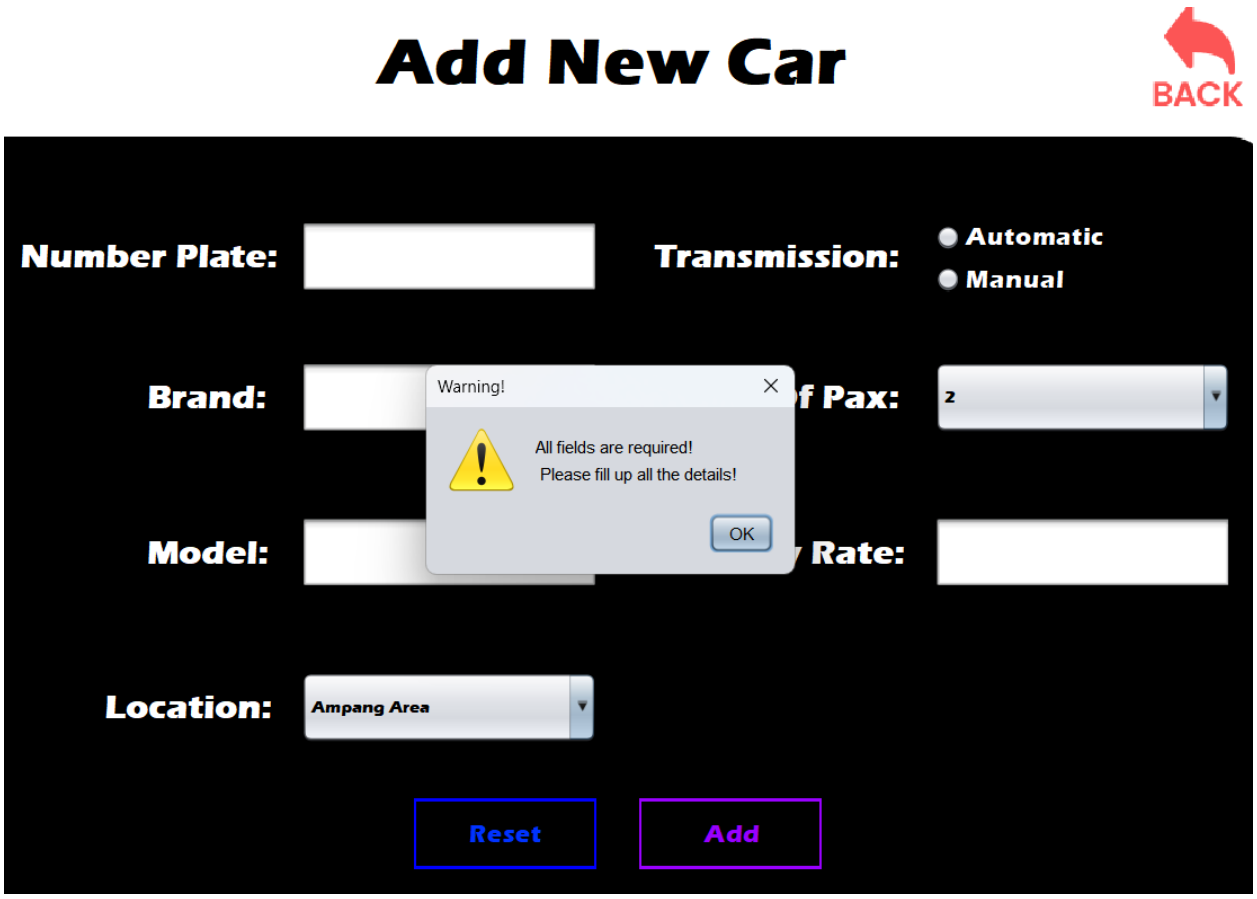


Figure 122 Manage Car Info

This warning notice will show if the information on the “Add New Car” page such as number plate, brand, model, location, transmission, number of pax and daily rate is incomplete. This notification instructs admin to enter all information.

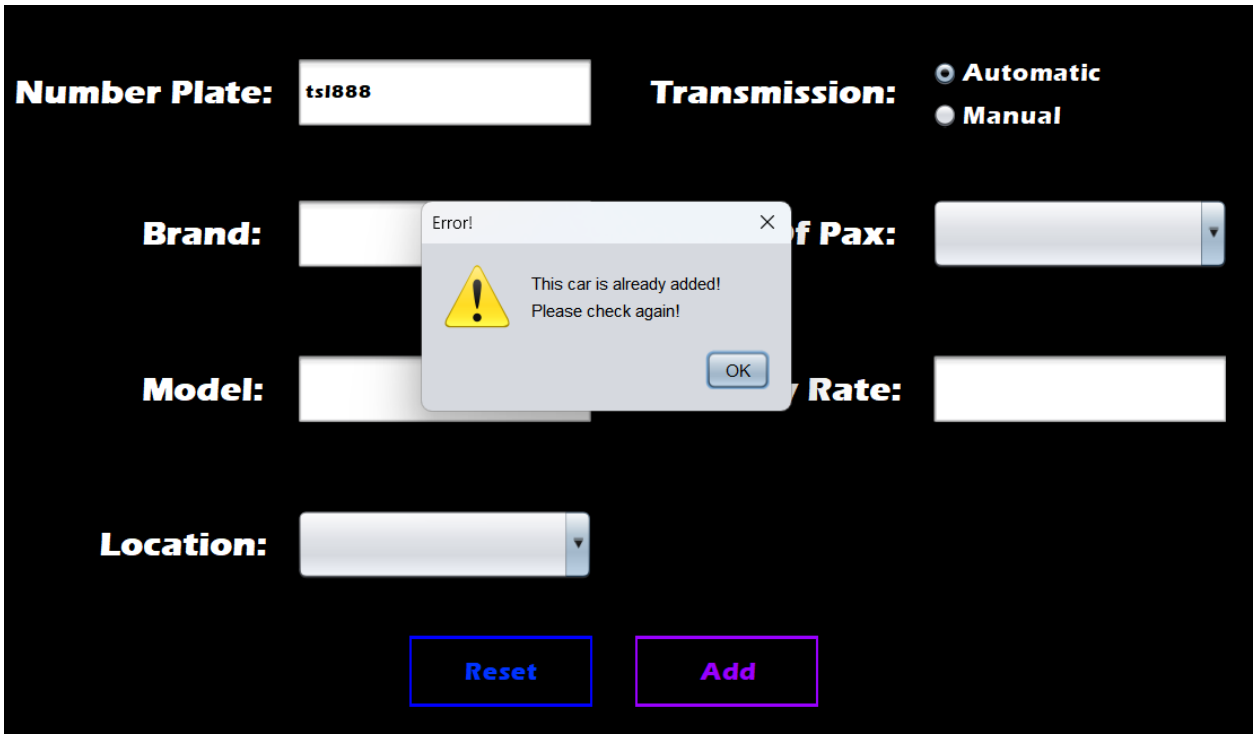


Figure 123 Manage Car Info

This message will show if the car number plate already added into the car.txt file.

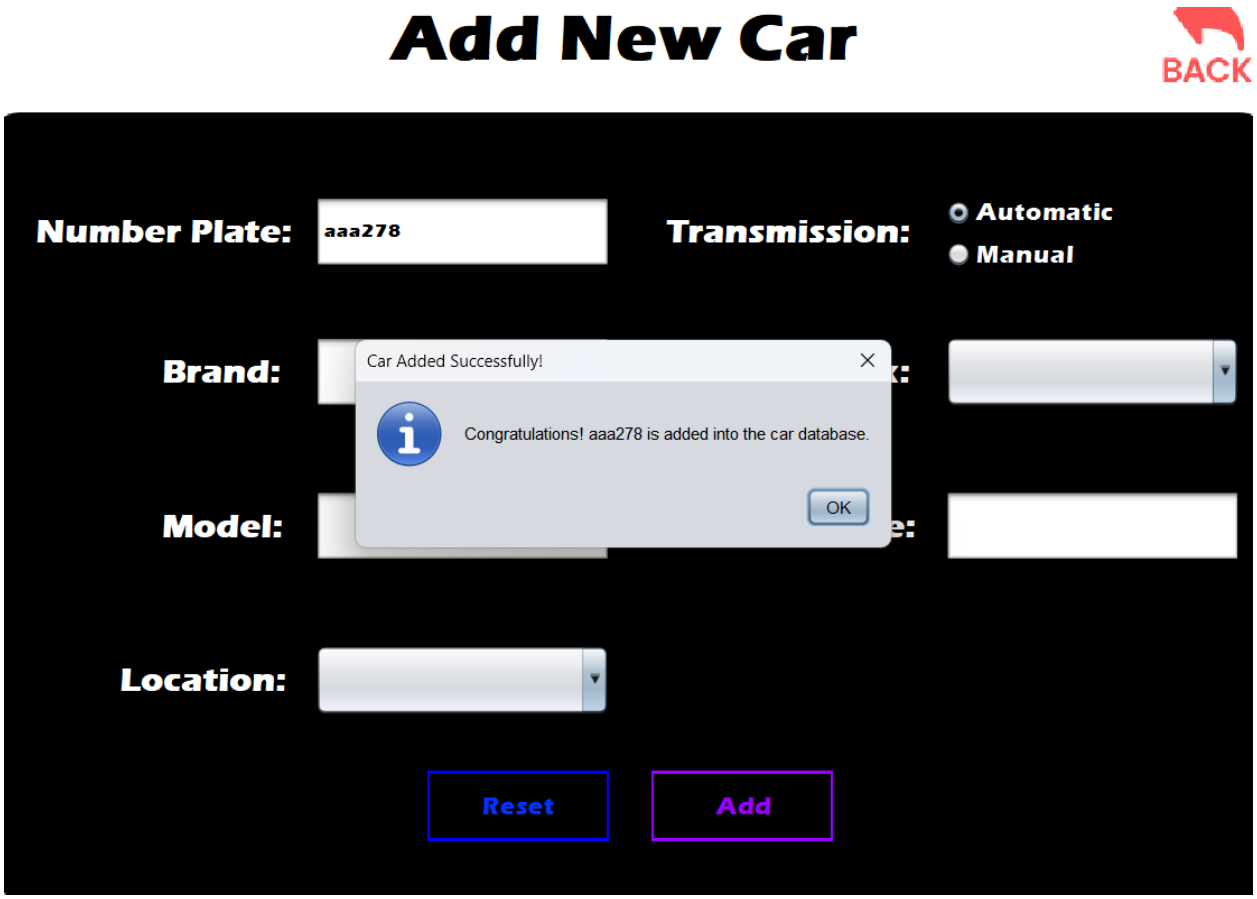


Figure 124 Manage Car Info

After admin enter all the valid info and click the "Add" button, the system will show this message and save the car details into car.txt file.

```
aaa001//Land Rover//Range Rover//-//Putrajaya Area//1010//Automatic//5//RENTED
tsl888//Mercedes-Benz//S-Class//4//Klang Area//900//Automatic//5//RENTED
etsy9999//Audi//E-tron//4//Kepong Area//800//Automatic//5//RENTED
pins222//Audi//A6//-//Putrajaya Area//700//Manual//5//AVAILABLE
sq66//Mercedes//CLS//-//KL Sentral//600//Automatic//5//AVAILABLE
shop444//BMW//7 Series//-//Ampang Area//500//Automatic//2//RENTED
ooo000//BMW//X7//4//Ampang Area//400//Automatic//5//RENTED
isrg333//Lexus//LS//-//Pandan Indah Area//600//Automatic//5//AVAILABLE
amd555//Genesis//G80//5//Klang Valley Area//700//Automatic//5//BLOCKED
aapl666//Alfa Romeo//Tonale Veloce//-//Klang Valley Area//2000//Automatic//5//AVAILABLE
```

Figure 125 Manage Car Info

After added the car, all the car details will save into the car.txt file.

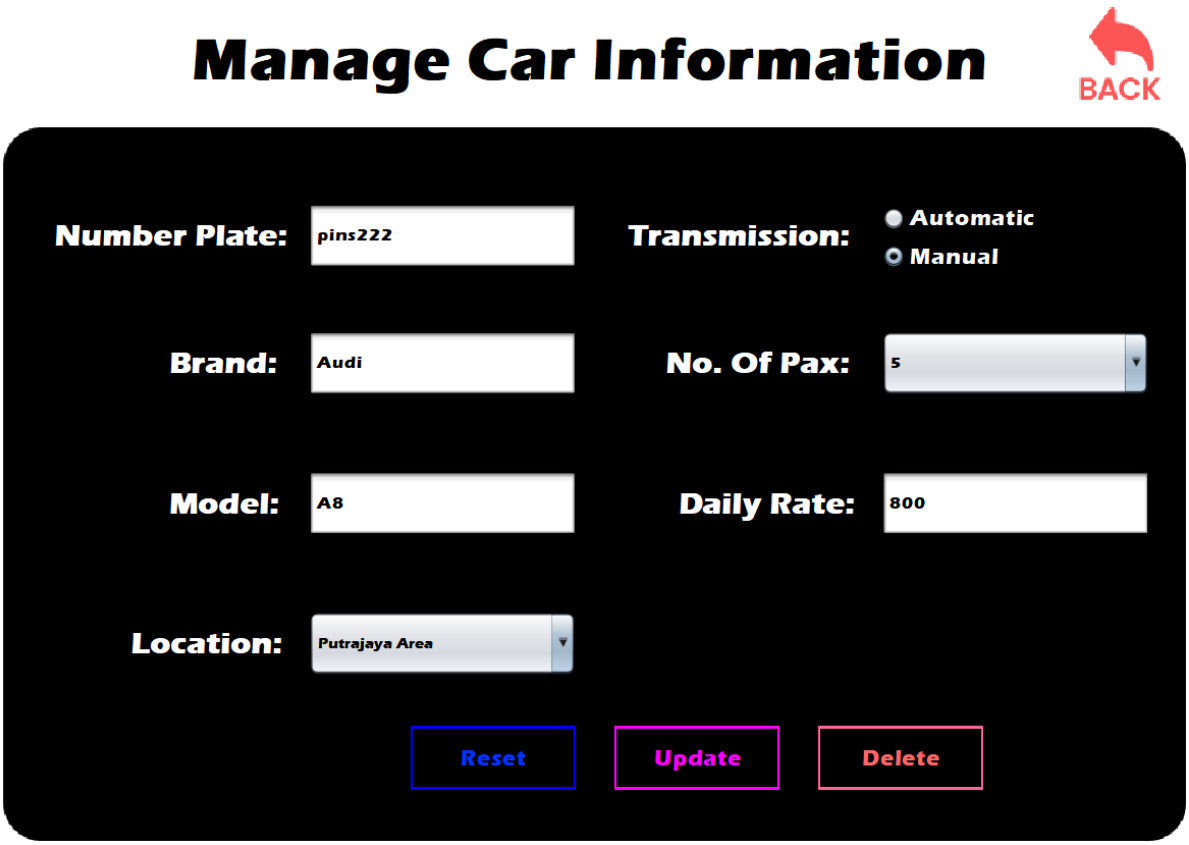


Figure 126 Manage Car Info

This page is for admin to update or delete the car by pressing on the “Update” button to update the car info or pressing on the “Delete” button to delete the car.

Update

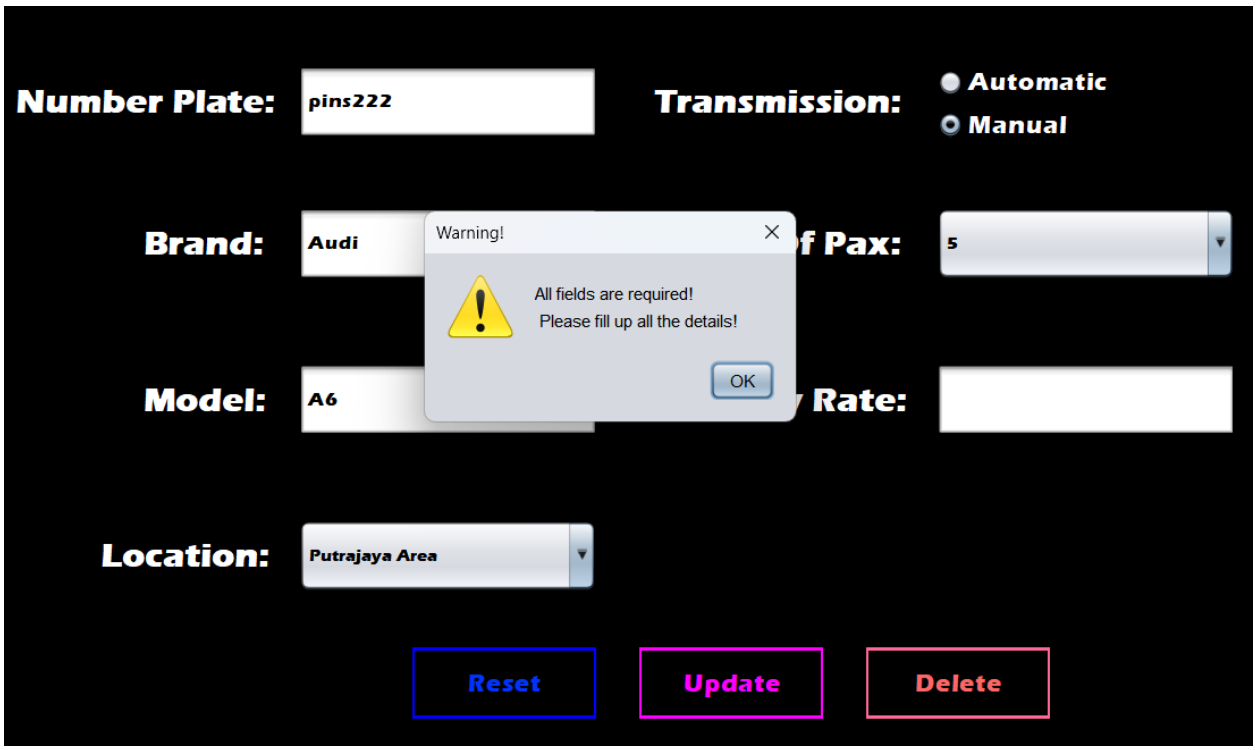


Figure 127 Manage Car Info

This warning notice will show if the car information on the “Manage Car” page is incomplete. This notification instructs admin to enter all information.

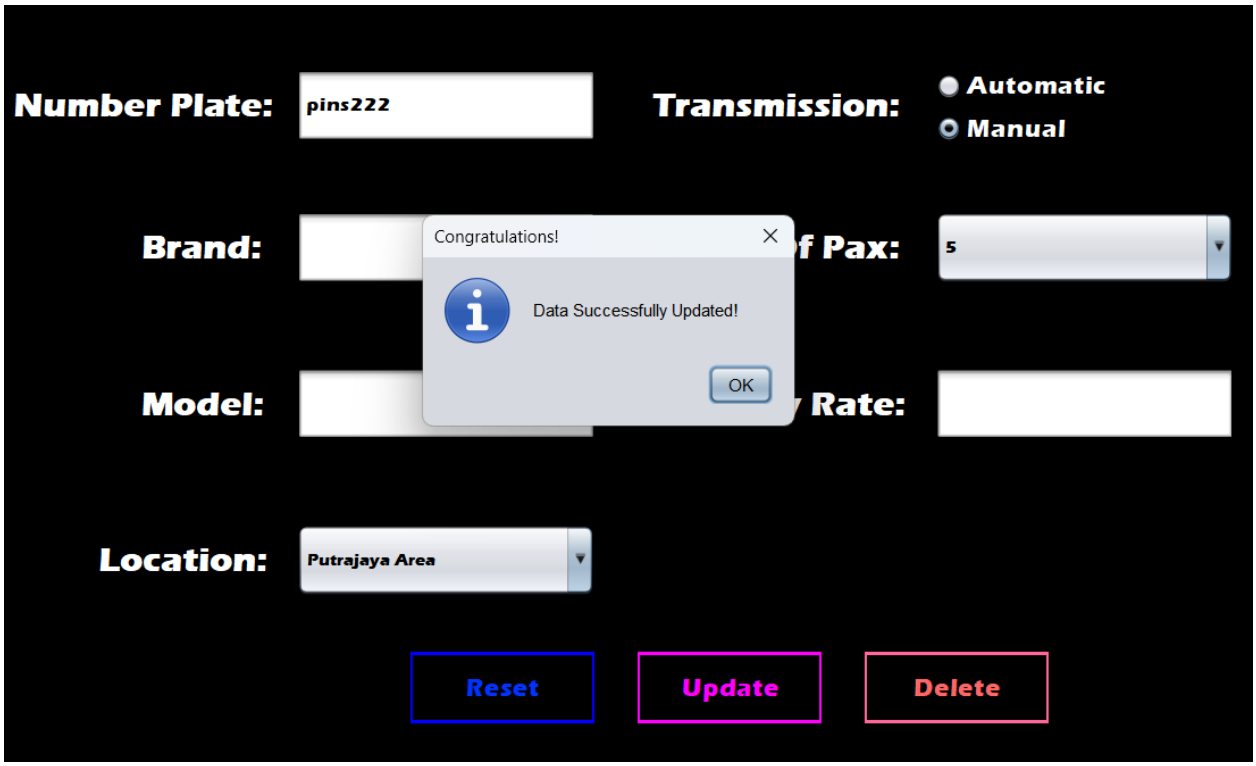


Figure 128 Manage Car Info

The system will display this message when admin have entered all the car information and clicked "Update."

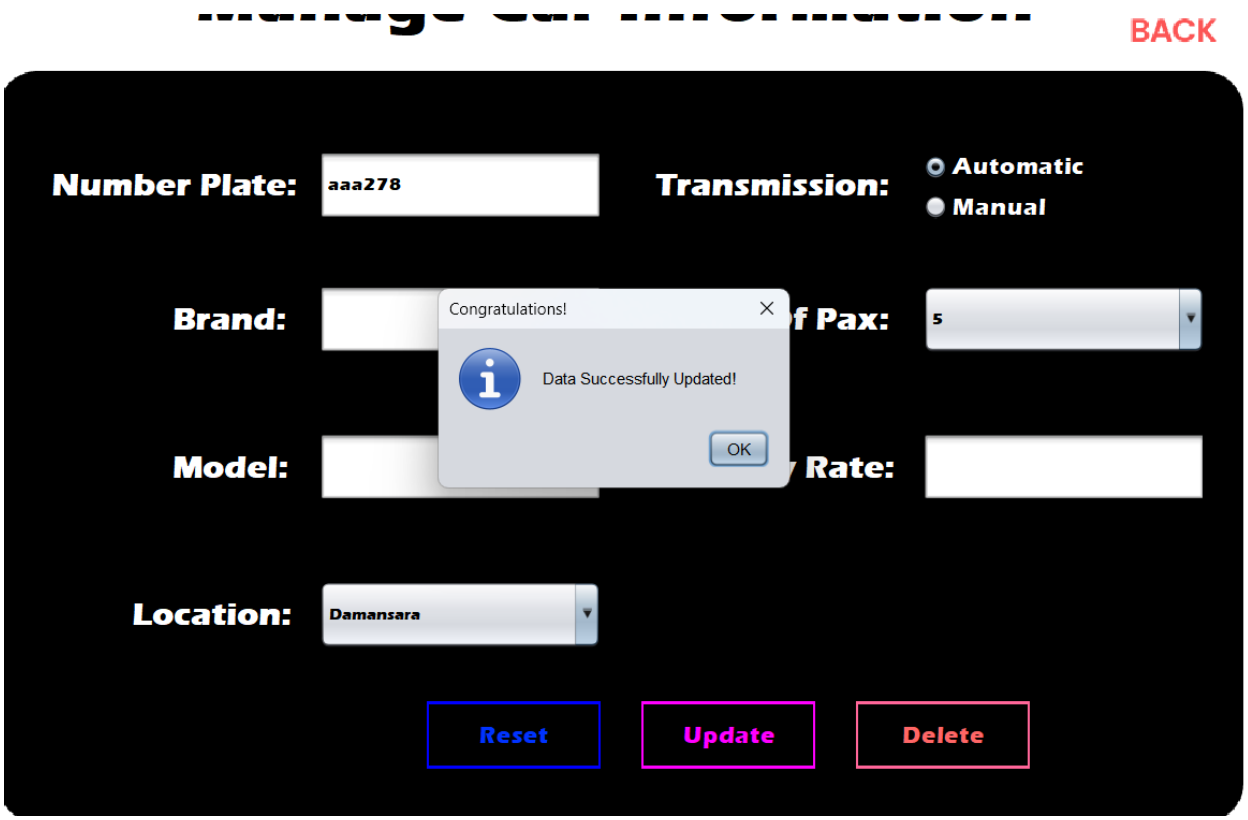


Figure 129 Manage Car Info

The system will display this notice and remove the car from the car.txt file according to the car's number plate once you click the "Delete" button.

5.4.5 Manage Booking

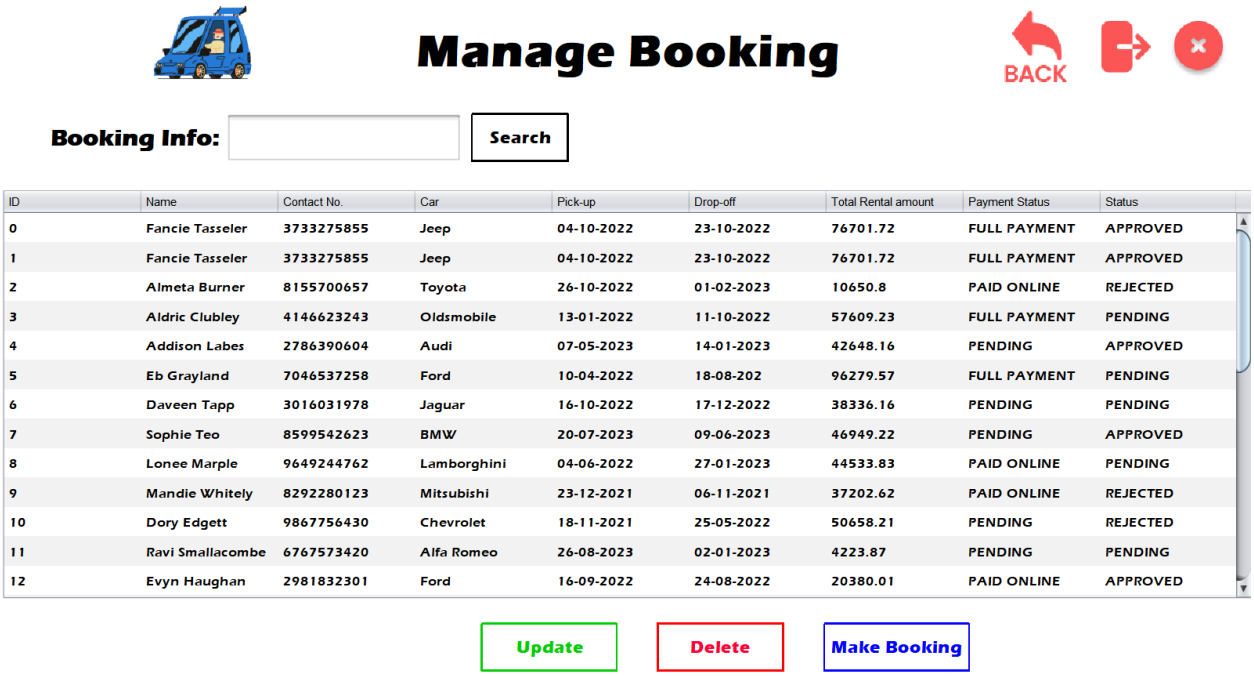


Figure 130 Manage Booking

This is “Manage Booking” page where for admin to check all customers bookings. Furthermore, admin able to select the booking which inside the table to update and delete the booking. Besides, there have one more button which is “Make Booking” button for admin to help customers to make booking.

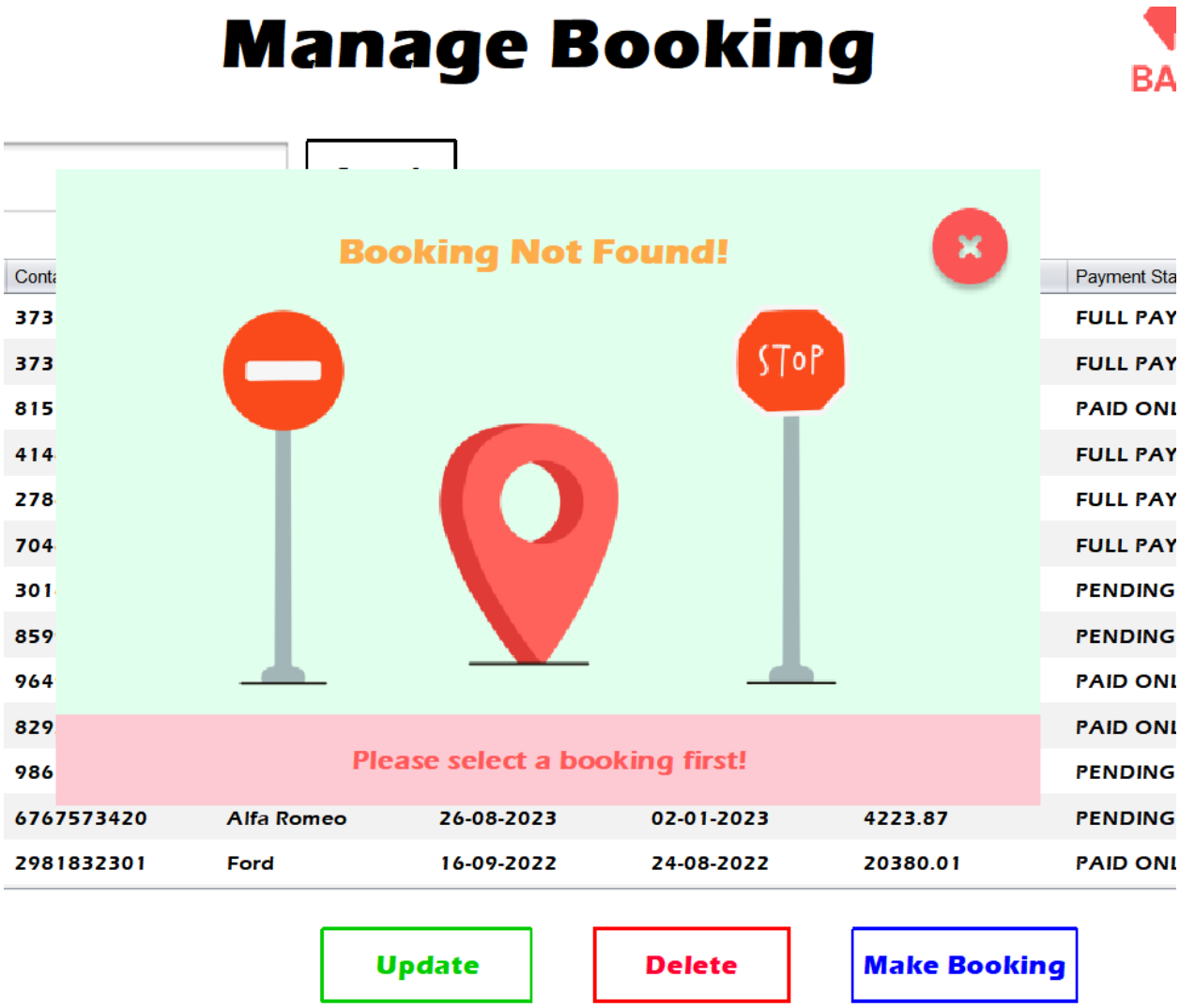


Figure 131 Manage Booking

This message will pop up if admin click on “Update” or “Delete” button without select any booking.

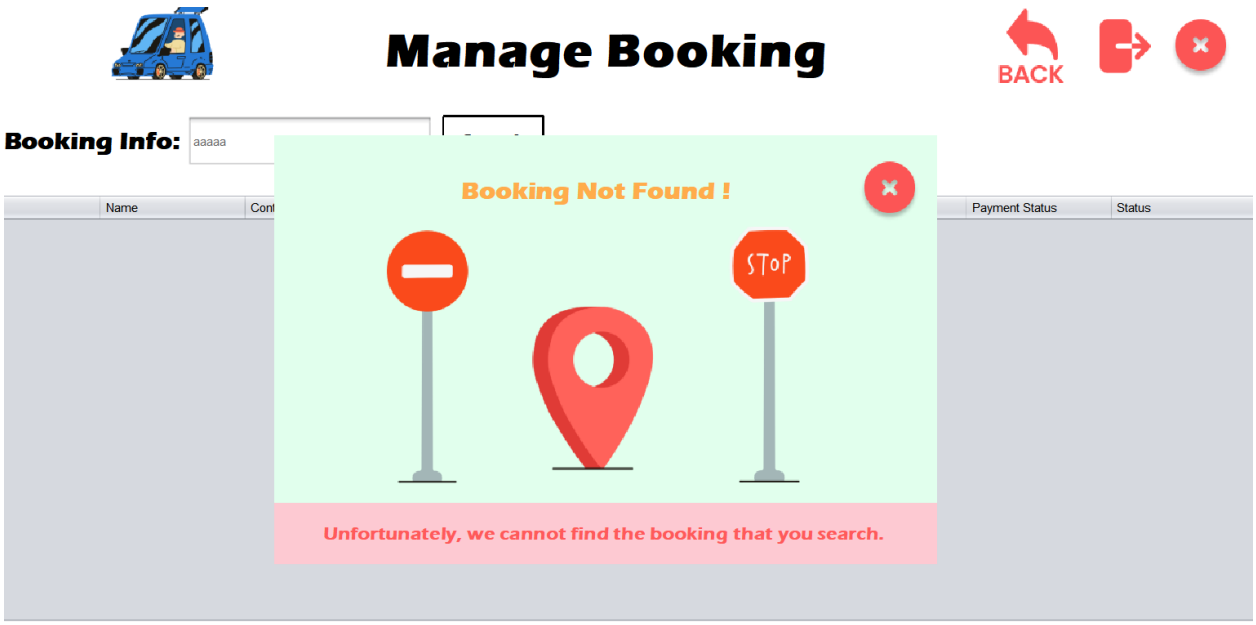


Figure 132 Manage Booking

This message will pop up if there are no booking found for the admin by pressing the search button to search the booking. Admin also can check all the booking by clicking the “Search” button without key in the booking info.

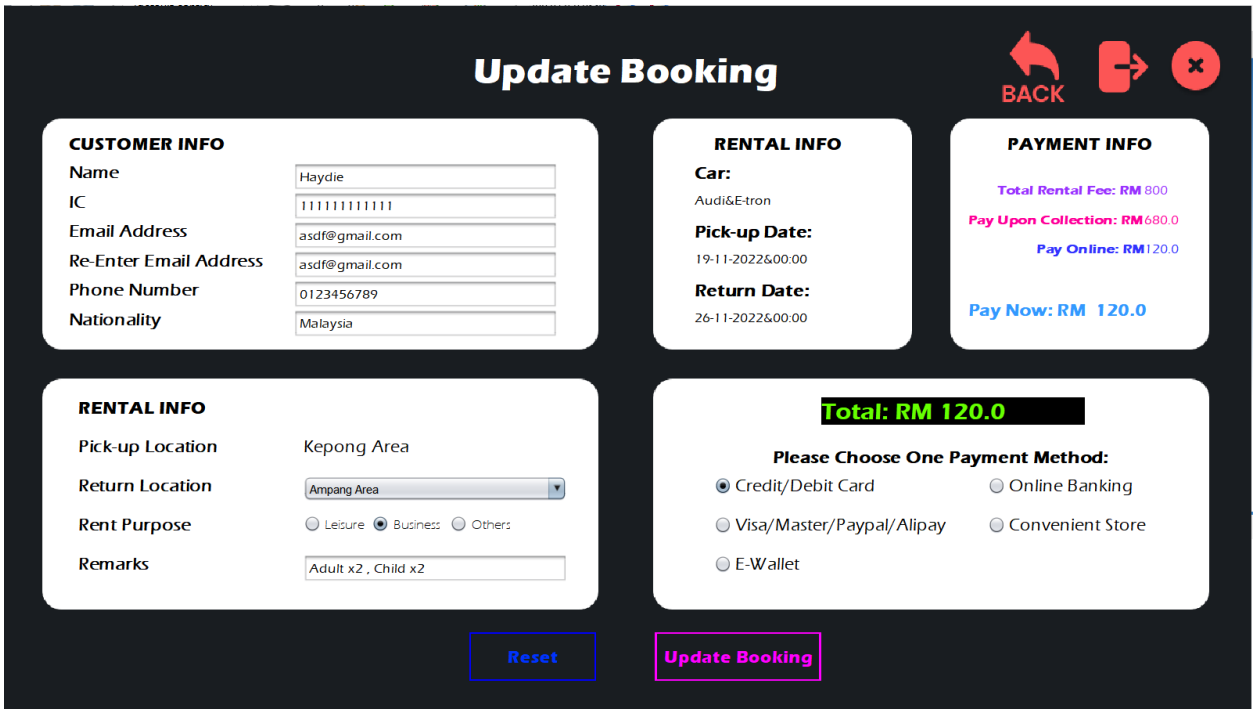


Figure 133 Manage Booking

This is “Update Booking” page where is for admin to modify the booking info.

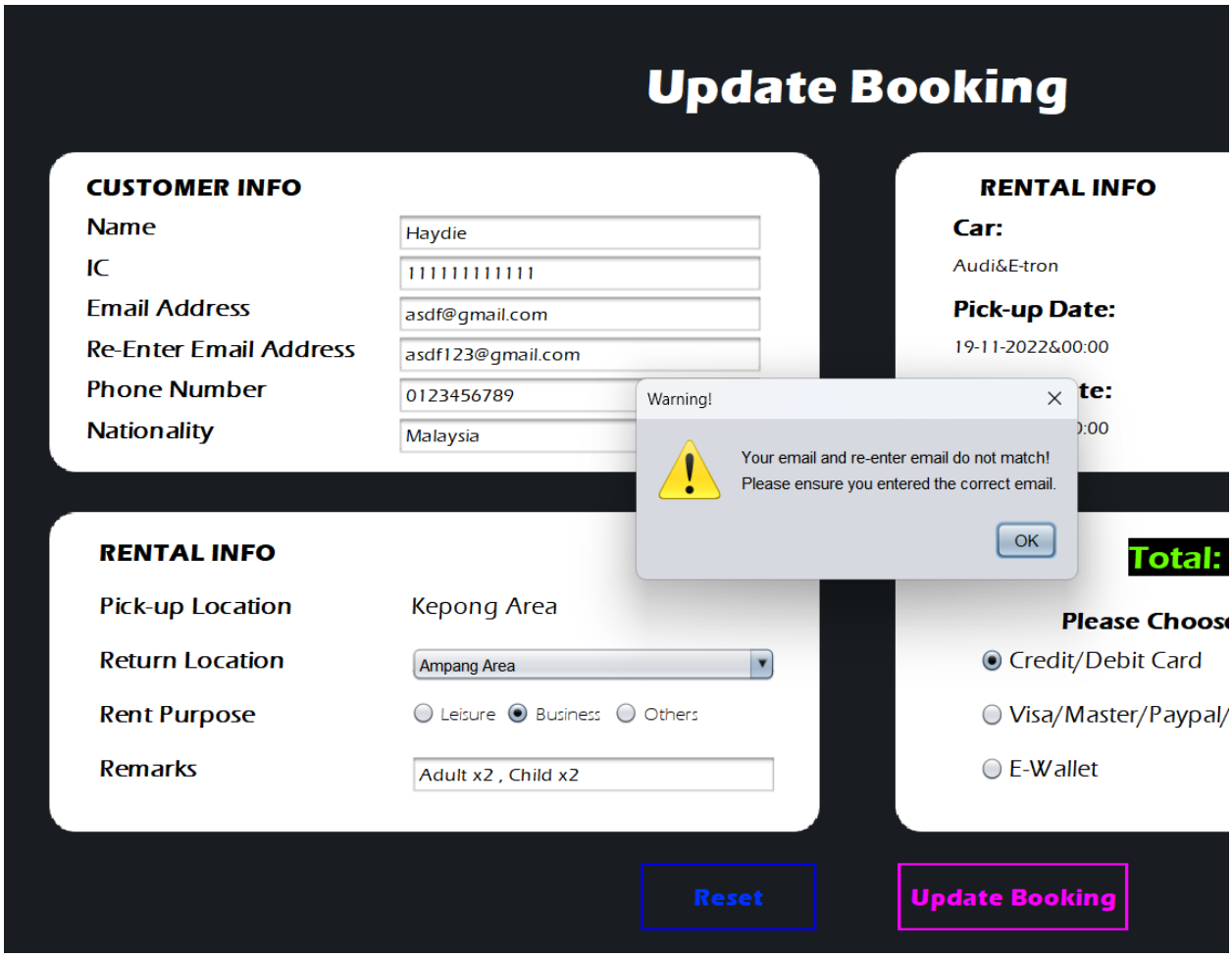


Figure 134 Manage Booking

This message will show if the Email Address is different with the Re-Enter Email Address in order to remind admin to check the email.

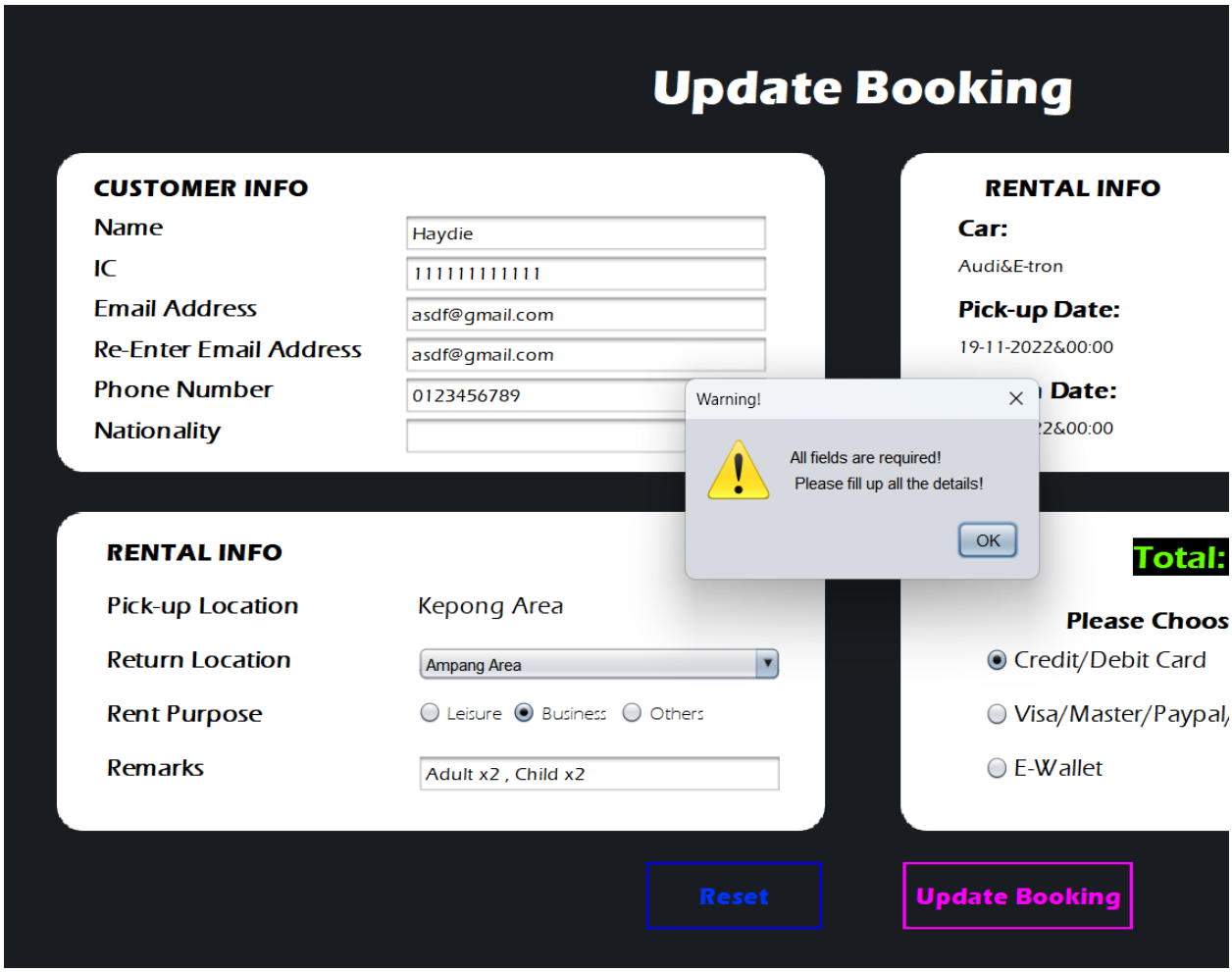


Figure 135 Manage Booking

This warning notice will show if the customer information on the Update Booking page is incomplete. This notification instructs admin to enter all information.

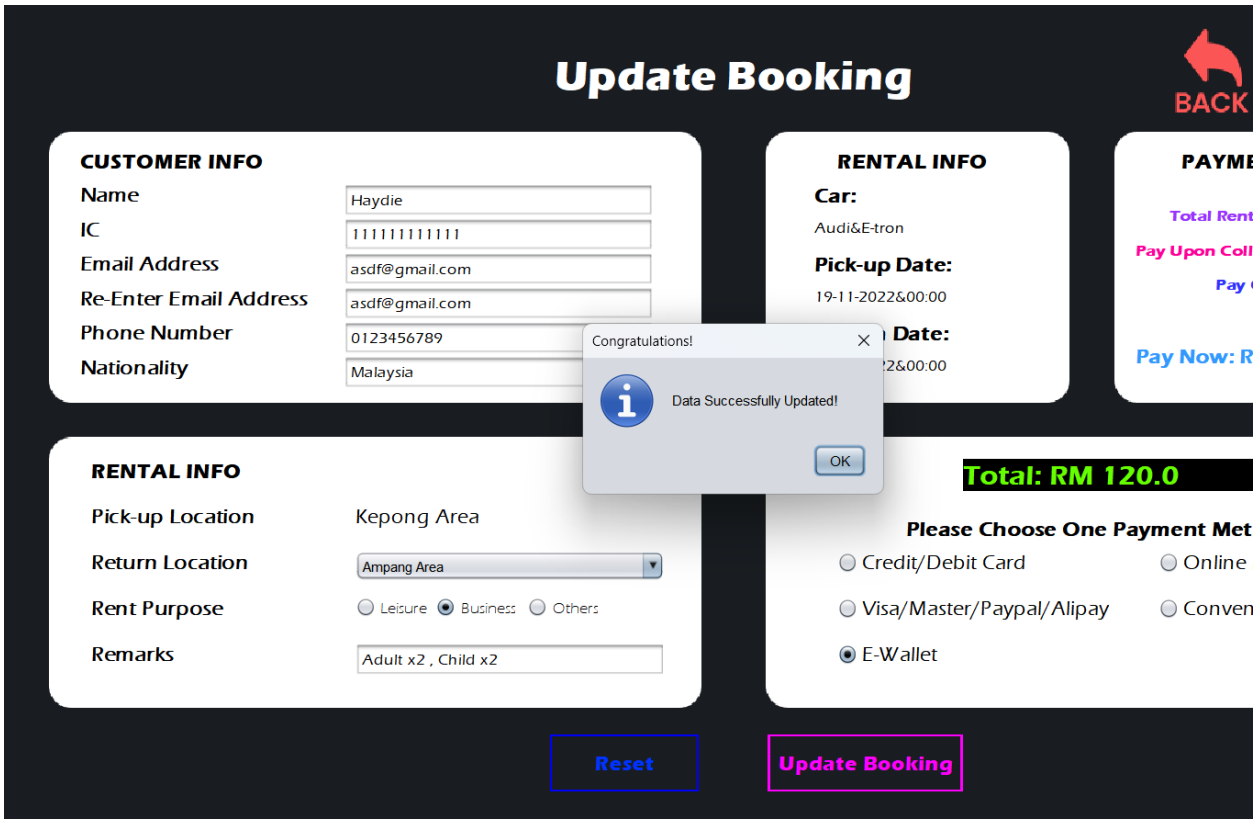


Figure 136 Manage Booking

This message will show when admin input all the valid information and click “Update Booking” button. The updated booking no need to wait for admin to approve because the booking are made by admin.

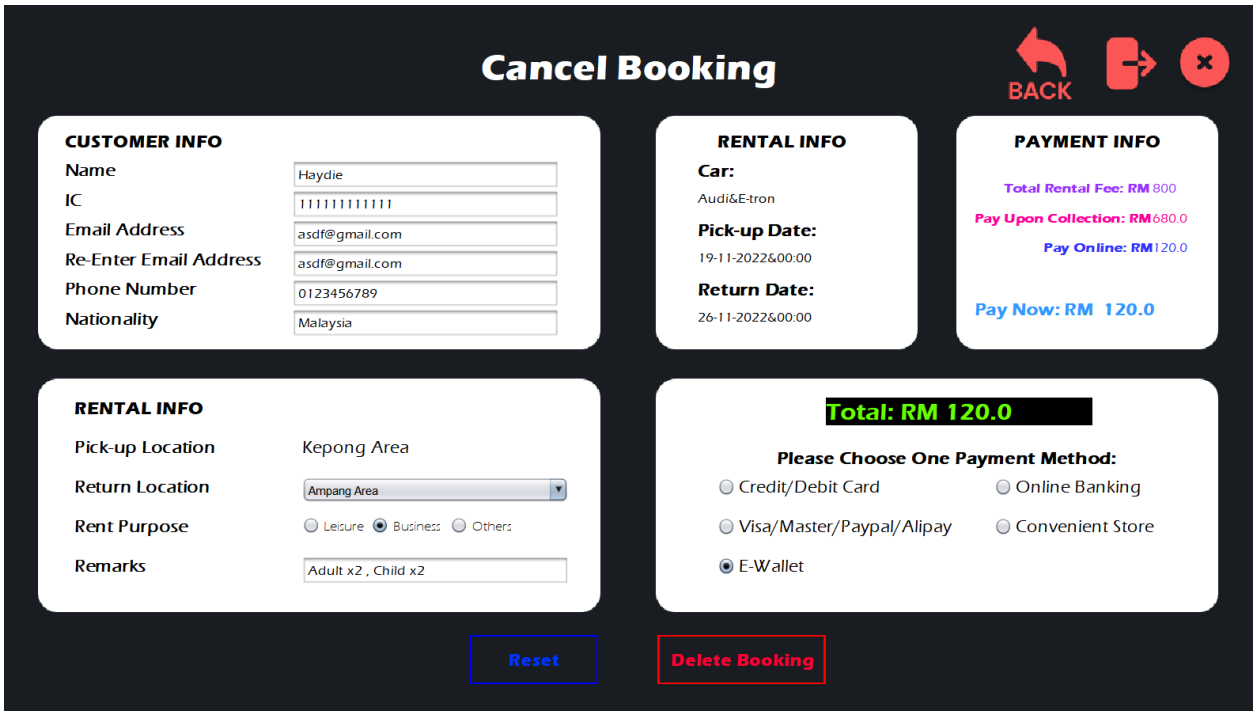


Figure 137 Manage Booking

This is “Cancel Booking” page which is for admin to cancel a booking.

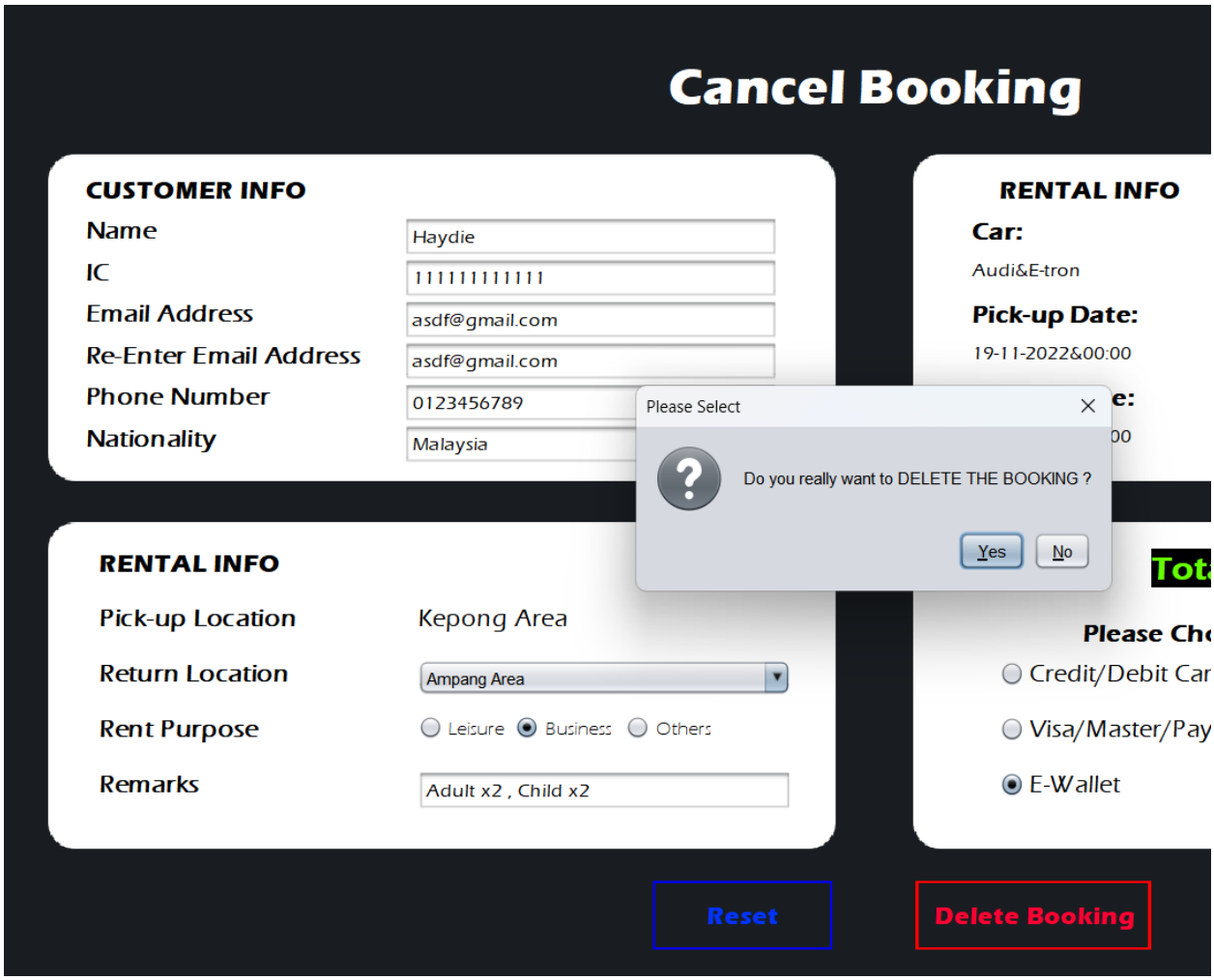


Figure 138 Manage Booking

This message will show for admin to confirm their decision if admin clicked on “Delete Booking” button.

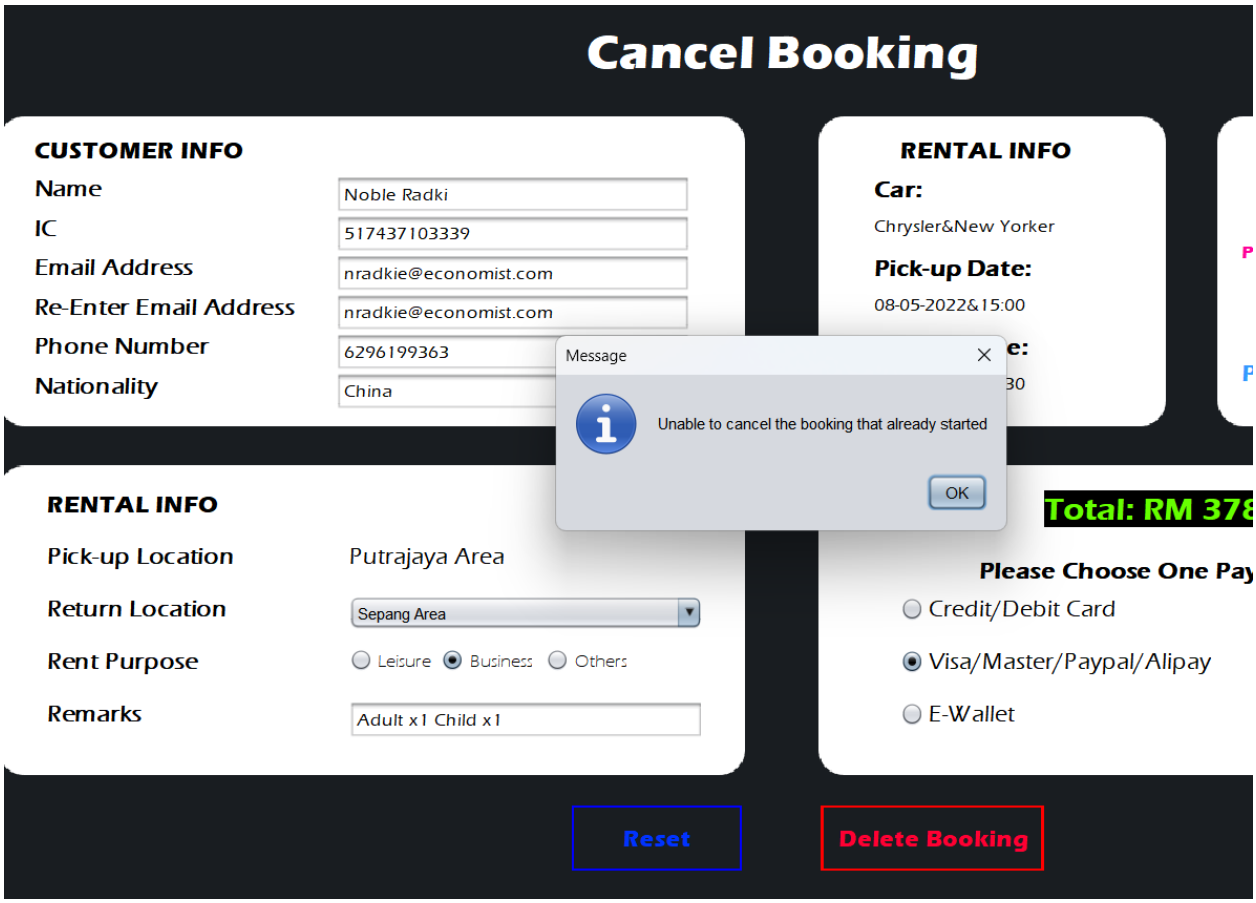


Figure 139 Manage Booking

This message will show if admin want to delete the booking, but the booking already started.

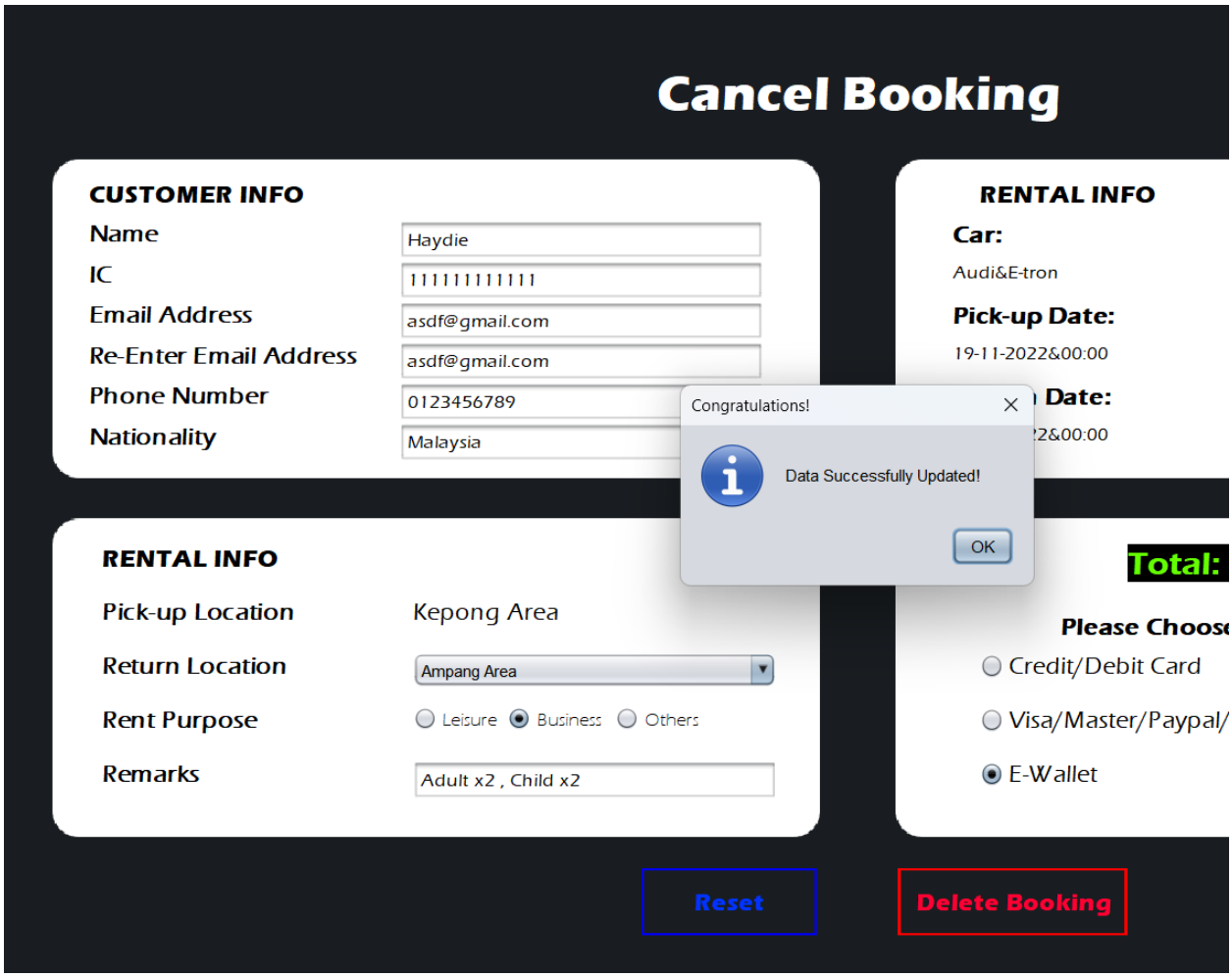


Figure 140 Manage Booking

This message will show if the booking deleted successfully.

Figure 141 Manage Booking

The admin may search for a car on this page by choosing the pickup location, pickup date, and pickup and return date.

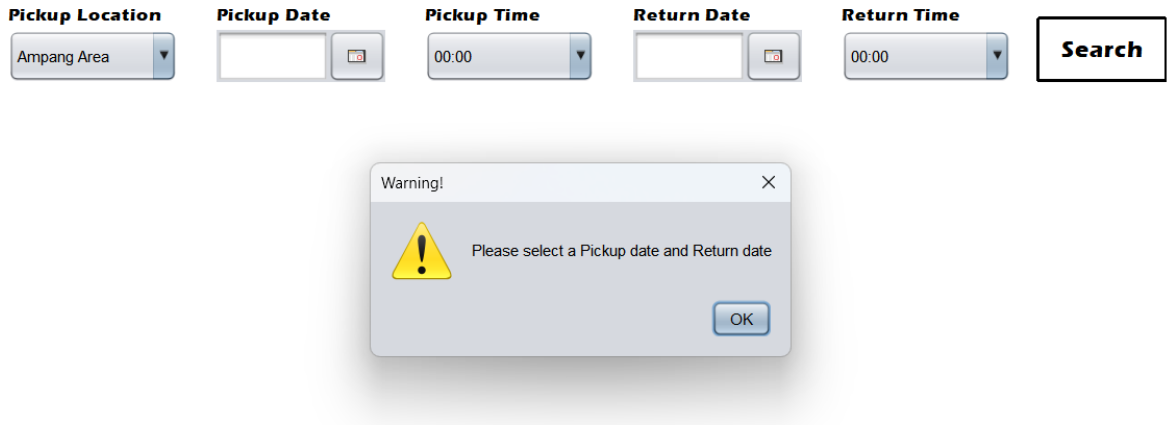


Figure 142 Manage Booking

This warning message would pop up, if customers forgot to select the Pickup Date or Return Date.

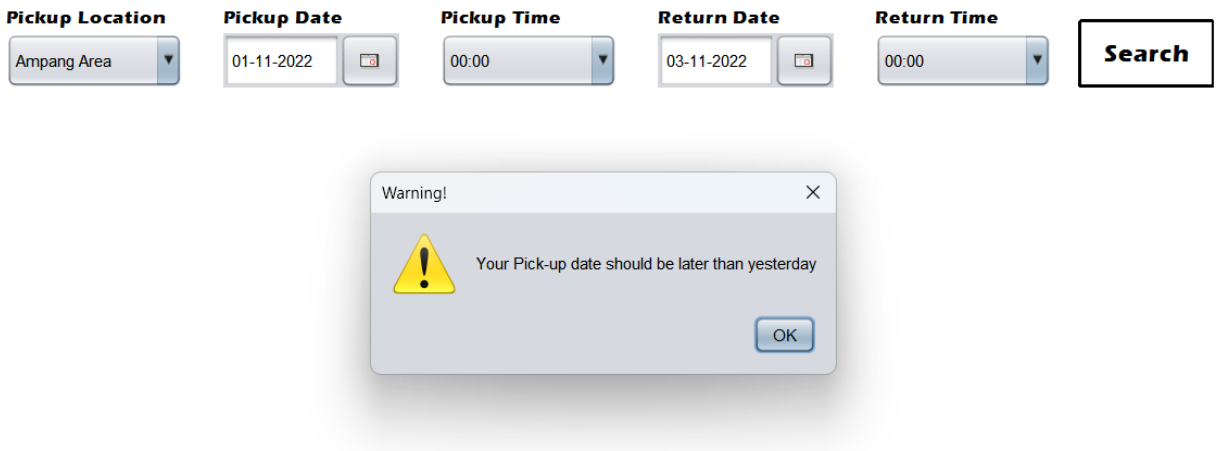


Figure 143 Manage Booking

This message will pop up if the Pickup Date is before today's Date.

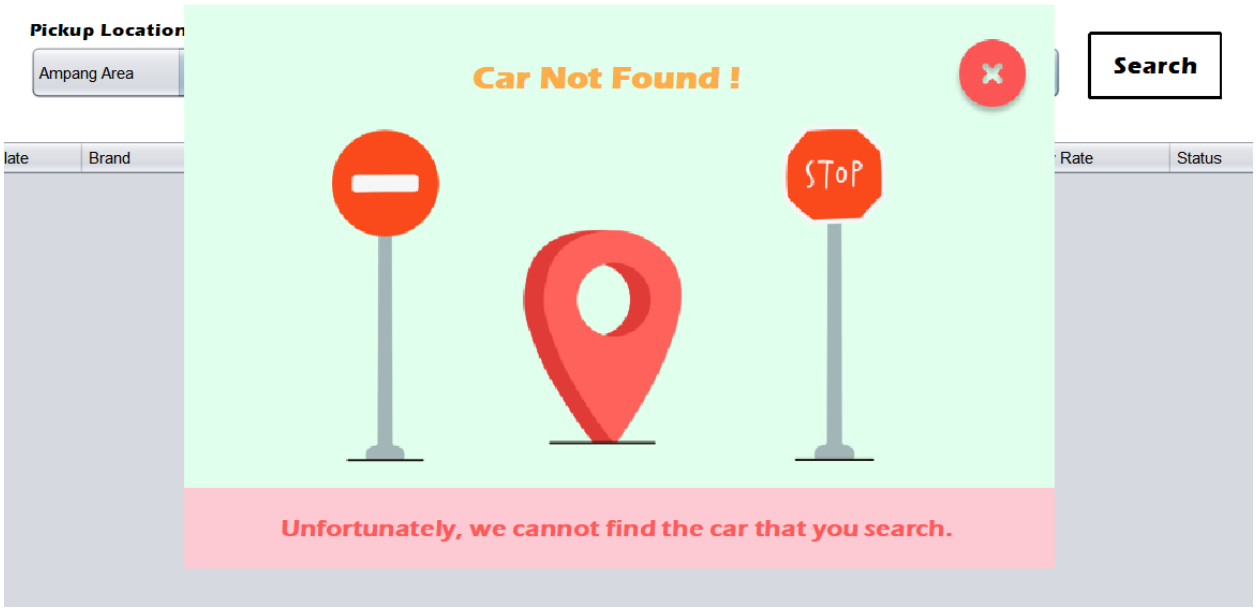


Figure 144 Manage Booking

This message will pop up if there are no available car between the date.

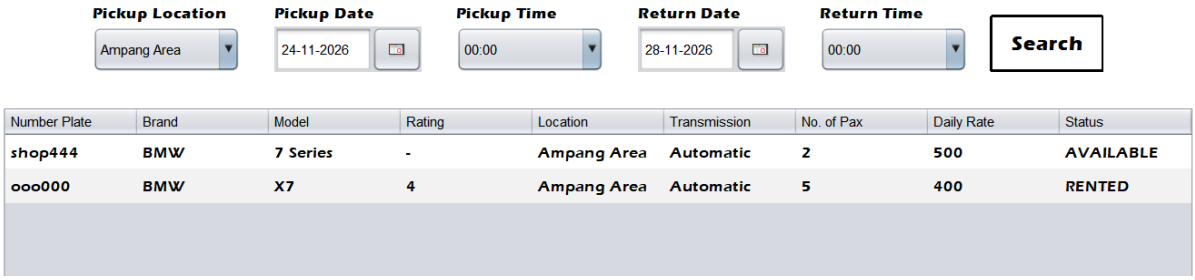


Figure 145 Manage Booking

After select all the option, it will show up a table and car details.

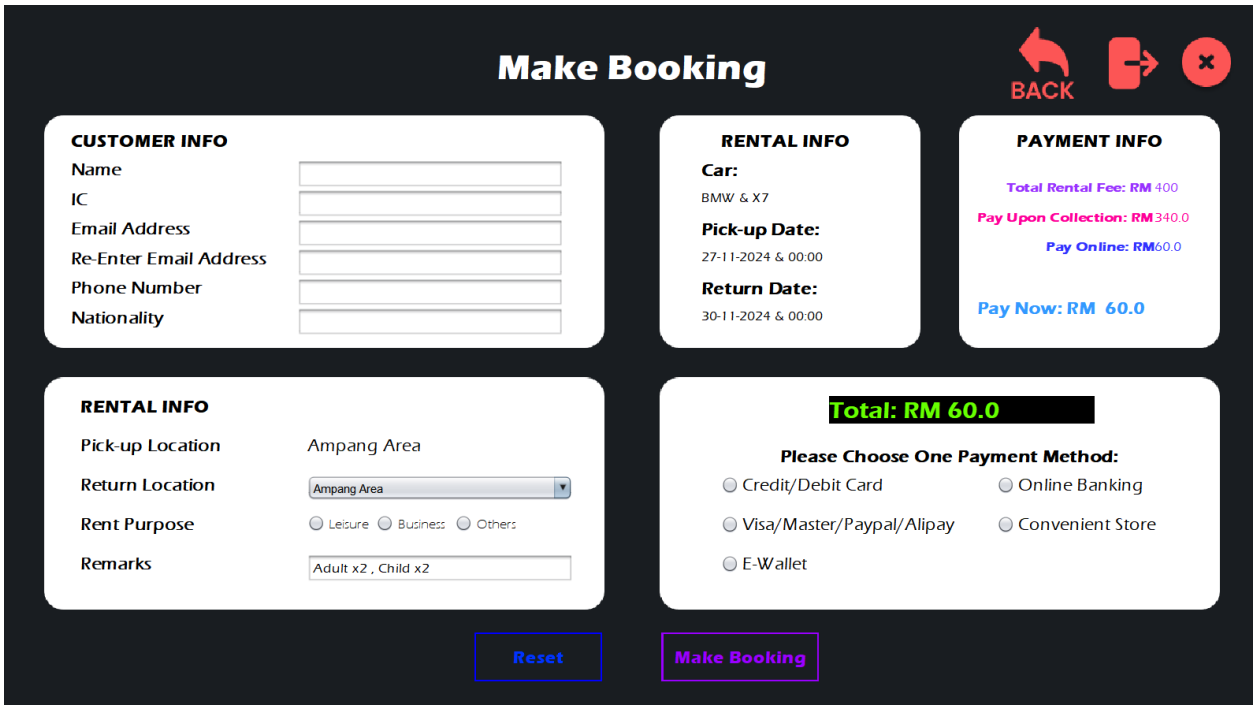


Figure 146 Manage Booking

This is the booking page where admin can enter customer information such as name, IC, email address, phone number, nationality. Besides, admin can select Return Location, Rent Purpose, Remarks and Payment Method for customers. Before press the “Make Booking” button, must check the Rental Info and Payment Info. After checking, click the “Make booking” to book the car.

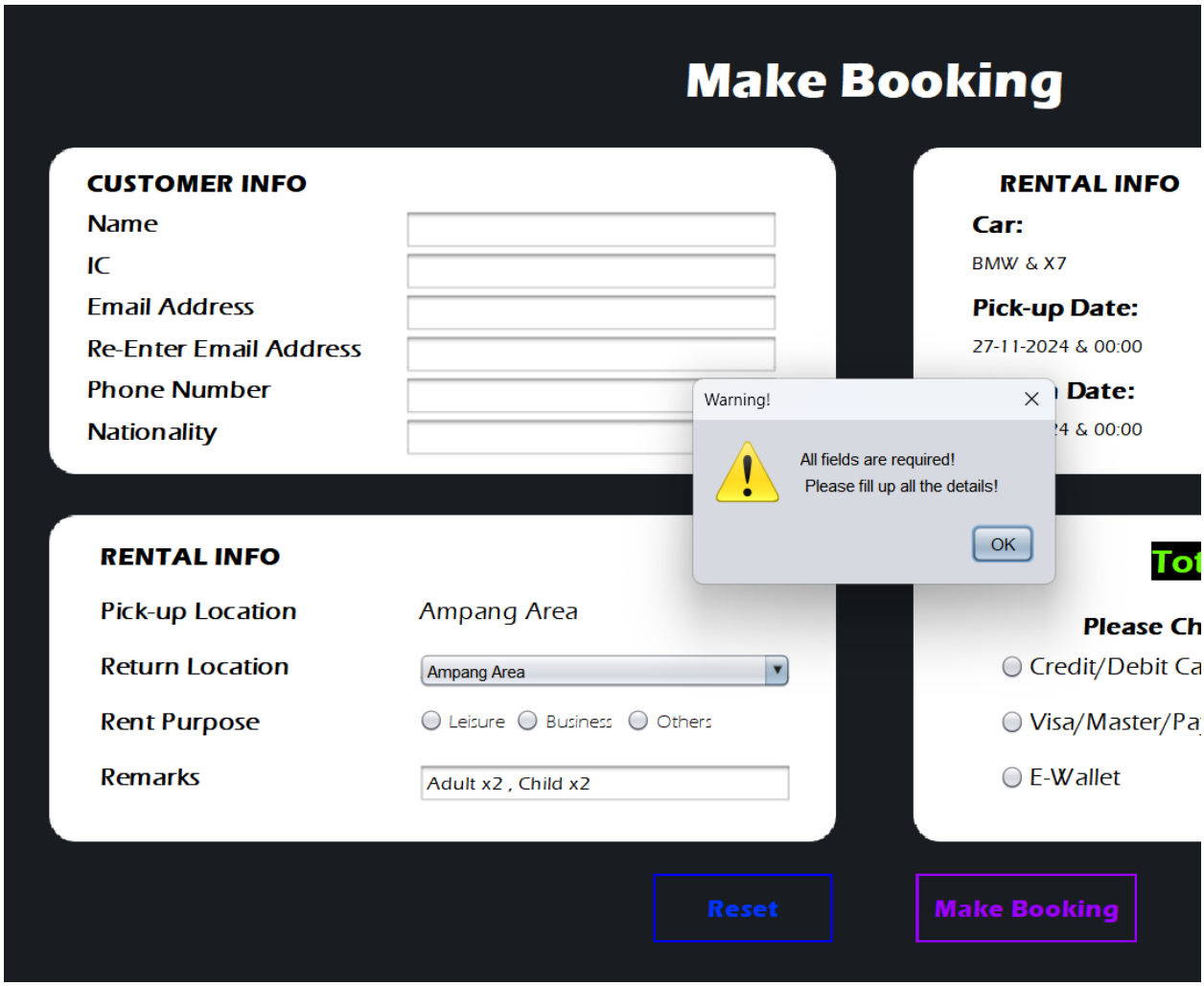


Figure 147 Manage Booking

This warning message will show up if admin didn't key in the information.

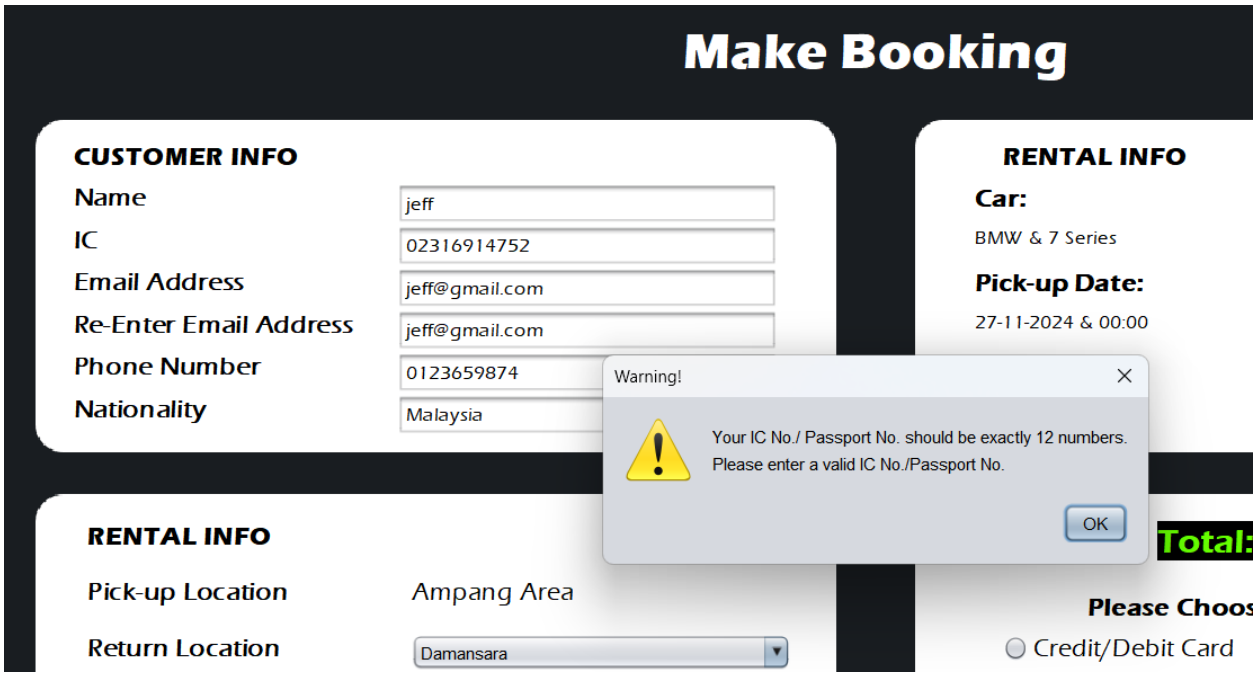


Figure 148 Manage Booking

If the IC number or passport number entered on the "Make Booking" page does not exactly contain 12 digits, a warning message will appear. This notice admin to request customer to provide a valid IC number or passport number into the system.

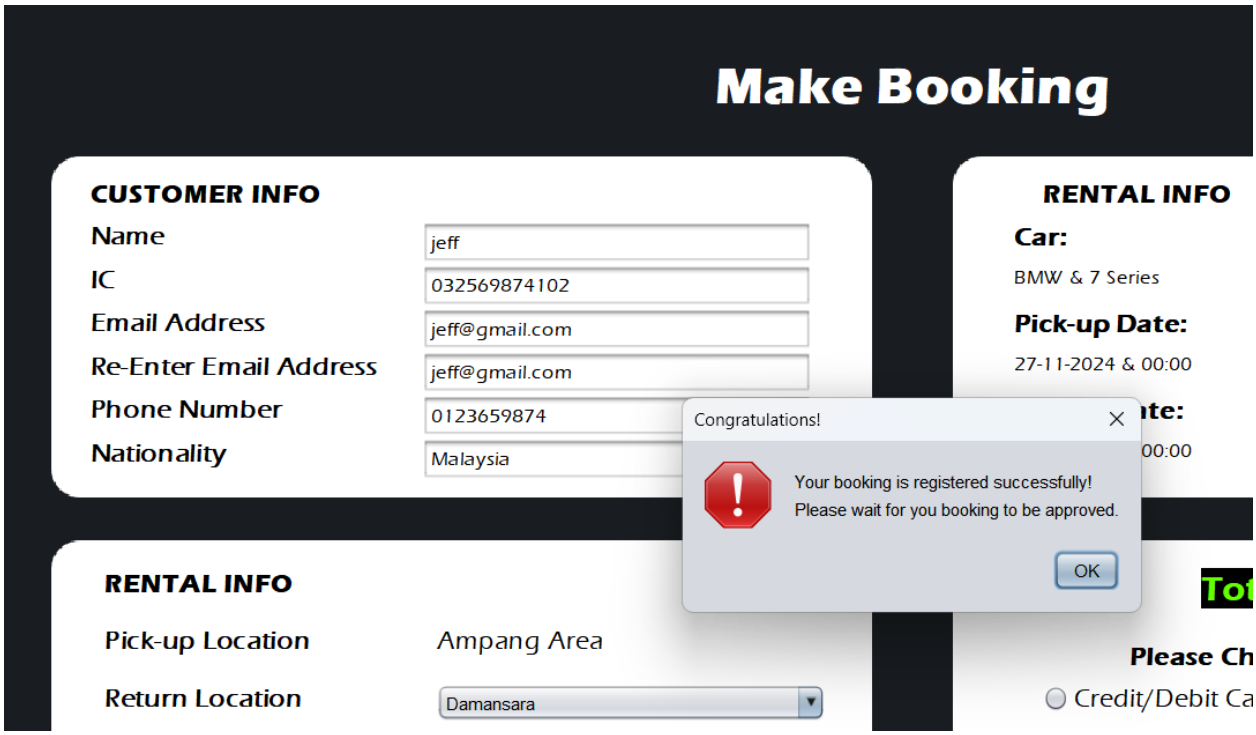


Figure 149 Manage Booking

After customers press the “Make booking” button. The system will show “Your booking is registered successfully” and customers need to wait for admin to approve the booking.

5.4.6 Booking Confirmation

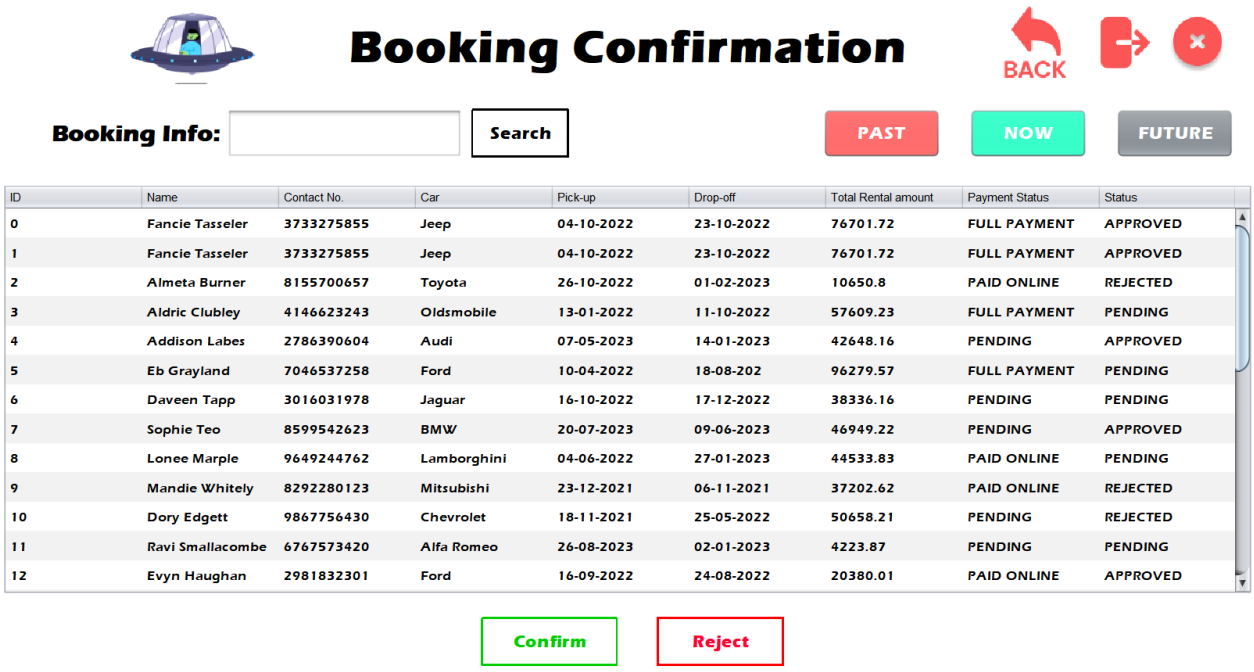


Figure 150 Booking Confirmation

This is “Booking Confirmation” page where will show all the customer booking inside the table.

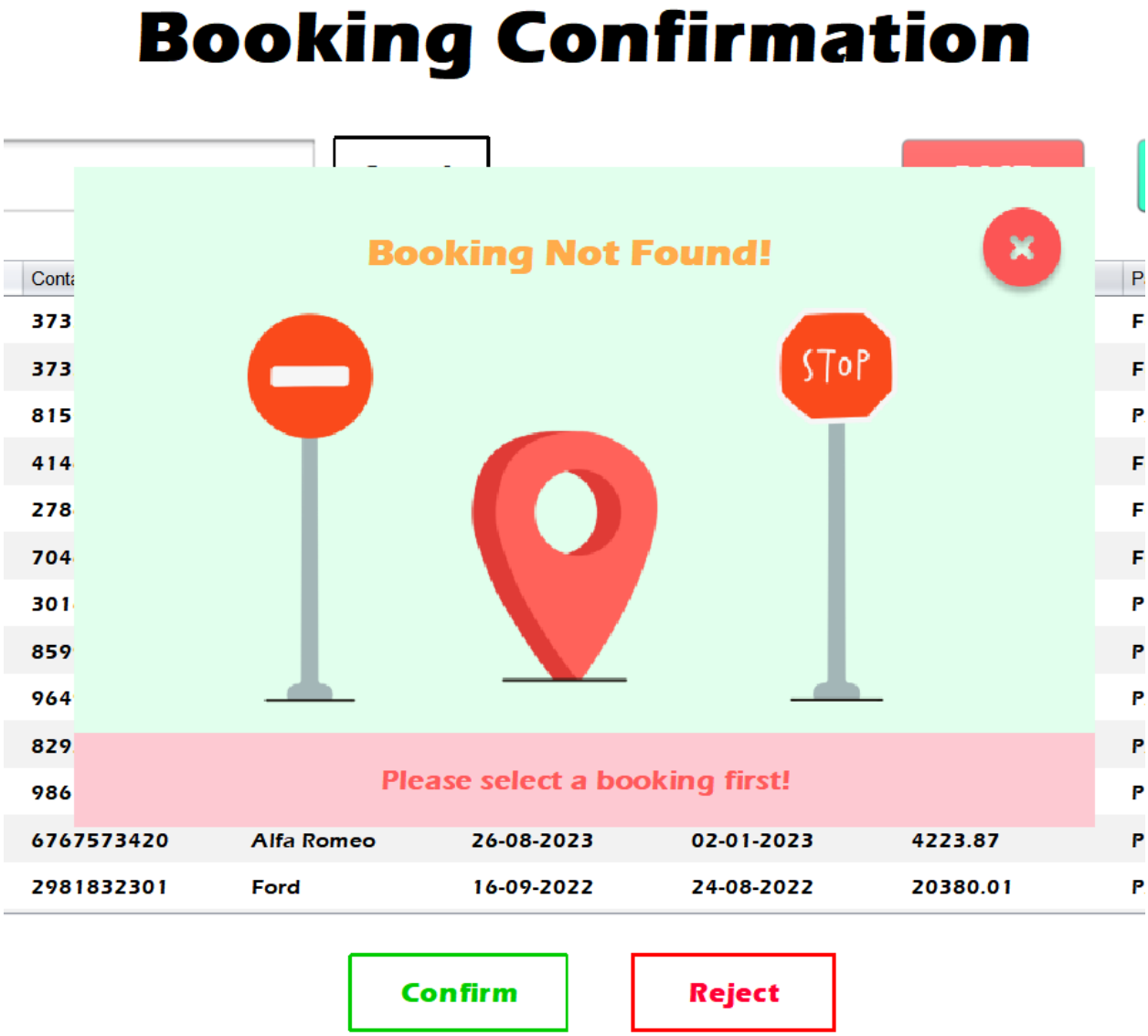


Figure 151 Booking Confirmation

This message will pop up if admin click on “Confirm” or “Reject” button without select any booking.

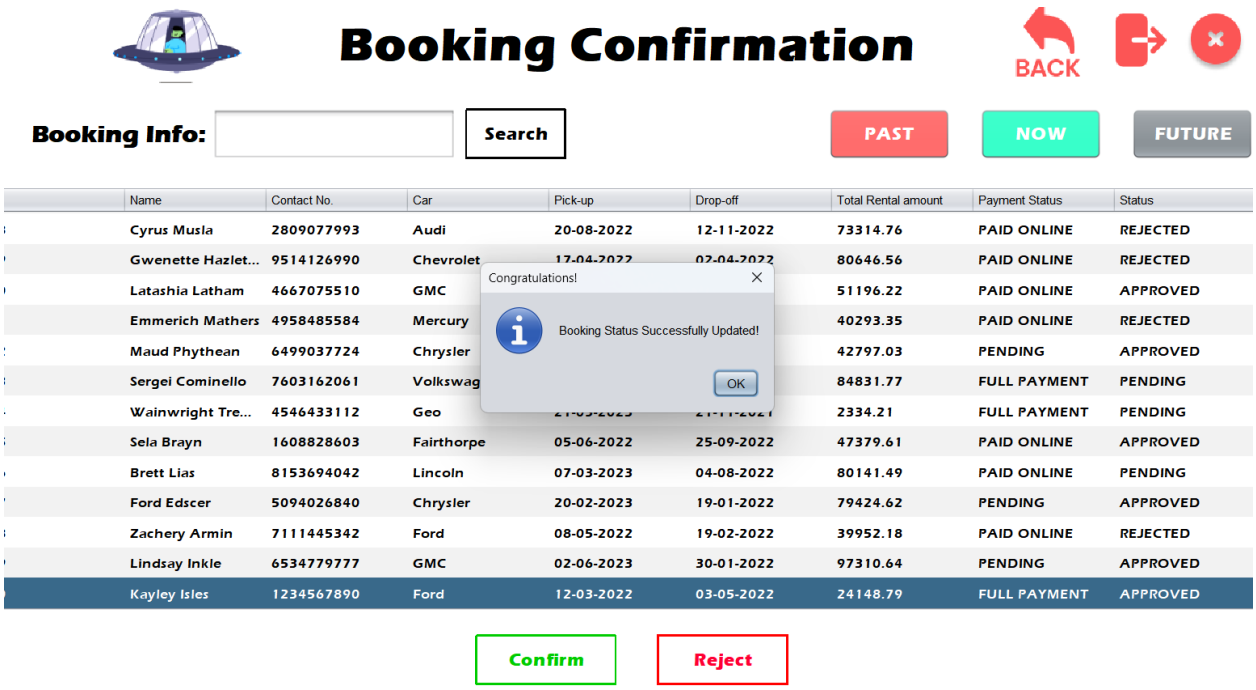


Figure 152 Booking Confirmation

This message will show after admin select a booking and click on “Confirm” button or “Reject” button.

5.4.7 Customer Booking History

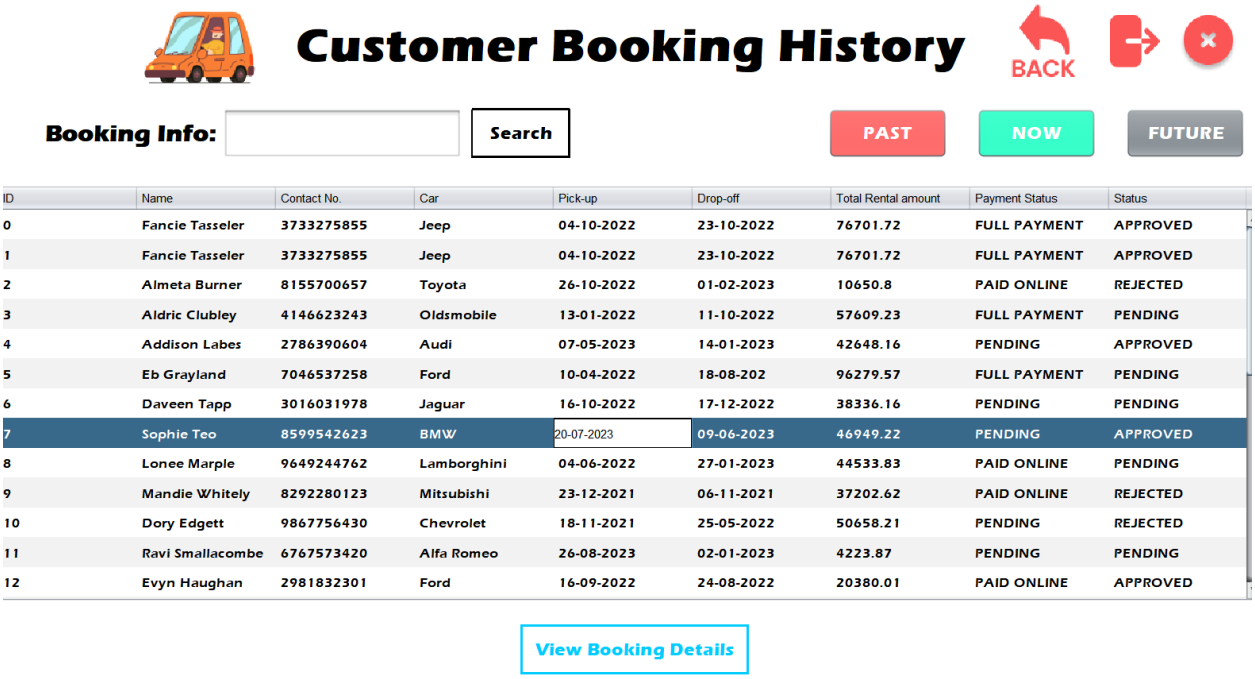


Figure 153 Customer Booking History

This is “Customer Booking History” page where admin able to see all customer’s bookings history.

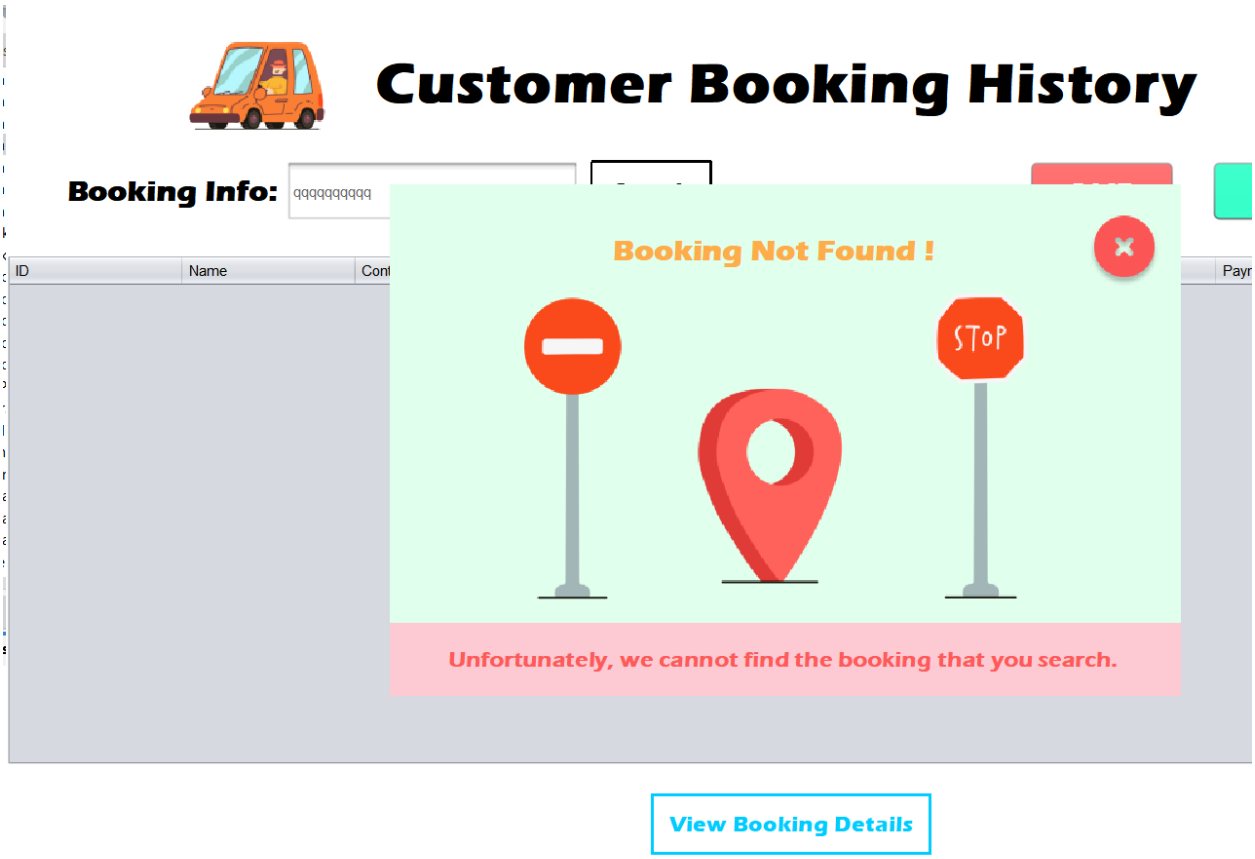


Figure 154 Customer Booking History

This message will show if there are no booking found according to the search.

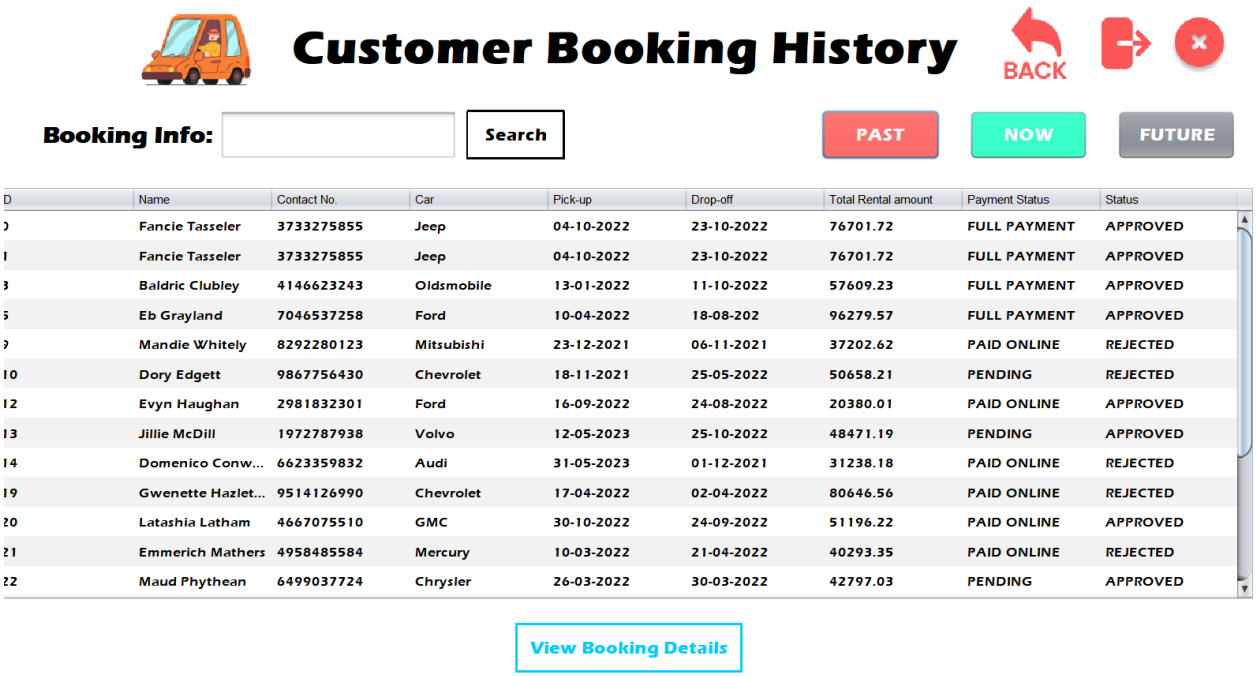


Figure 155 Customer Booking History

Admin can click on the “PAST” button to sort the date and system will show the bookings which bookings are past.

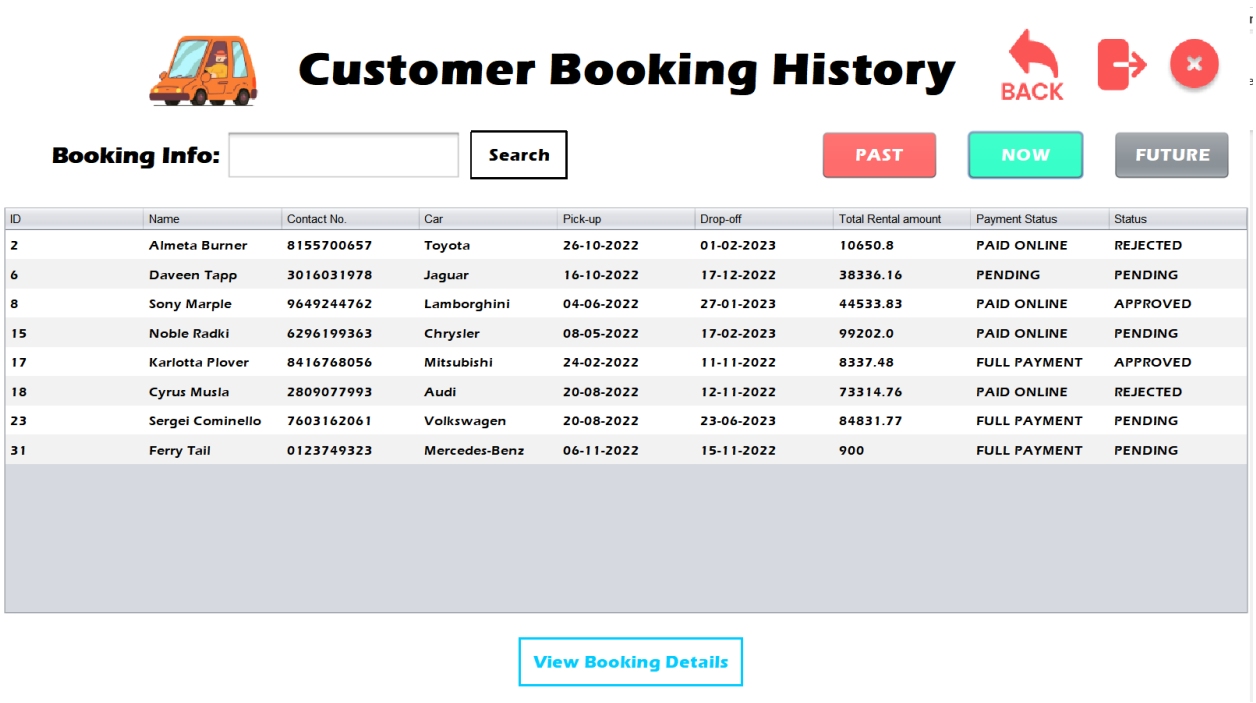


Figure 156 Customer Booking History

Admin can click on the “NOW” button to sort the date and system will show bookings for today.

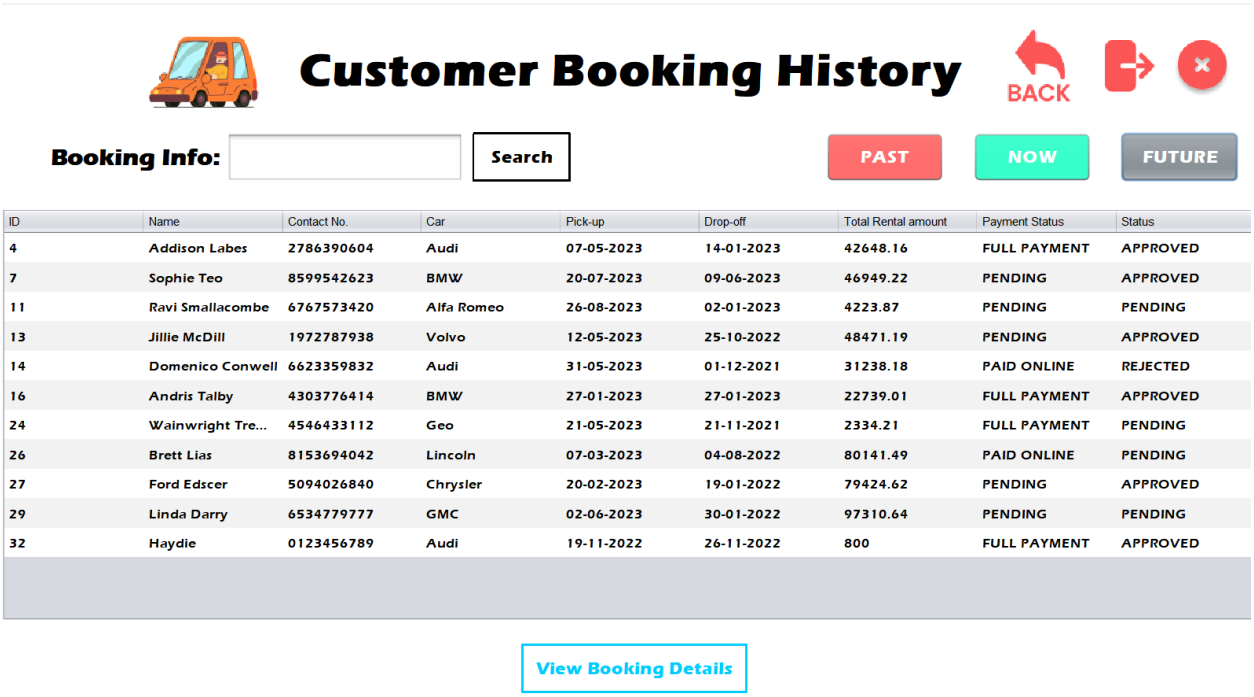


Figure 157 Customer Booking History

Admin can click on the “FUTURE” button to sort the date and system will show the bookings which bookings are coming soon.

Figure 158 Customer Booking History

Admin can click on each of their booking from “Customer Booking History” page to check the booking details. Moreover, there have two button such as “Feedback” and “Receipt” button for admin to check the receipt and help customer to submit their feedback.

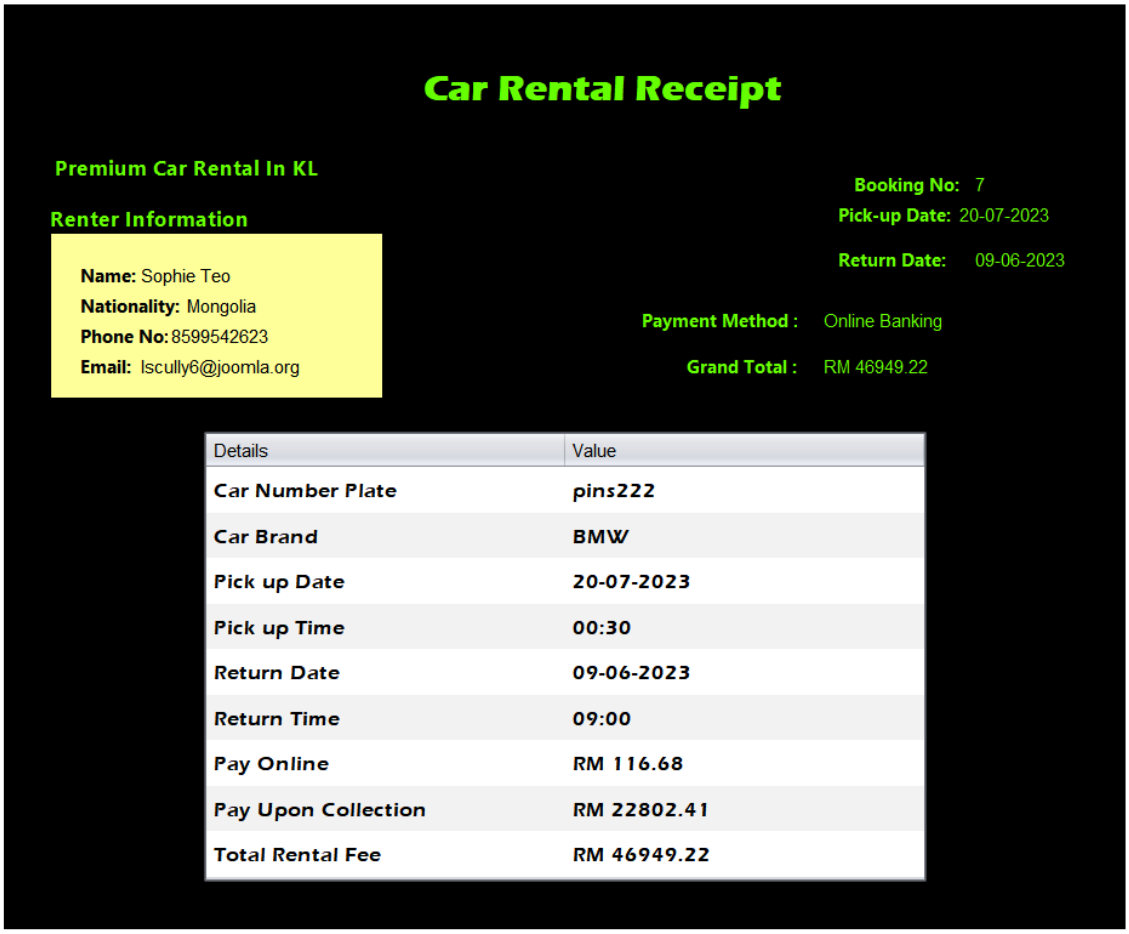


Figure 159 Customer Booking History

This is the “Car Rental Receipt” page where admin can view the details of the receipt such as date, payment total, car and so on.

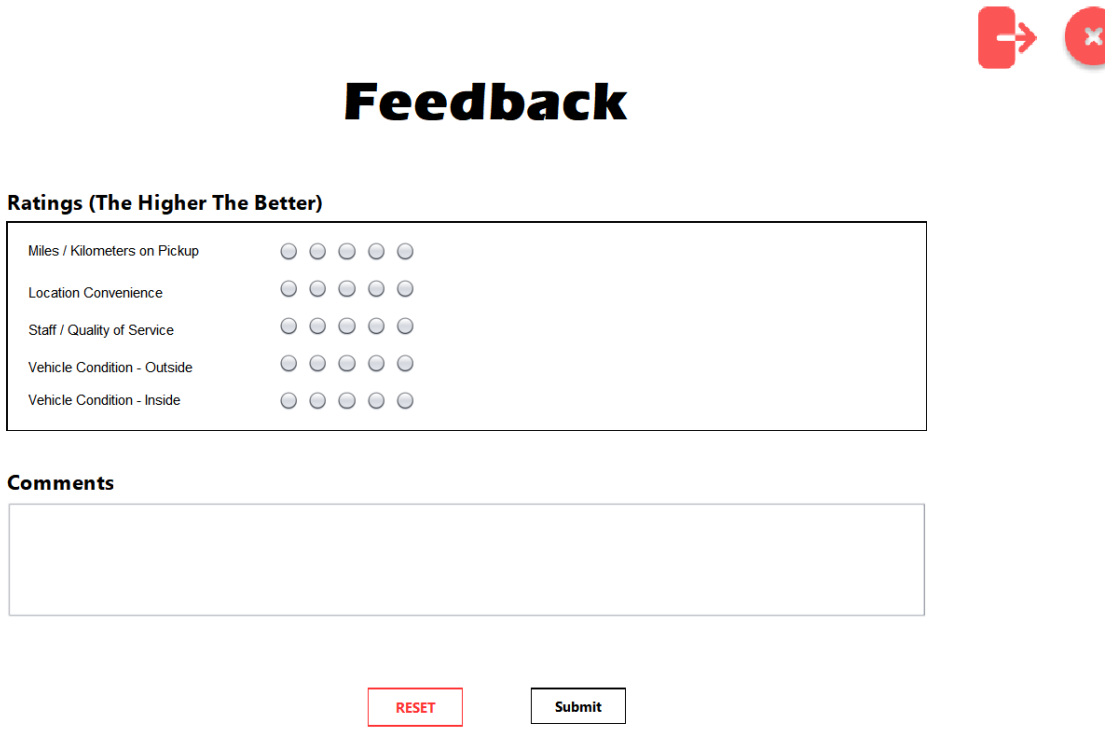


Figure 160 Customer Booking History

This is “Feedback” page for admin to help customers to give rating and comments for their booking. In the Rating column, there has “Miles or kilometre on Pickup”, “Location Convenience”, “Staff or Quality of Service”, “Vehicle Conditions on Outside” and “Vehicle Conditions on Inside” and there has one text field for customer to enter their comments.

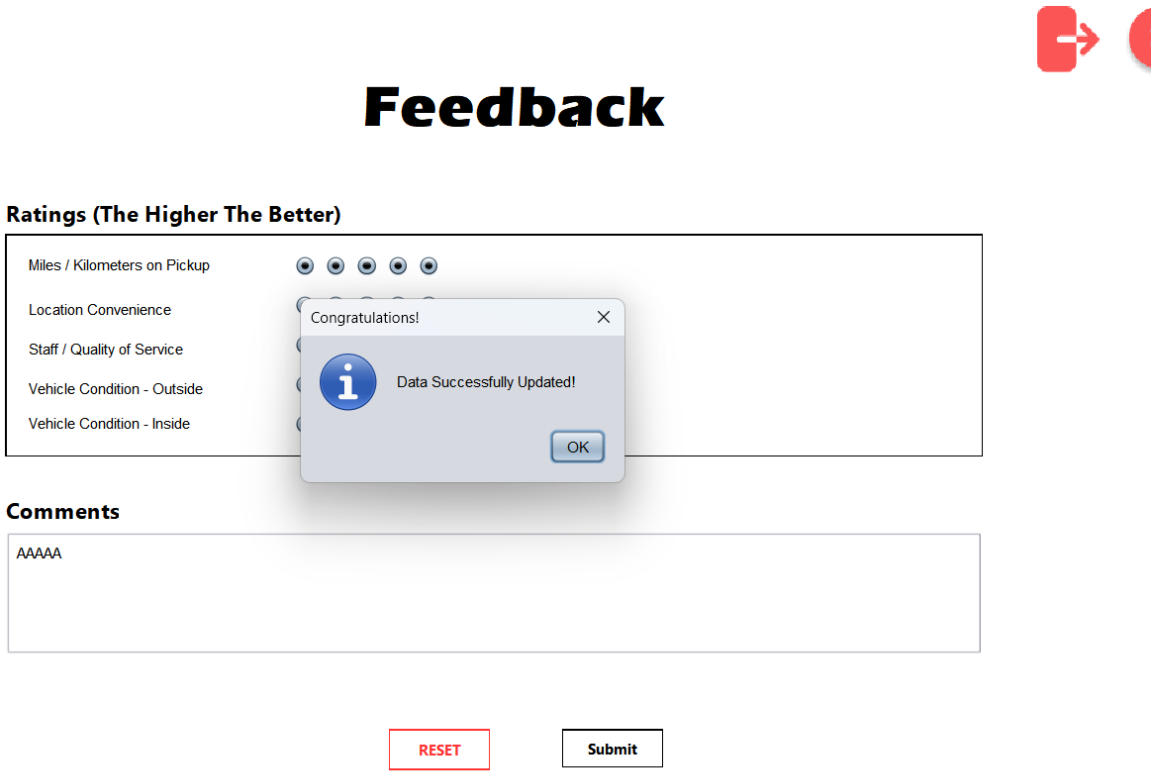


Figure 161 Customer Booking History

After entering all the information, click the “Submit” button to submit their feedback or click the “RESET” button to clear all the feedback.

5.4.8 Payment Collection

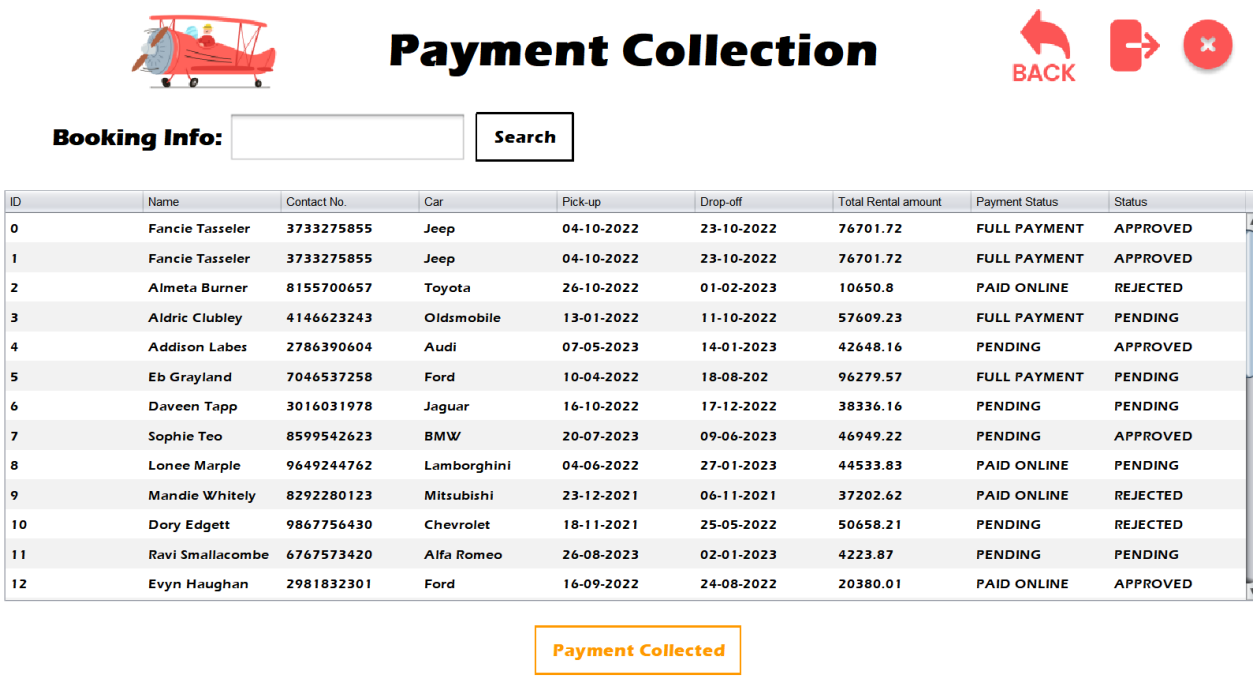


Figure 162 Payment Collection

This page is “Payment Collection” page where admin can change customer’ payment status after they pay the total amount.

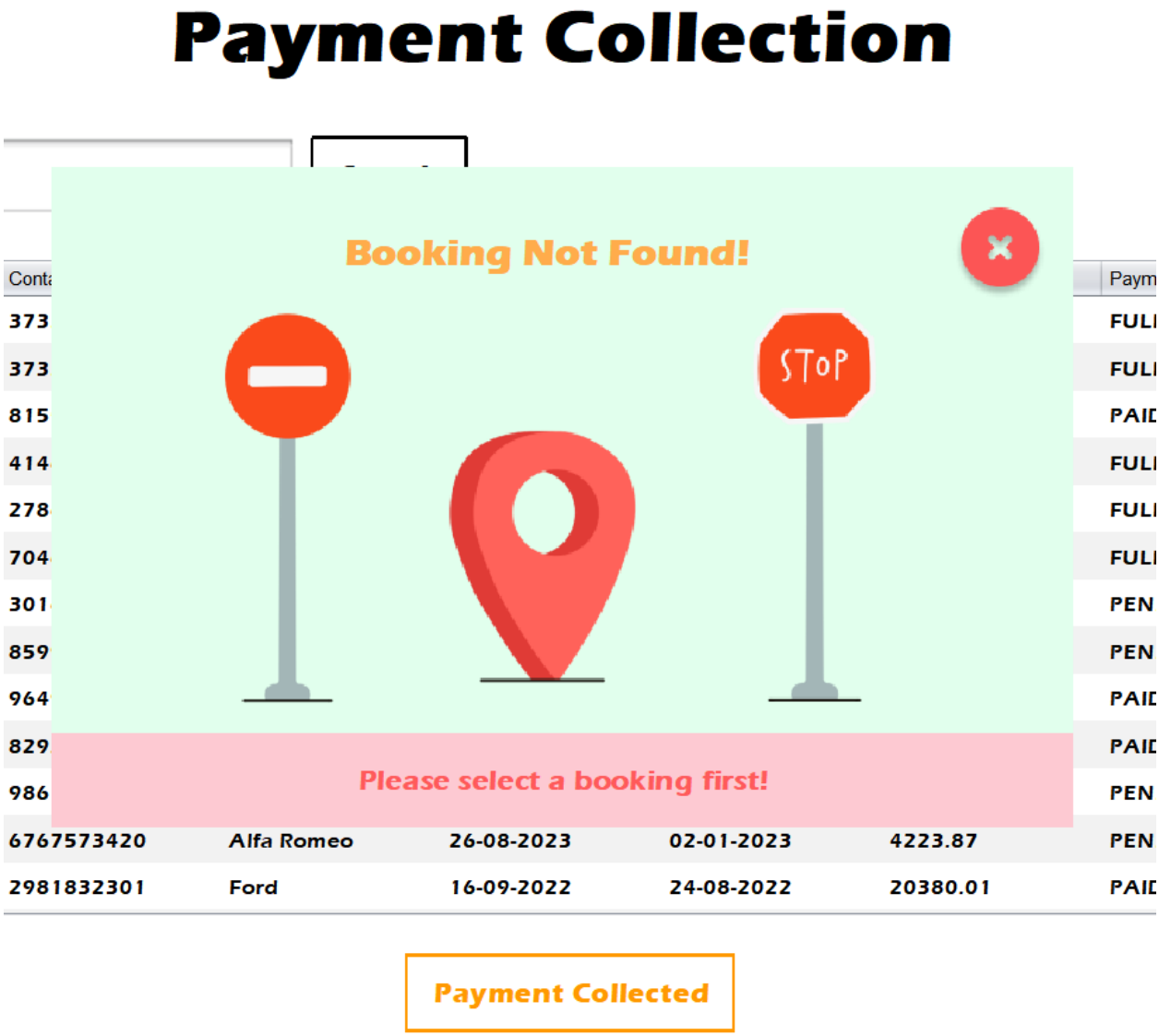


Figure 163 Payment Collection

In the “Payment Collection” page, this error message will show if admin click the “Payment Collected” button without select any booking .

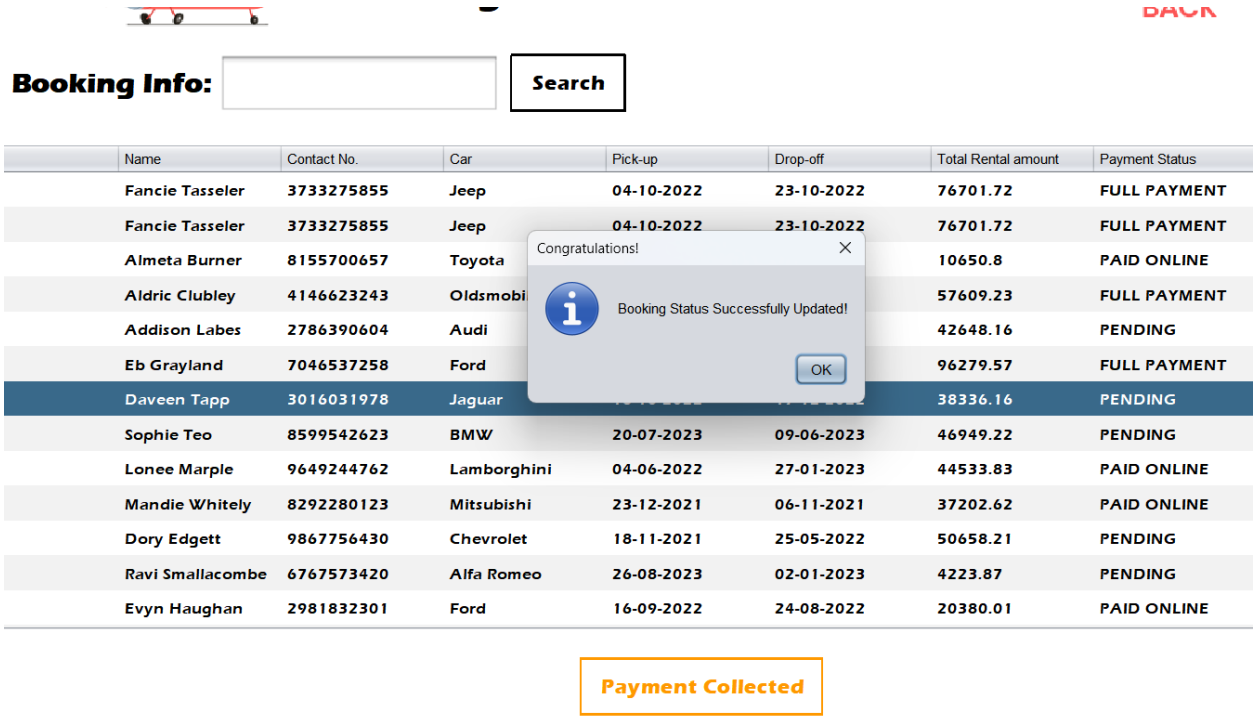


Figure 164 Payment Collection

This message will show after admin select a booking and click on the “Payment Collected” button.

5.4.9 Block A Car

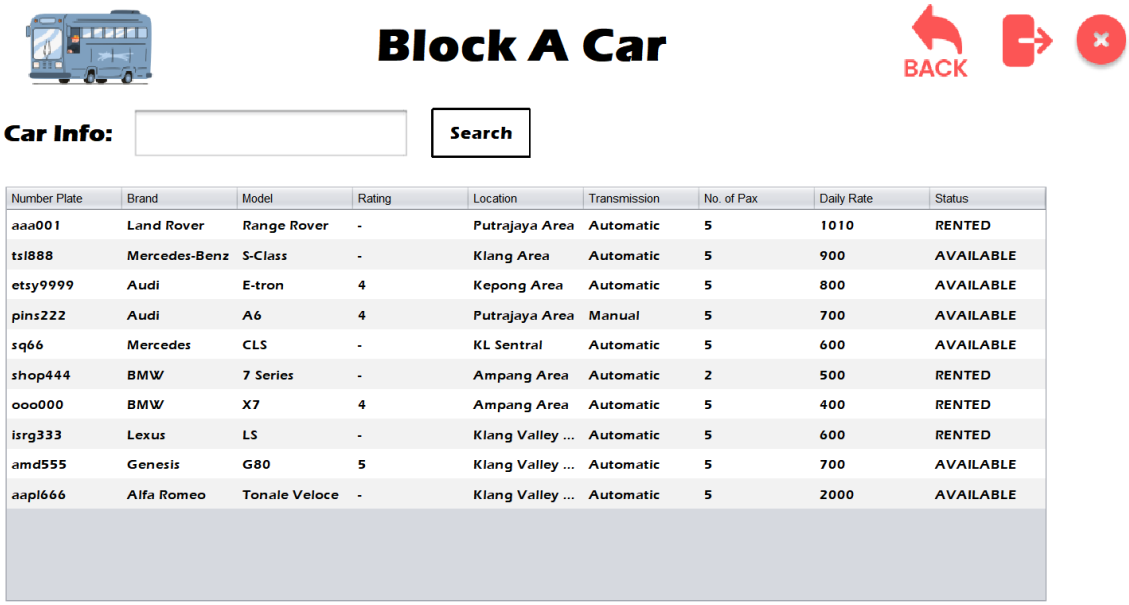


Figure 165 Block A Car

This is “Block A Car” page where admin able to block the car therefore customer will not be able to book the car. This is because the car may under maintenance thence it is unable to service.

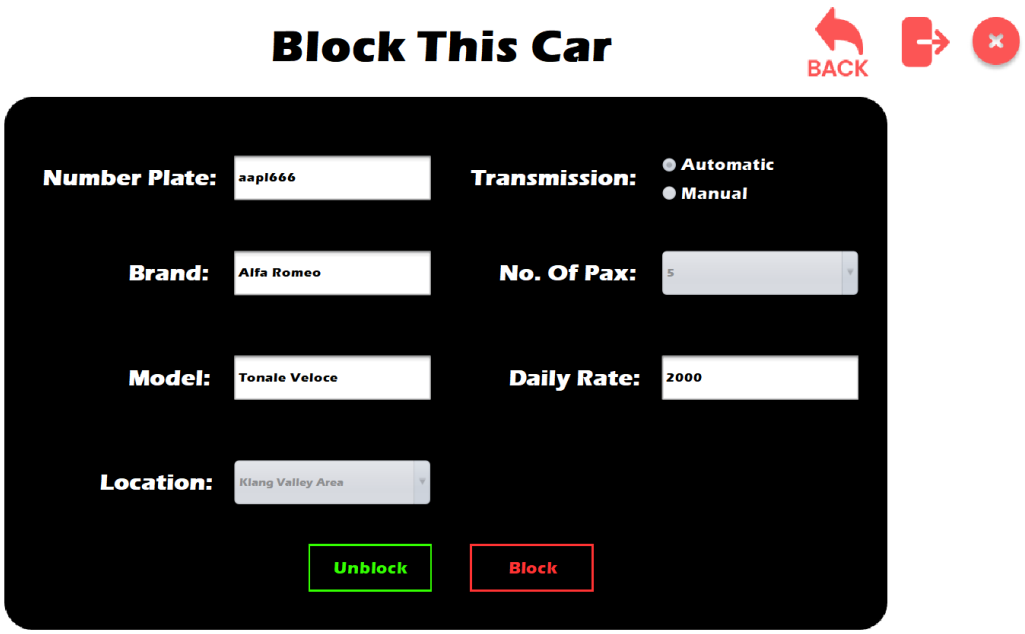


Figure 166 Block A Car

This page is to block or unblock the car.

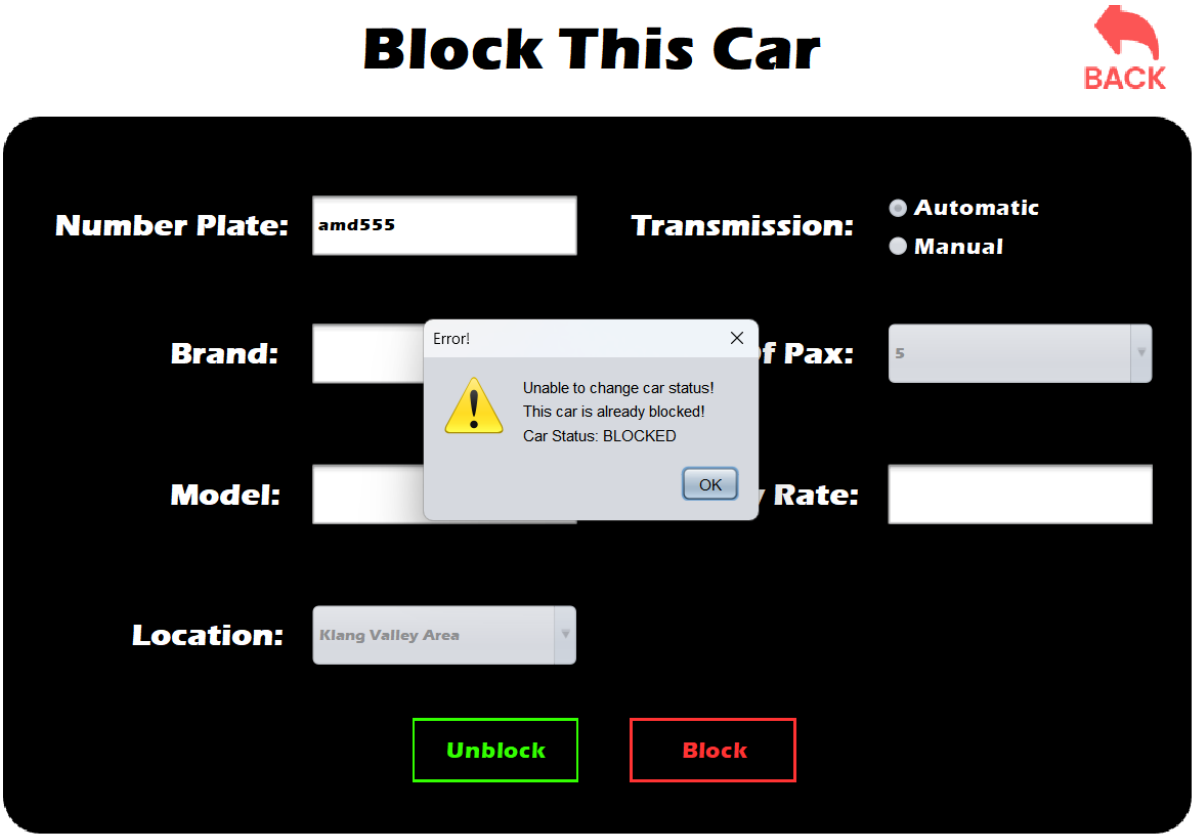


Figure 167 Block A Car

This warning message will show if admin try to block a car but the car are already blocked.

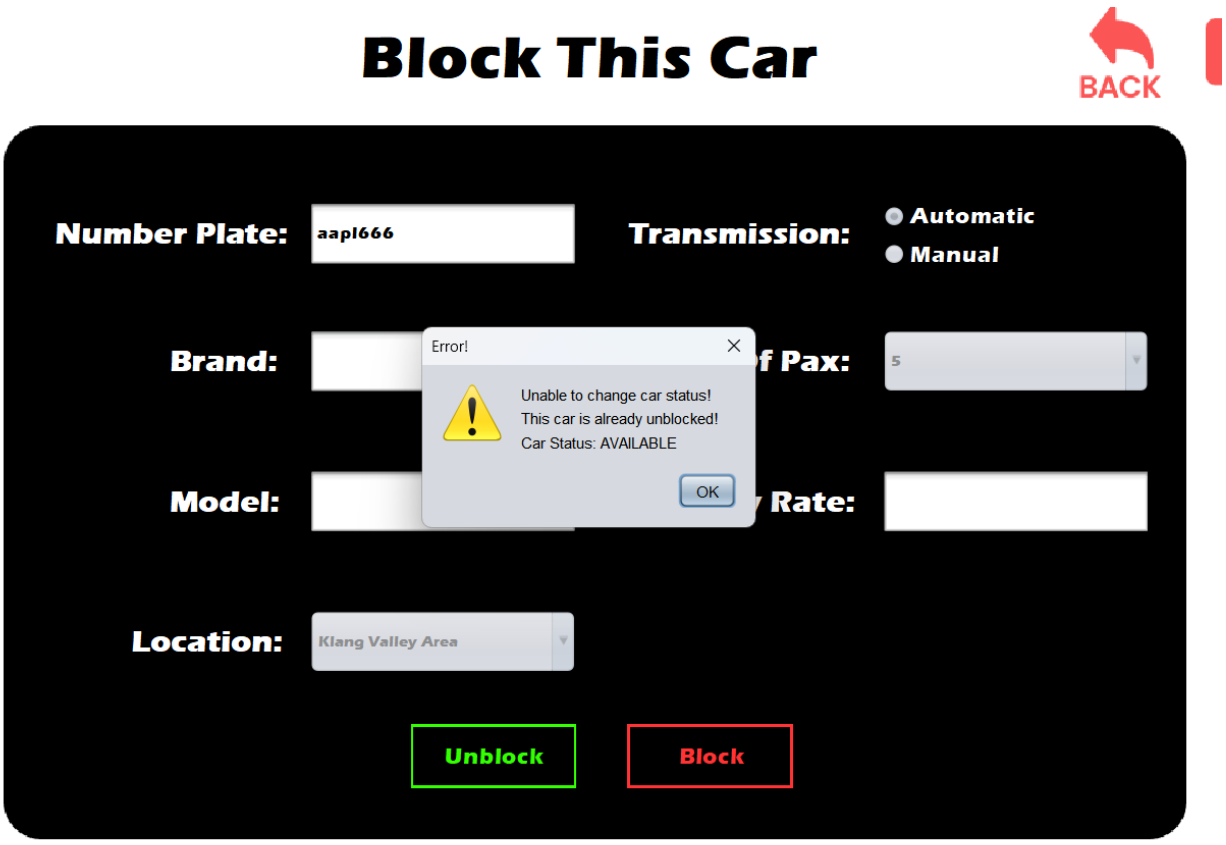


Figure 168 Block A Car

This warning message will show if admin try to unblock a car but the car are already unblocked.

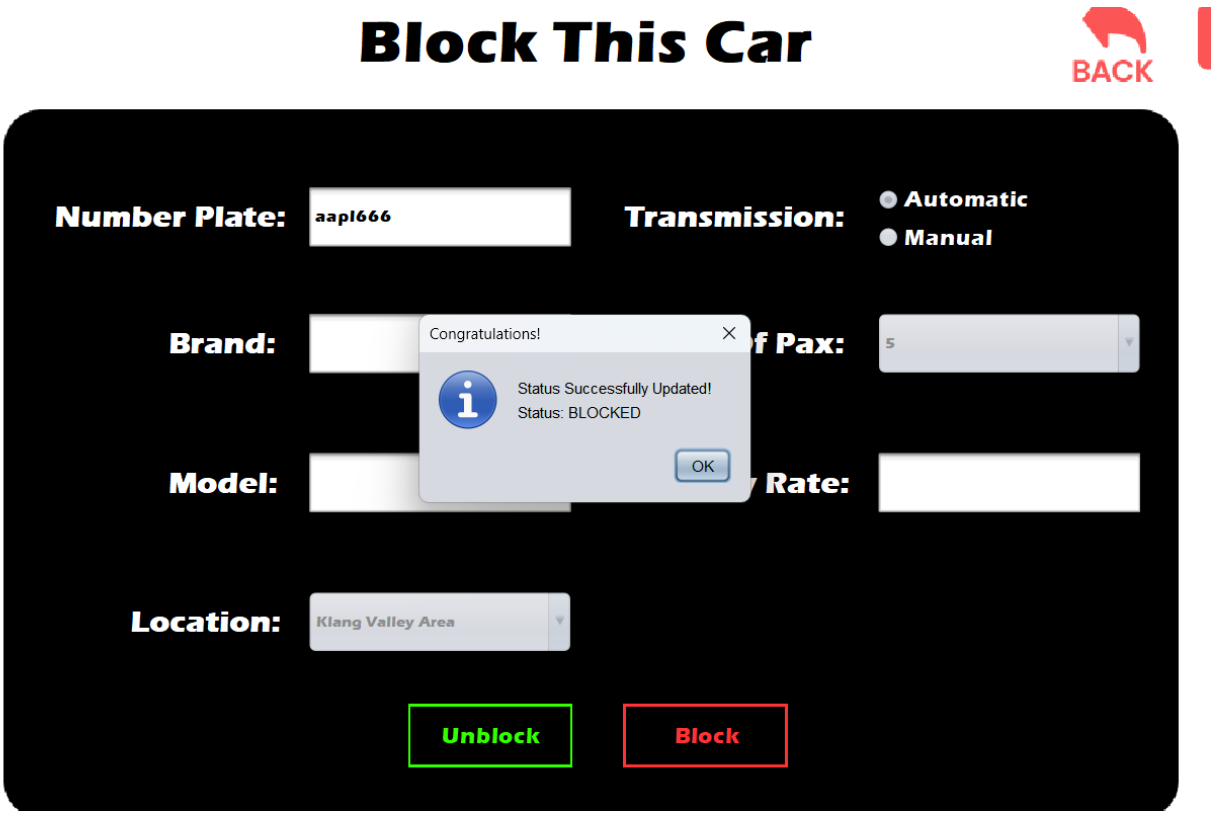


Figure 169 Block A Car

This message will show if the car is blocked successfully.

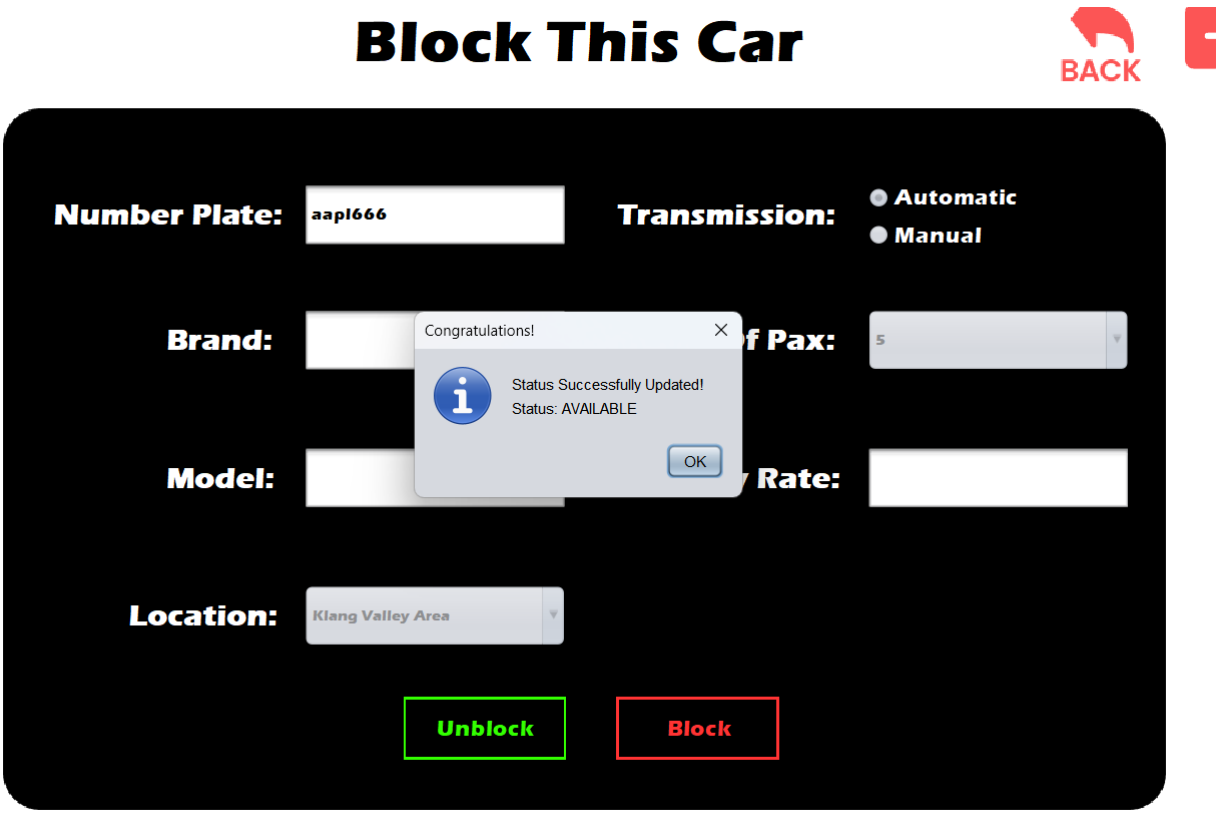


Figure 170 Block A Car

This message will show if the car is unblocked successfully.

5.4.10 Manage Customer Registration

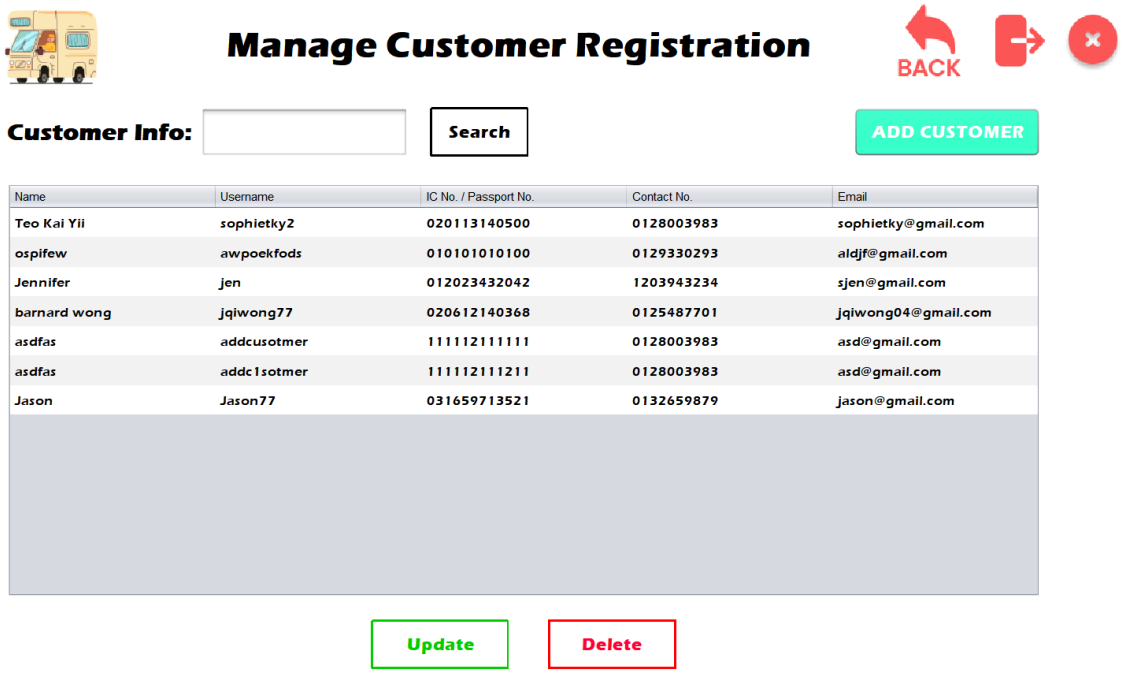


Figure 171 Manage Customer Registration

This is “Manage Customer Registration” page where admin can update or delete the customer information.

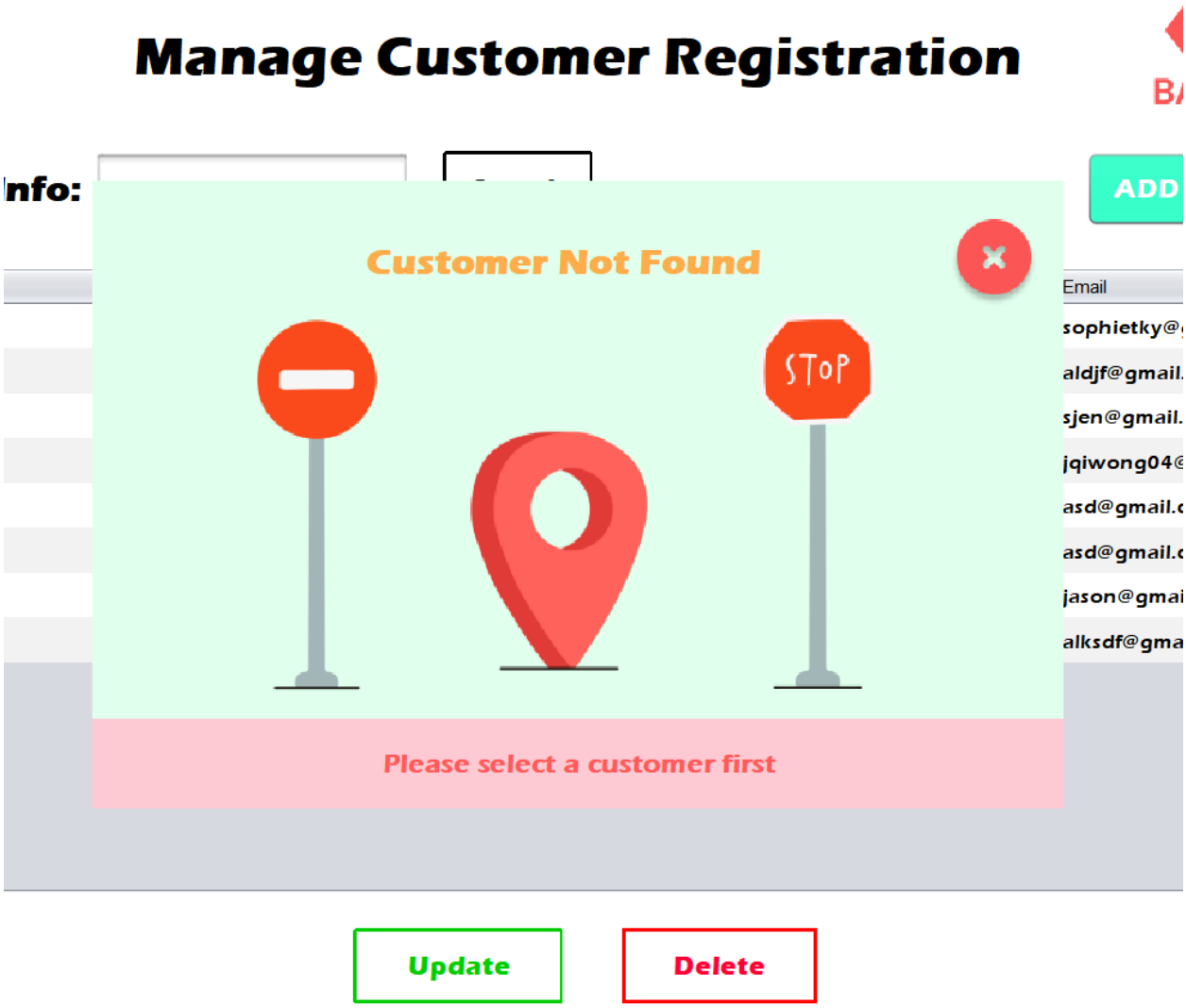


Figure 172 Manage Customer Registration

This error message will show if admin click on “Update” button or “Delete” button without select any customer.

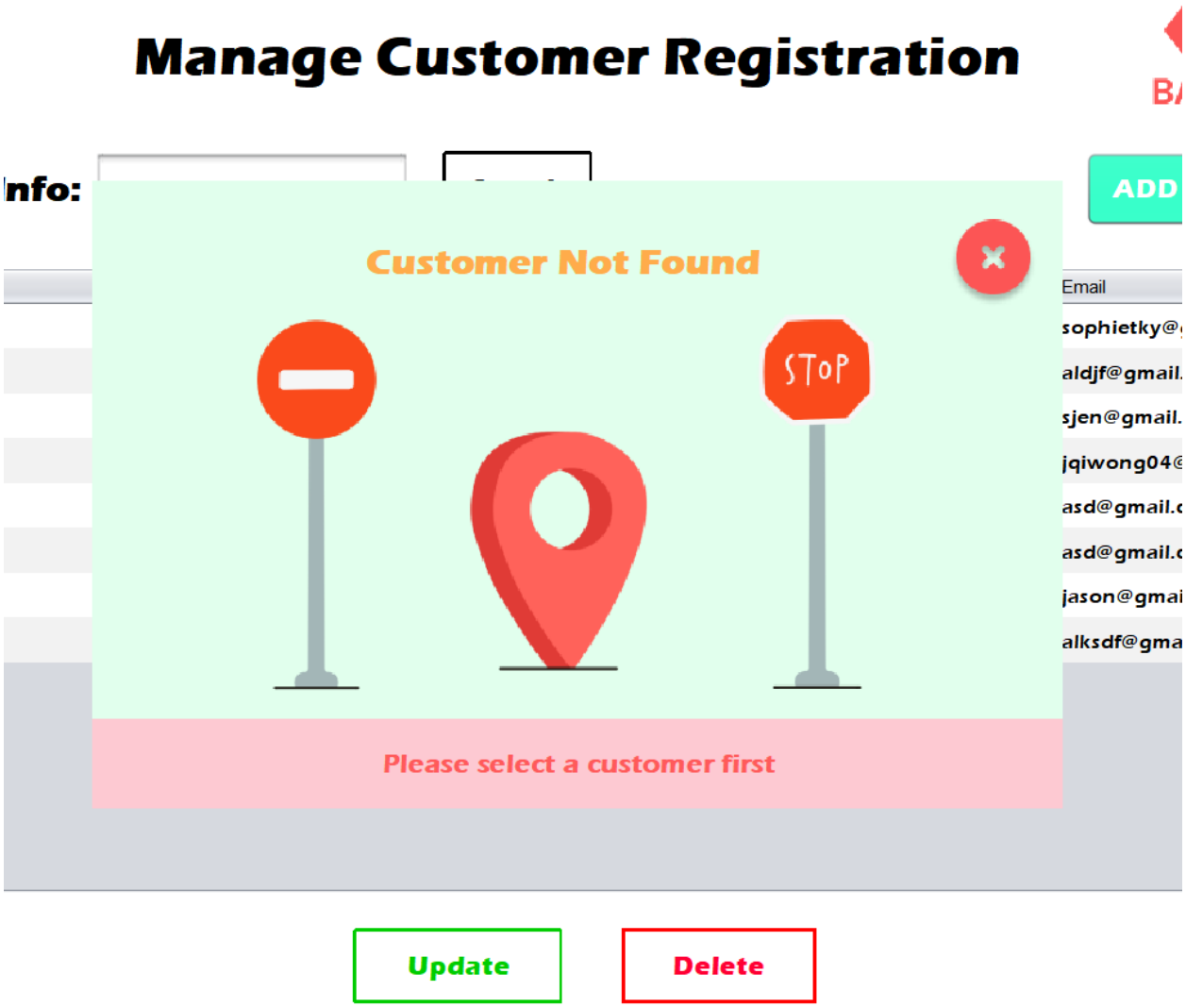


Figure 173 Manage Customer Registration

This message will show if there are no customer found according to the search.

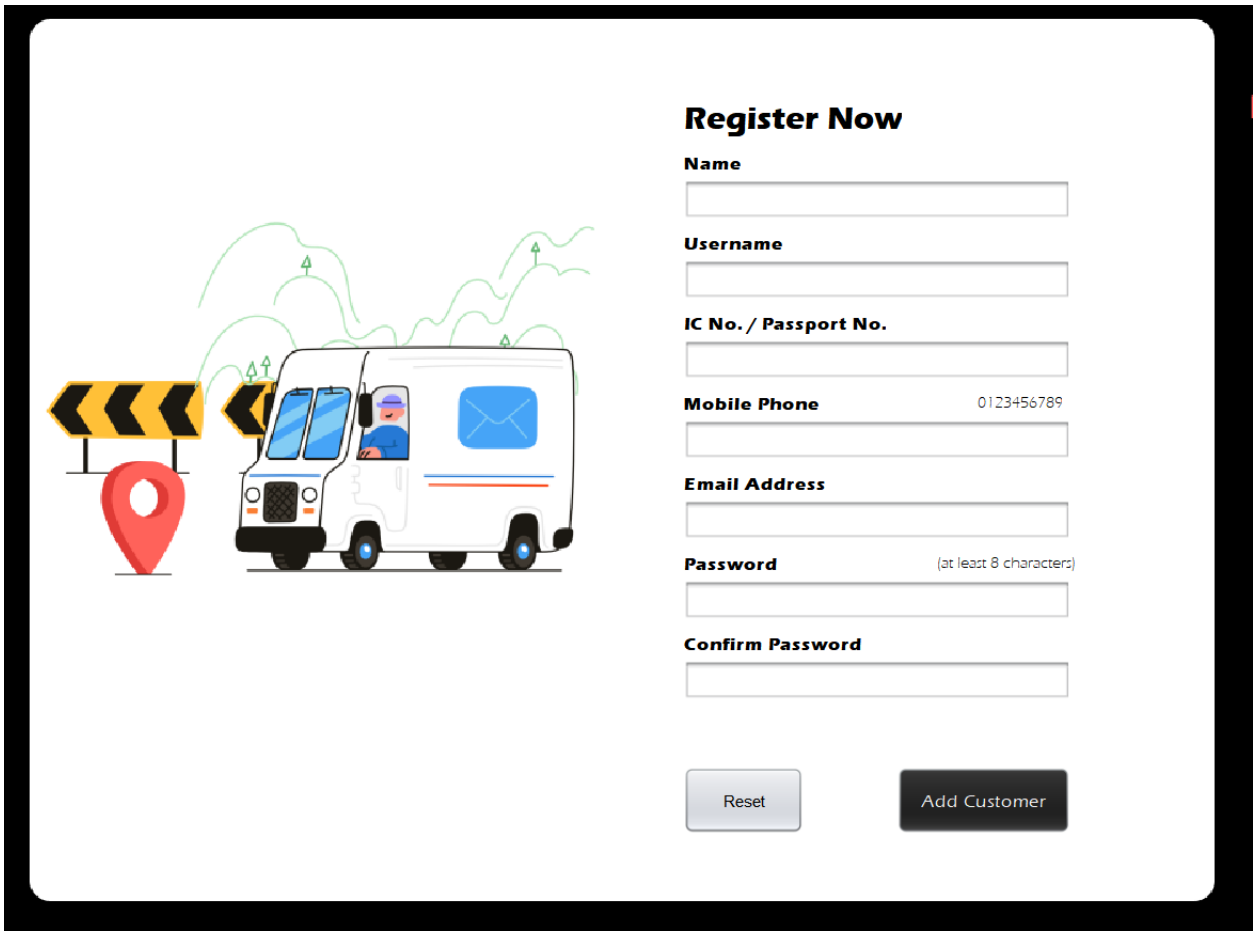


Figure 174 Manage Customer Registration

After admin click on the “Add Csutomer” button, it will redirect admin to “Registration” page wo add a new customer.

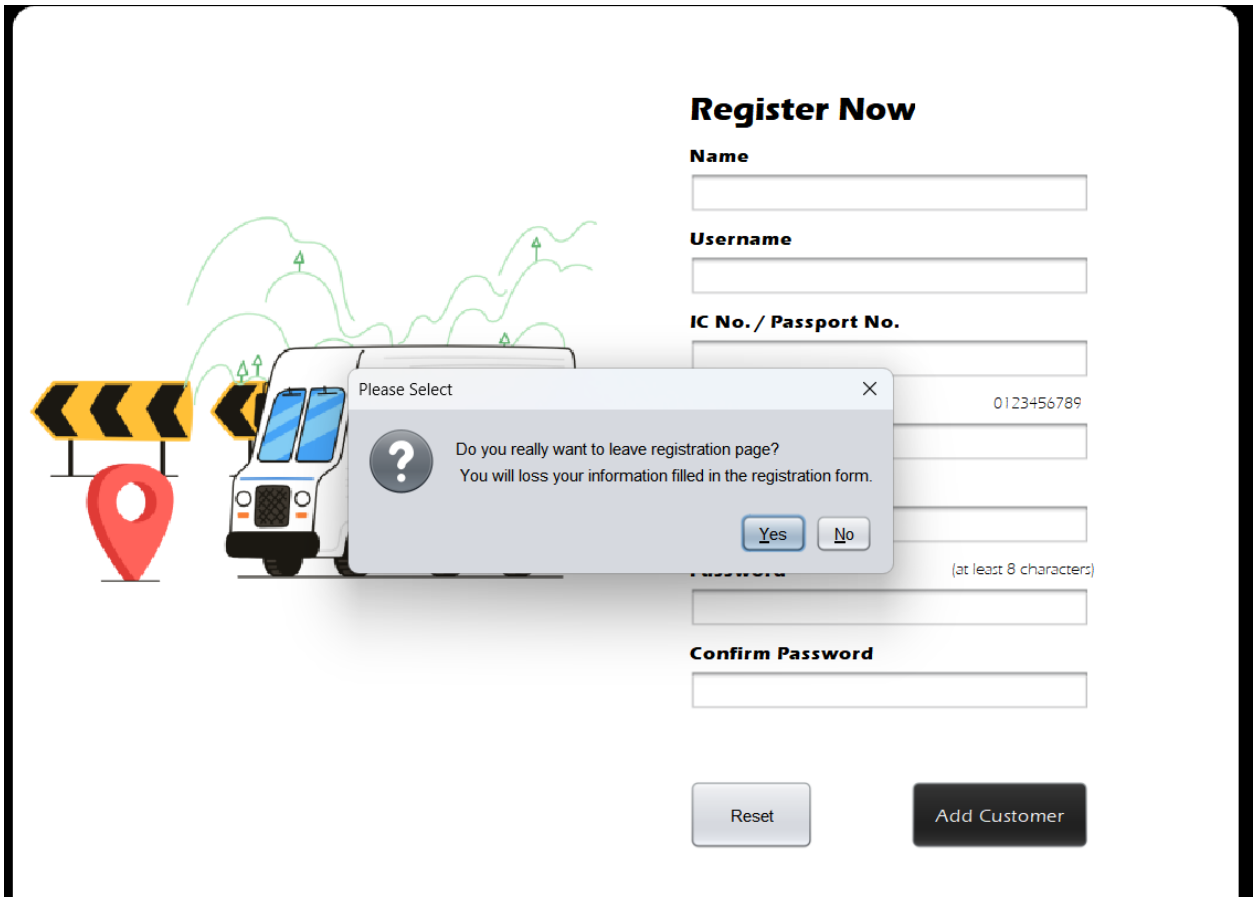


Figure 175 Manage Customer Registration

This message will show if admin try to exit this page during registration. Click “Yes” to cancel the process or click “No” to continue the registration.

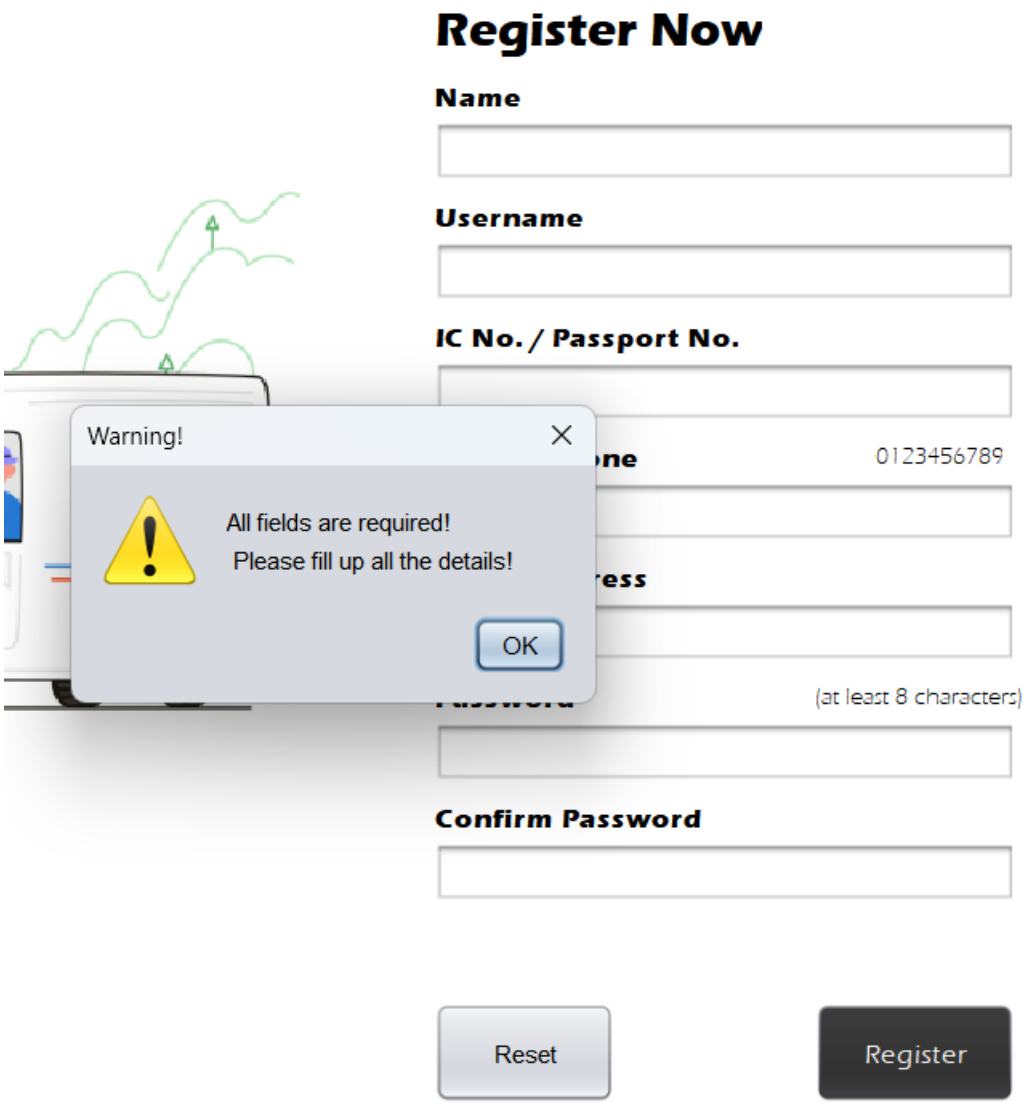


Figure 176 Manage Customer Registration

This warning notice will show if the personal information on the Registration form is incomplete. This notification instructs admin to enter all information.

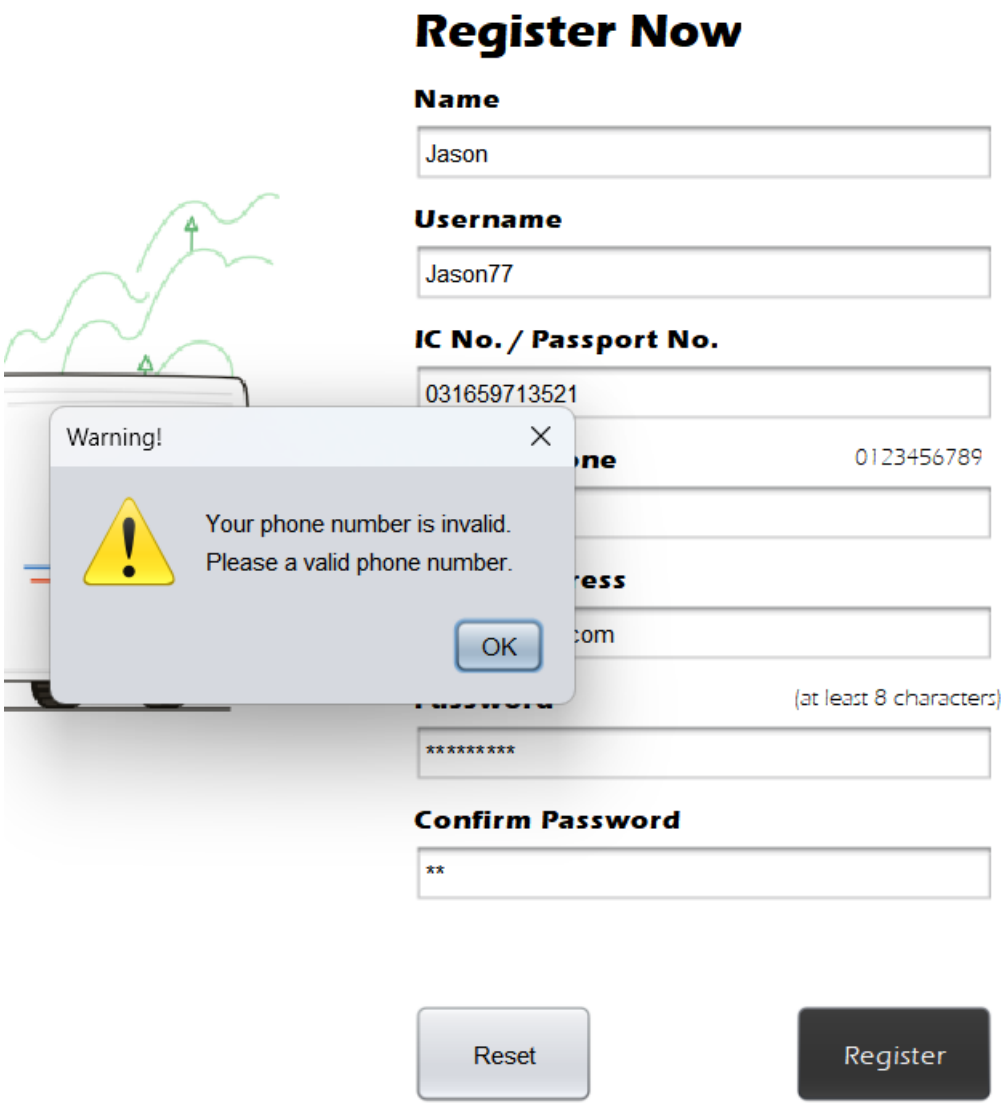


Figure 177 Manage Customer Registration

This warning notice will show if the contact number in the registration form does not contain precisely 10 digits. This notice admin to requests customer to provide a proper contact number to the system.

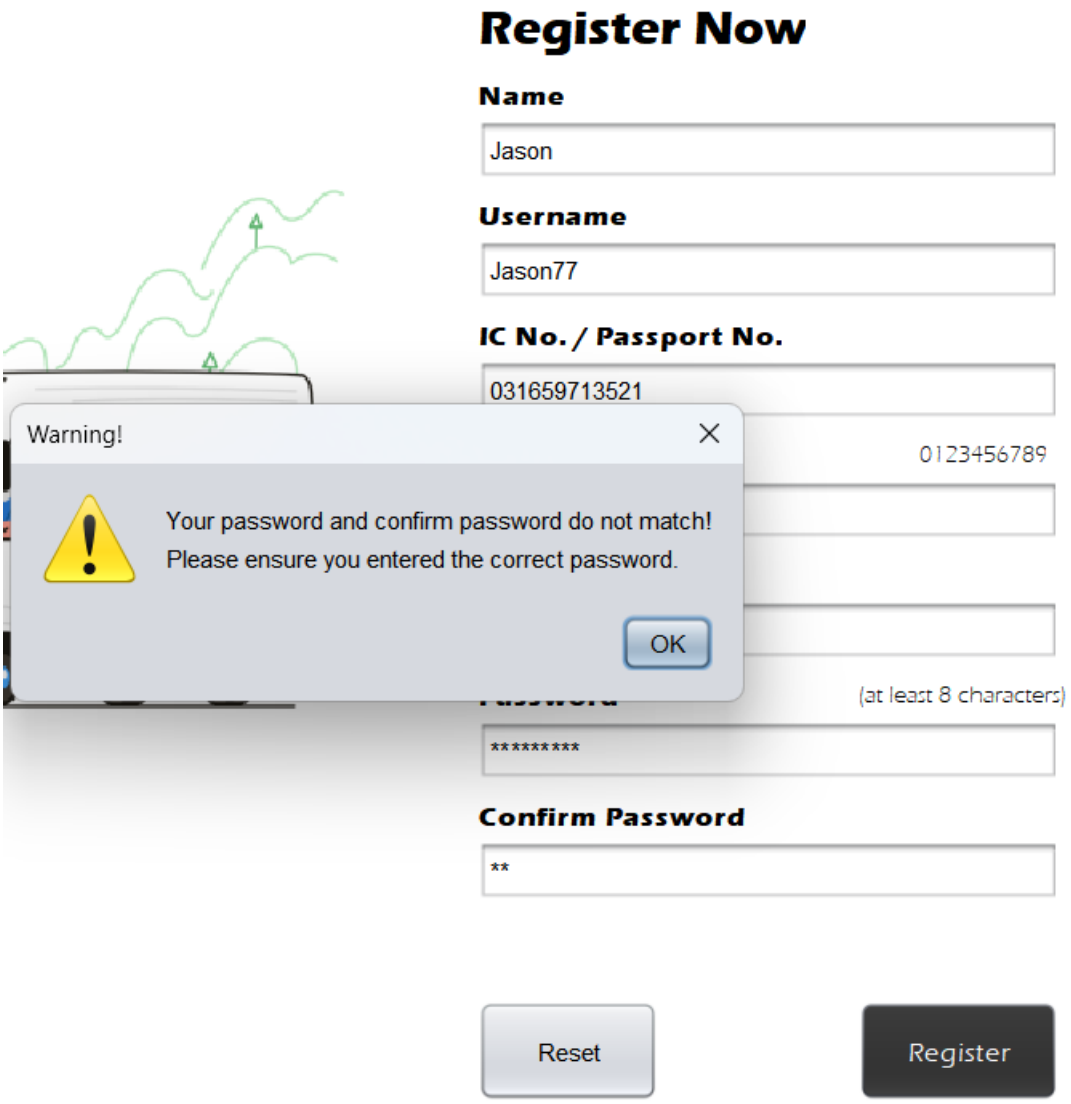


Figure 178 Manage Customer Registration

If the password and confirm password entered on the registration form do not match, this warning message will appear. This notification requests that admin key in a valid password in order to prevent typing errors.

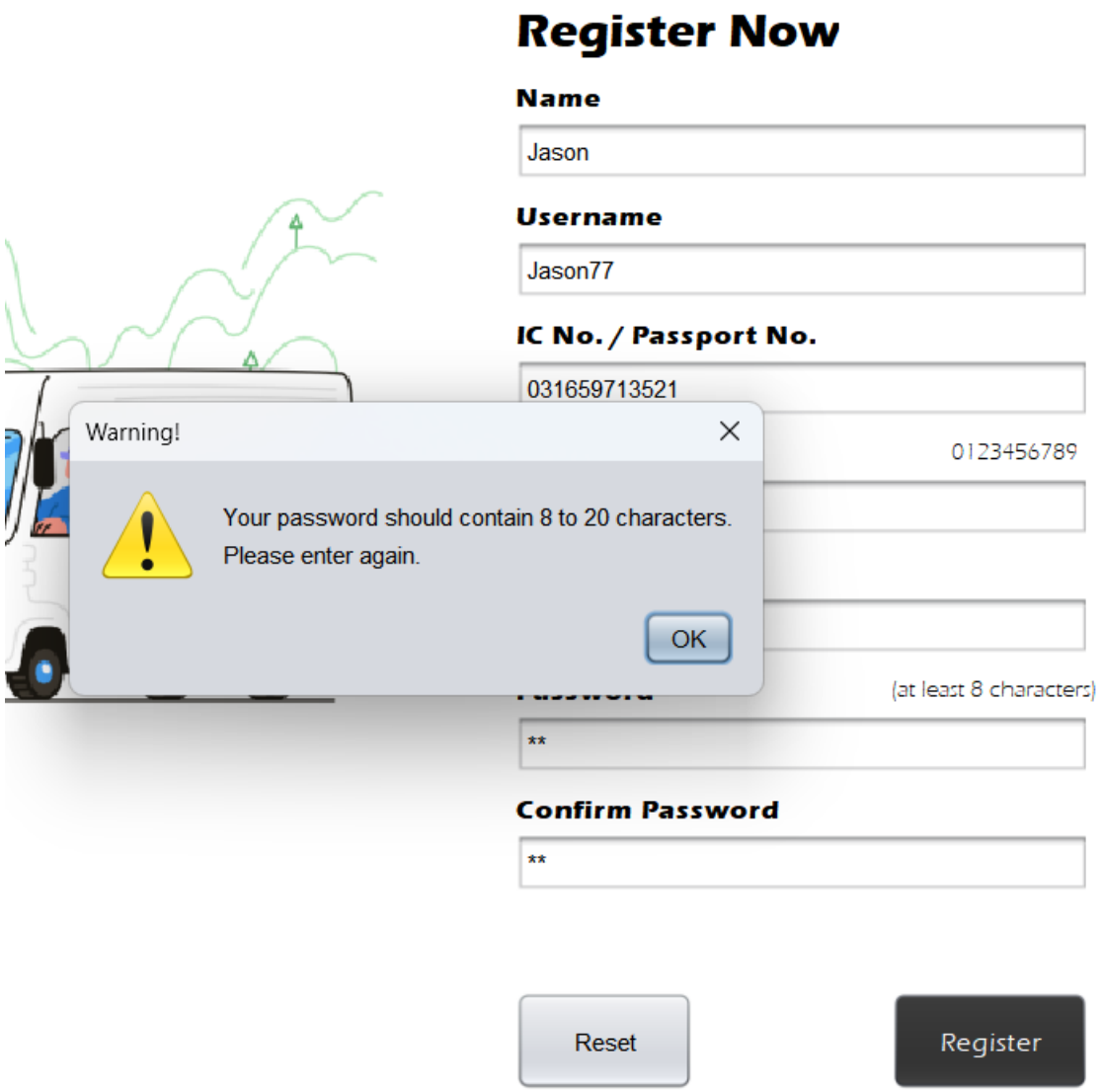


Figure 179 Manage Customer Registration

If the password entered on the registration form does not include between 8 and 20 characters, a warning message will appear. This notice requests that admin give a valid password.

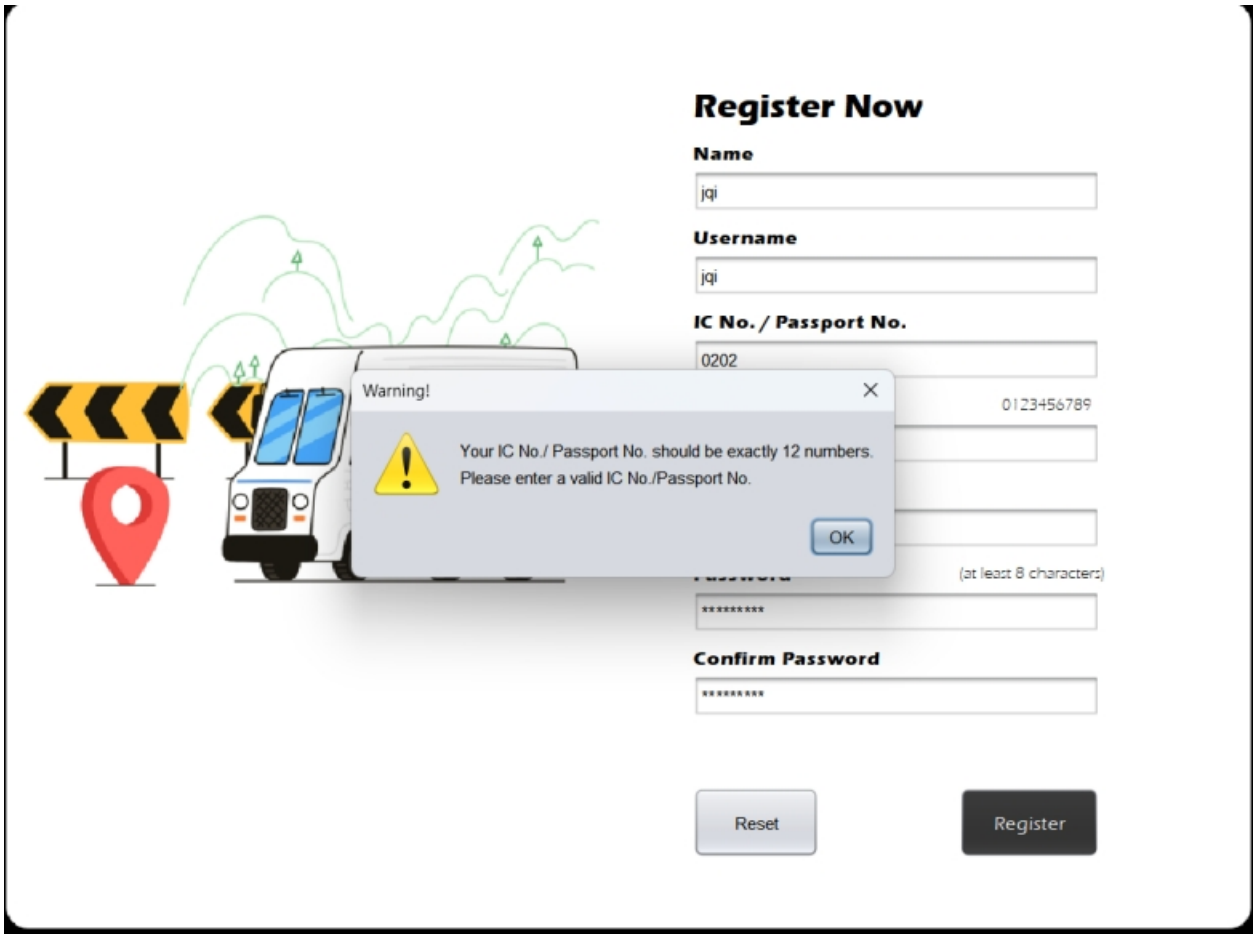


Figure 180 Manage Customer Registration

If the passport number or IC number entered on the registration form does not exactly include 12 digits, a warning message will be shown. This notice requests that customers provide a valid IC number or passport number into the system.

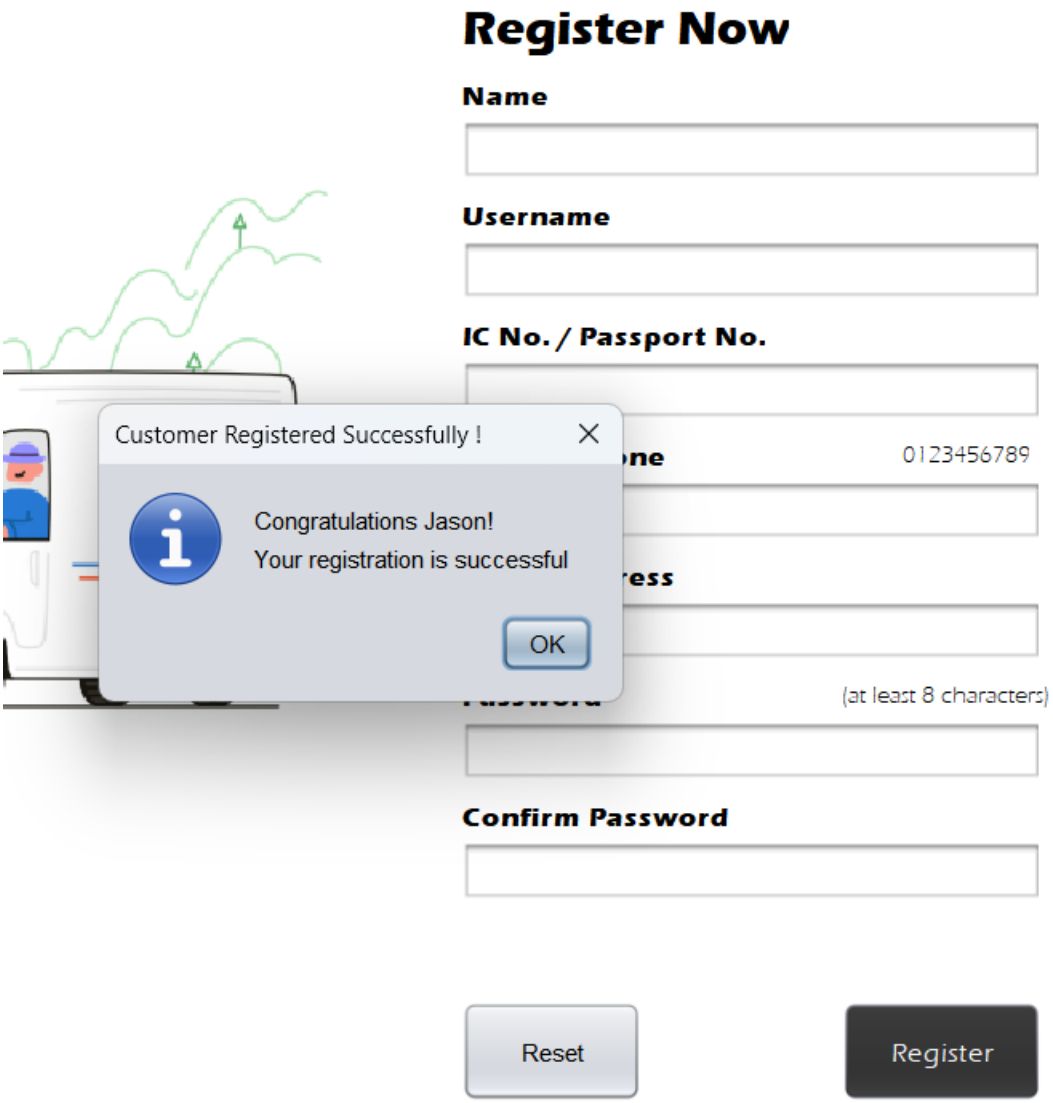


Figure 181 Manage Customer Registration

This message will appear if customers information is valid, click "Register" to proceed to the "Login page."

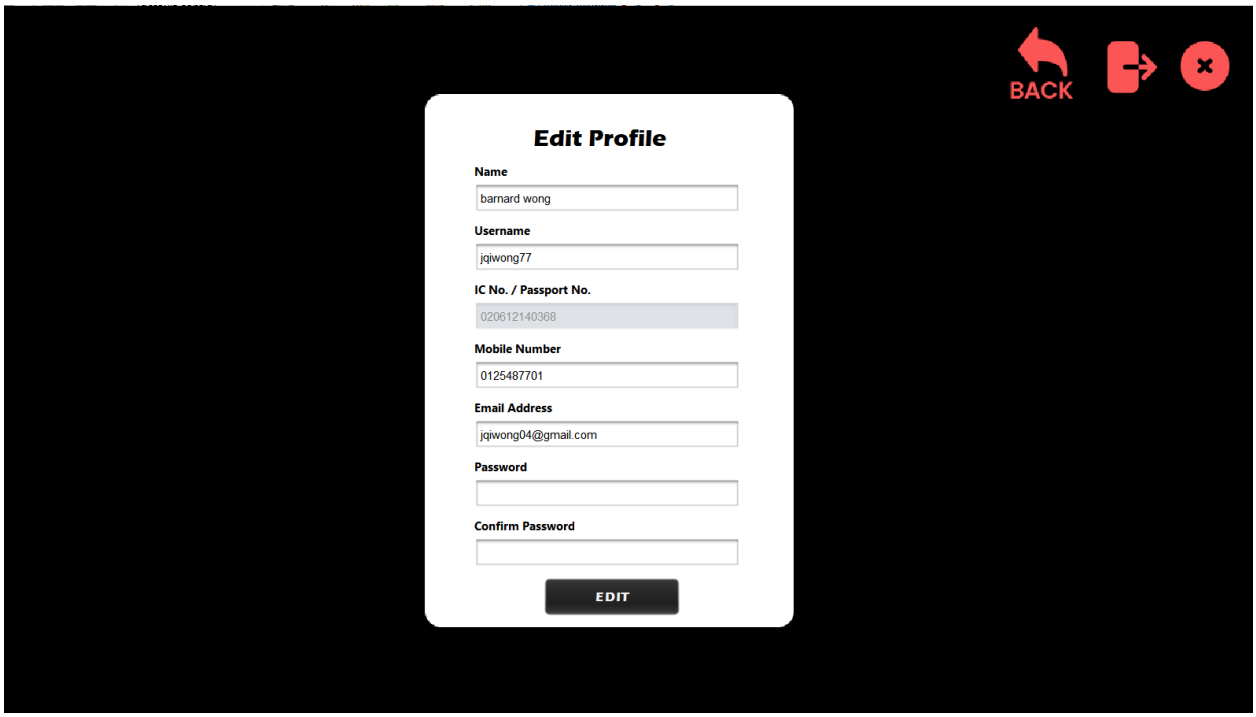


Figure 182 Manage Customer Registration

This page is “Edit Profile” page which is for admin to update their customer information except IC number.

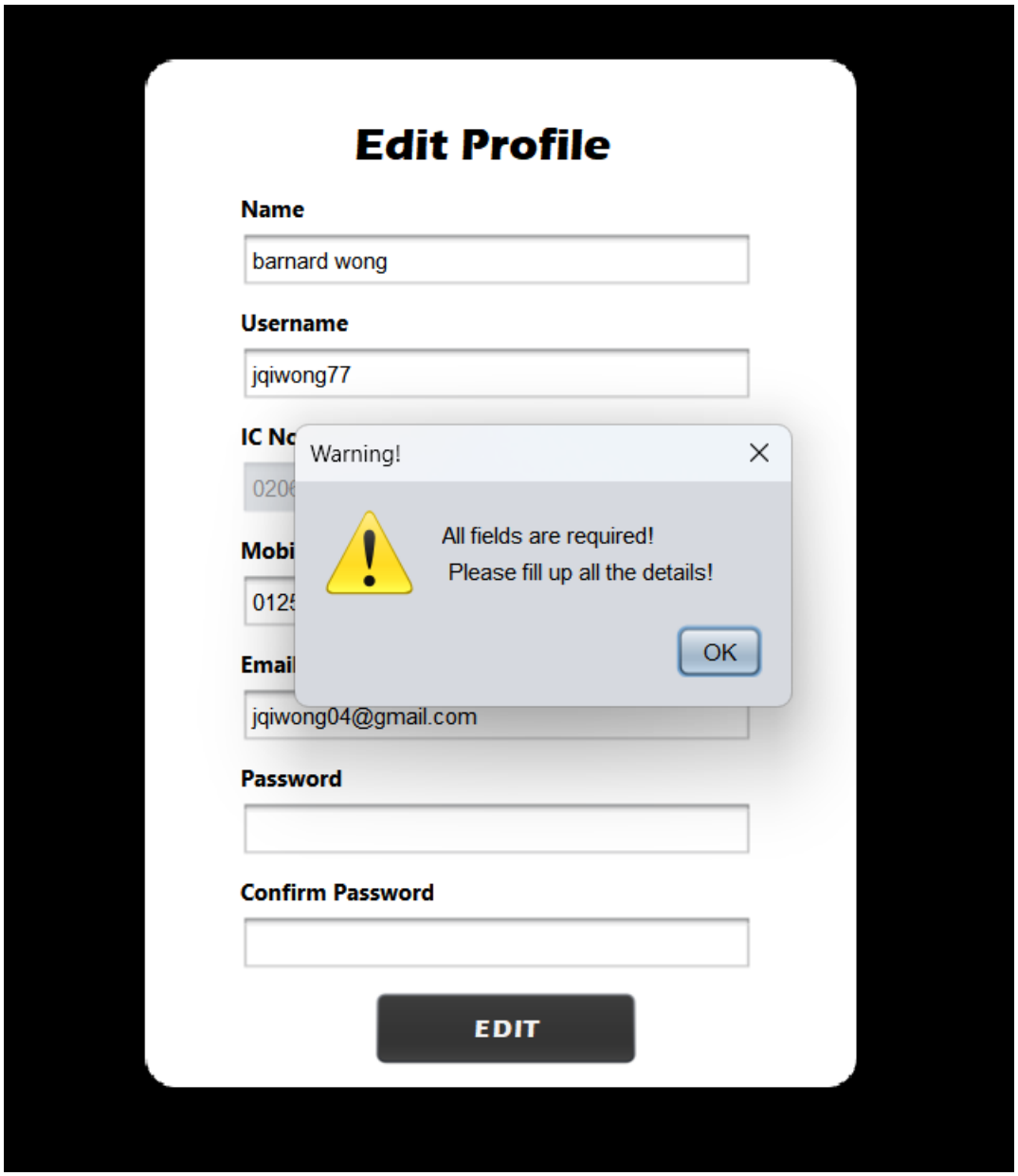


Figure 183 Manage Customer Registration

This warning notice will show if the customer information on the Edit Profile page is incomplete. This notification instructs admin to enter all information.

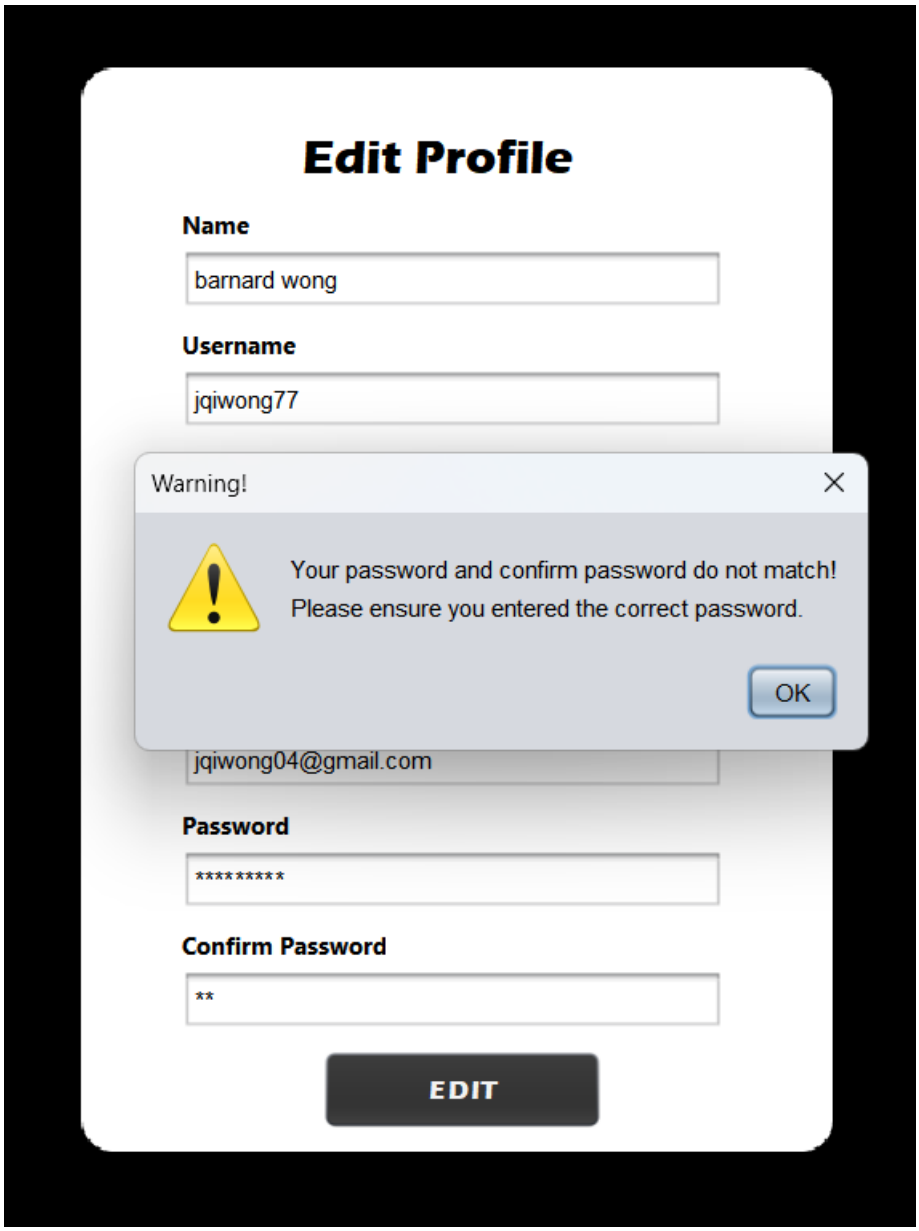


Figure 184 Manage Customer Registration

If the password and confirm password used on the Edit Profile page do not match, this warning message will appear. This notification requests that admin key in a valid password in order to prevent typing errors.

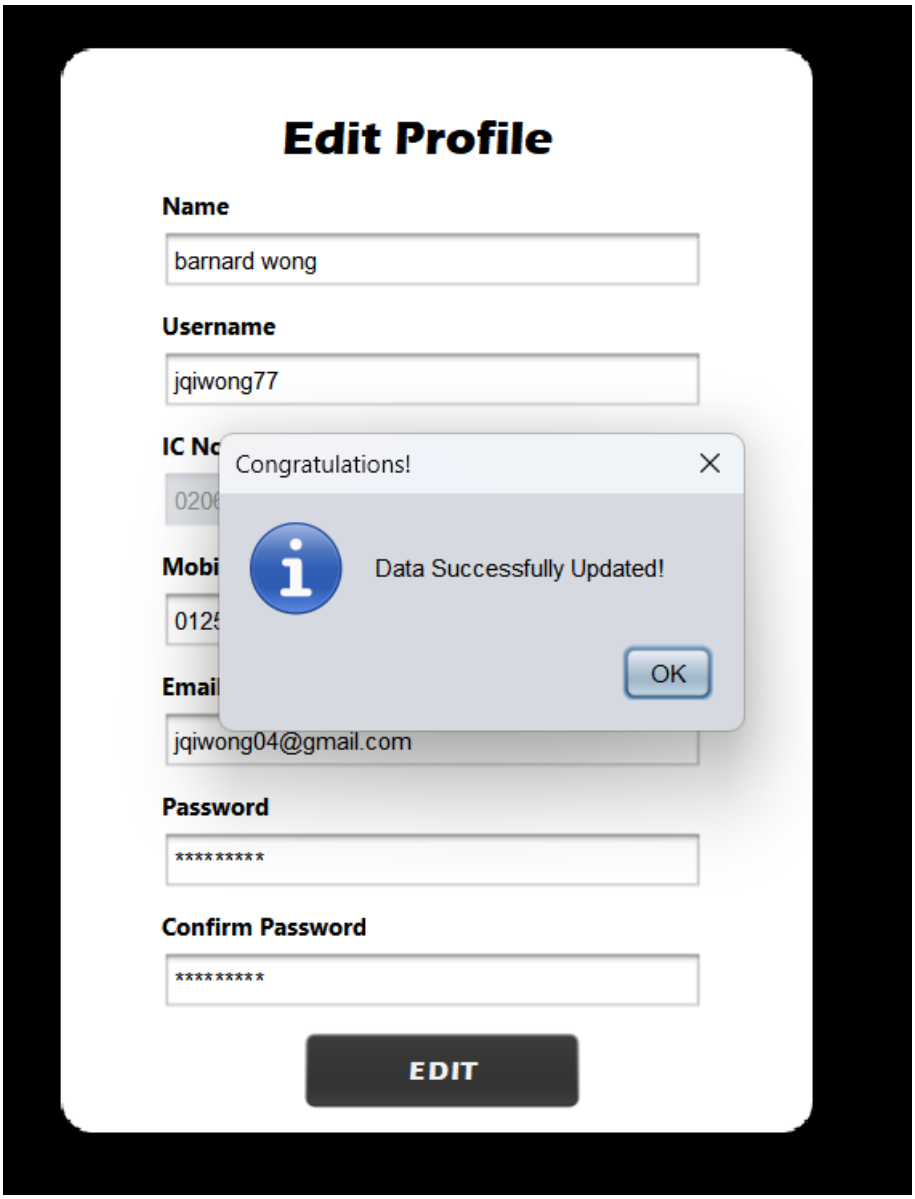


Figure 185 Manage Customer Registration

After entering all the valid information, the system will update the information and show this message by pressing the “EDIT” button.

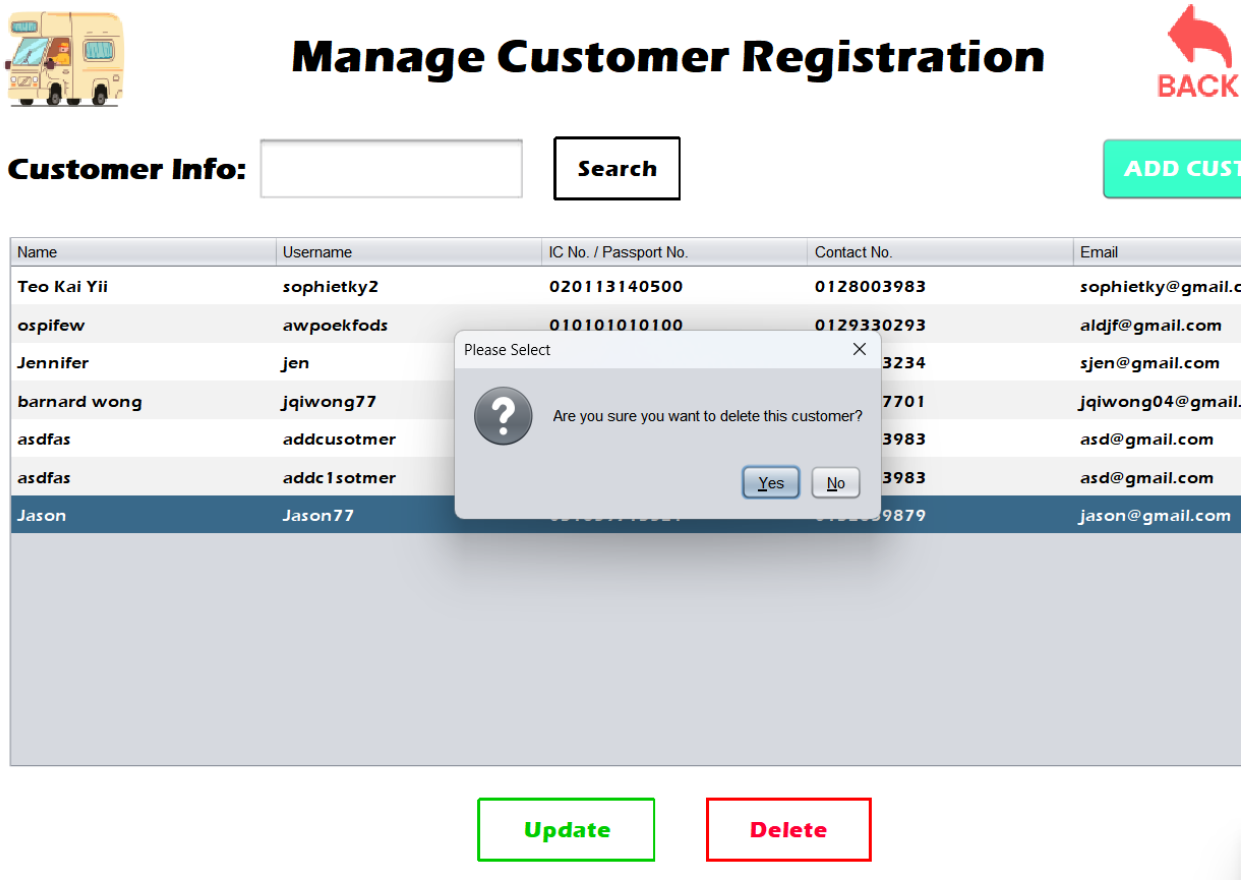


Figure 186 Manage Customer Registration

This message will show for admin to confirm their decision if admin click on “Delete Booking” button. Press “Yes” to delete the customer from text file otherwise click “No” to cancel the process.

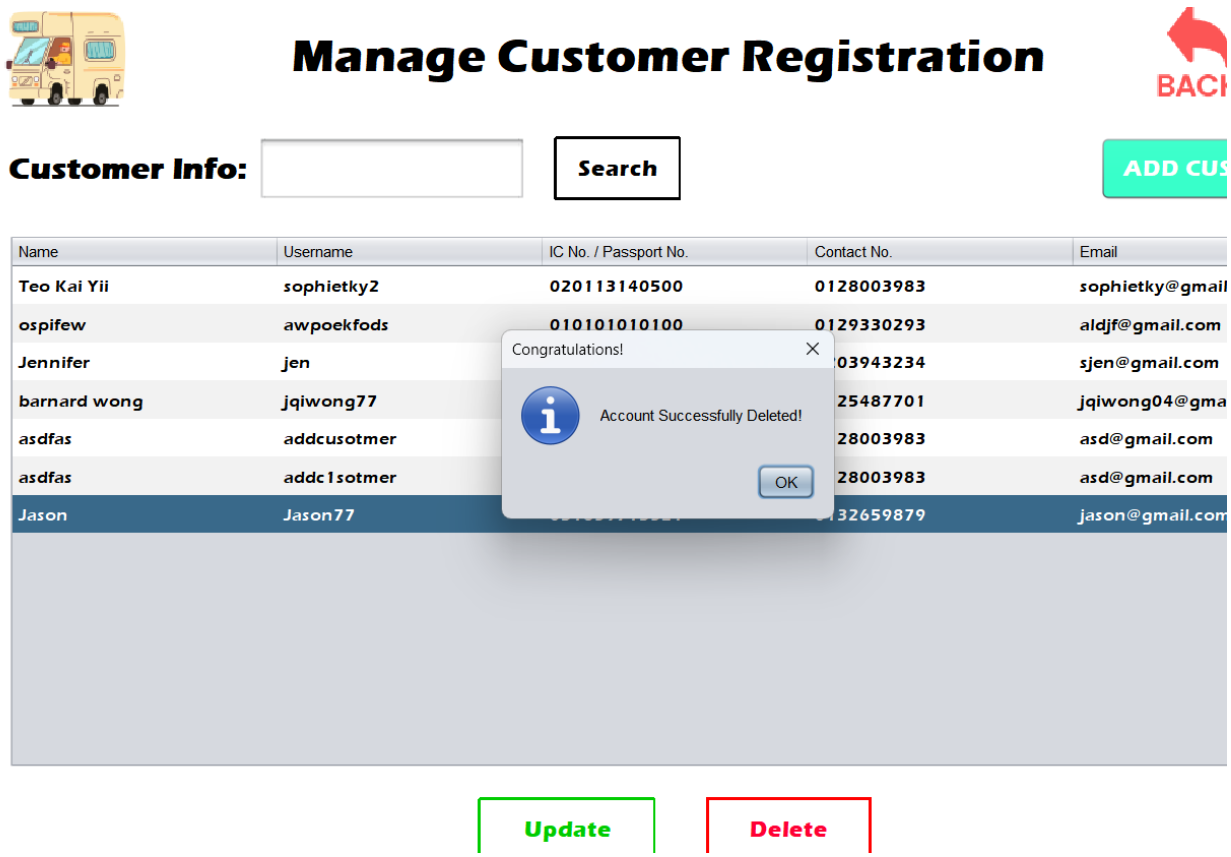


Figure 187 Manage Customer Registration

This message will show if the customer deleted successfully.

5.4.11 Return A Car

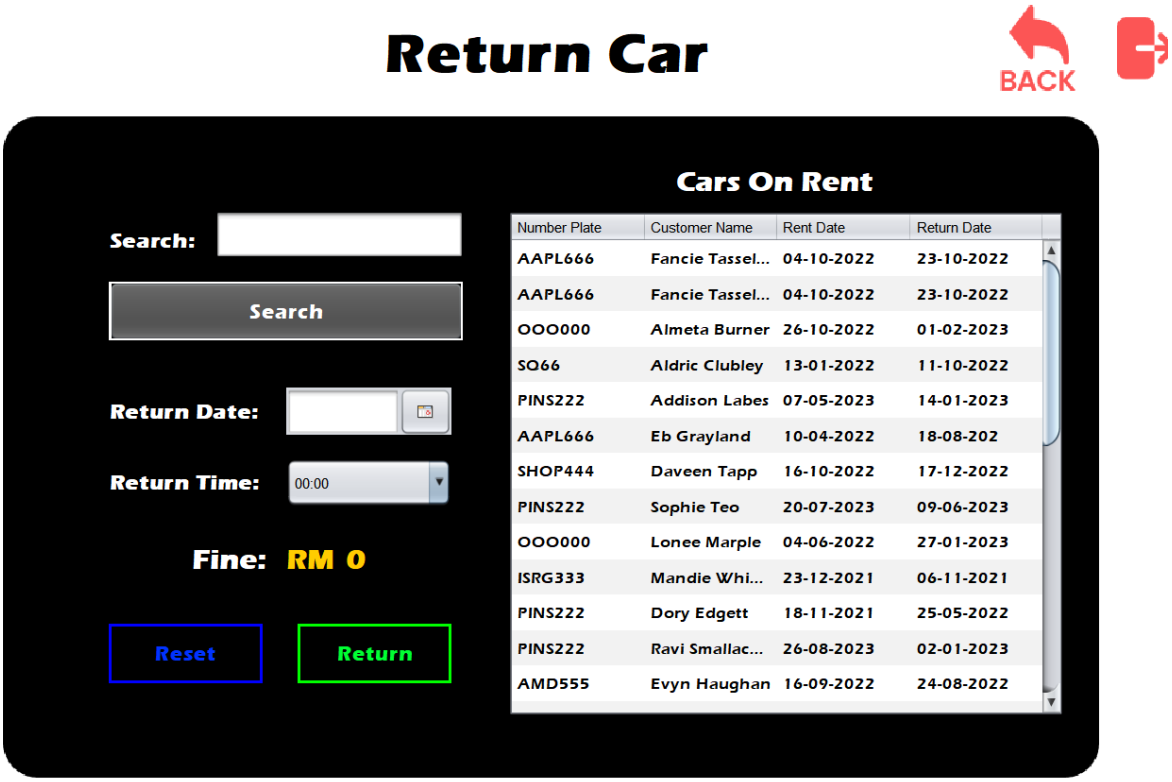


Figure 188 Return A Car

This is “Return Car” page where for admin to change the status after customer return the car.

Return Car

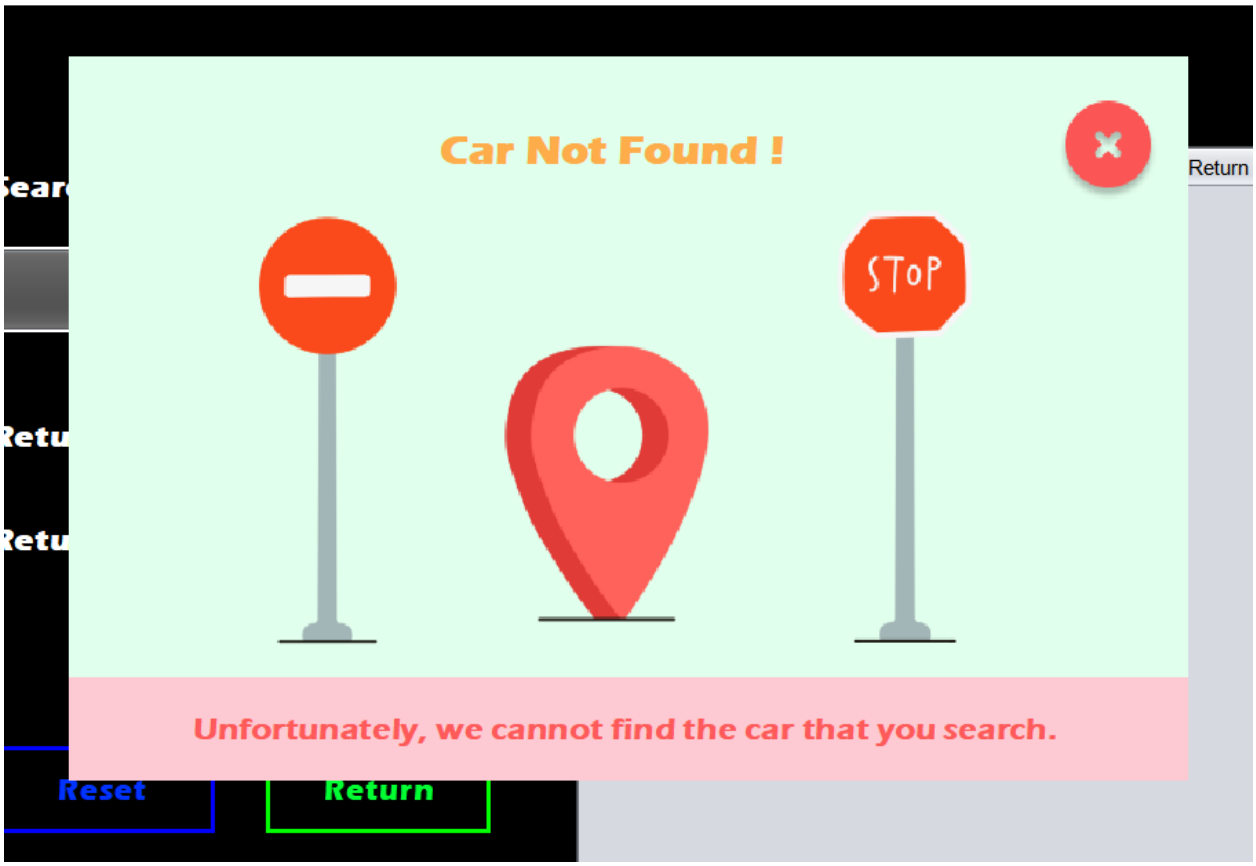


Figure 189 Return A Car

This message will show if there are no car found according to the search.

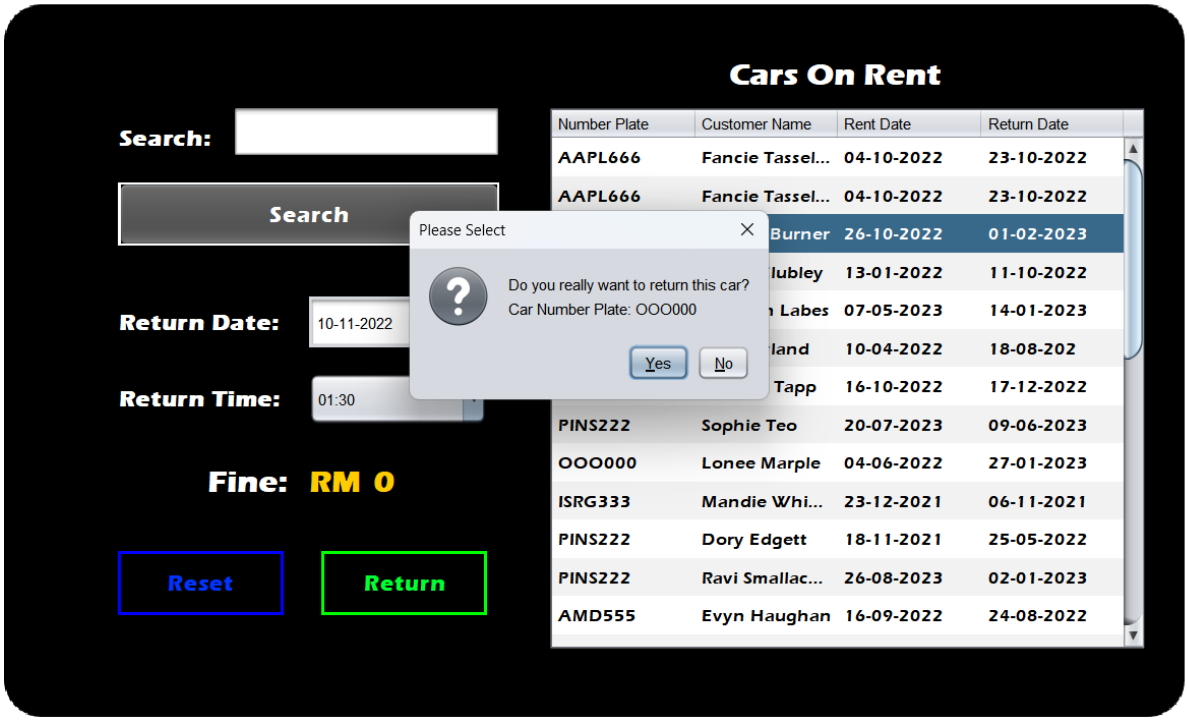


Figure 190 Return A Car

This message will show for admin to confirm their decision if customer click on “Return” button. Press “Yes” to change the car status otherwise click “No” to cancel the process.

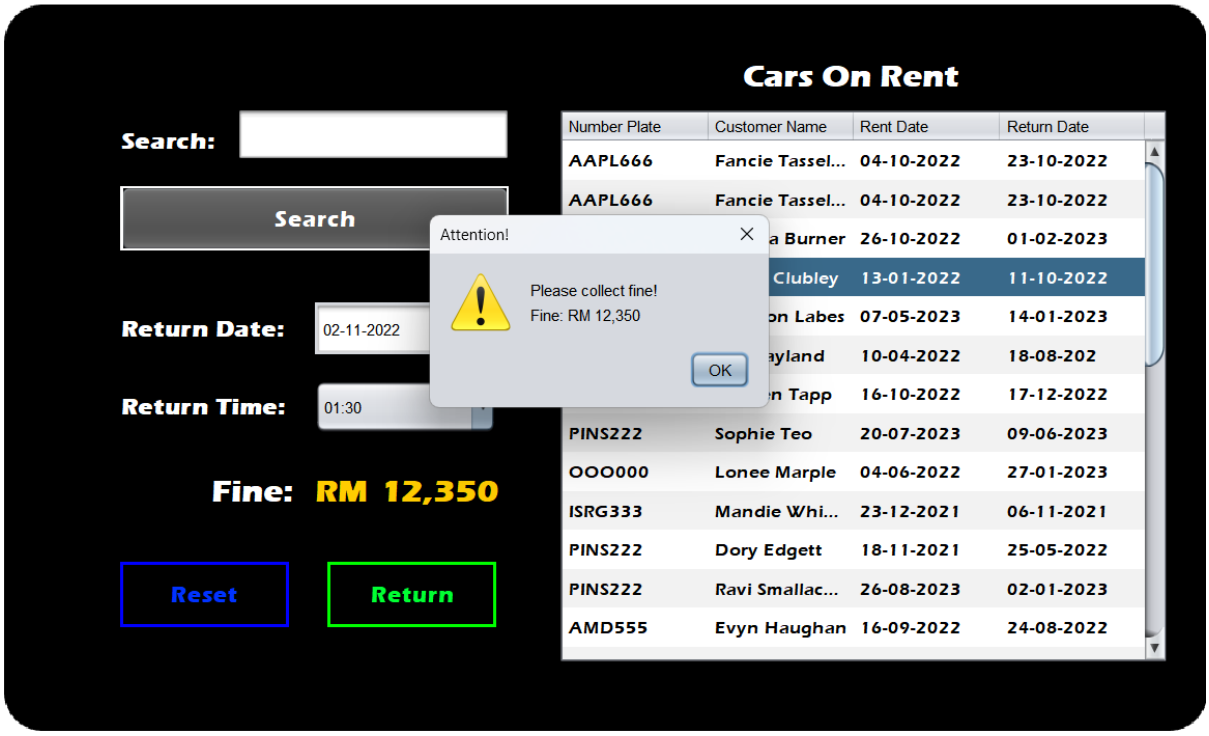


Figure 191 Return A Car

This warning message will show if the customer late to return the car and calculate the fine for customers.

5.4.12 Reports

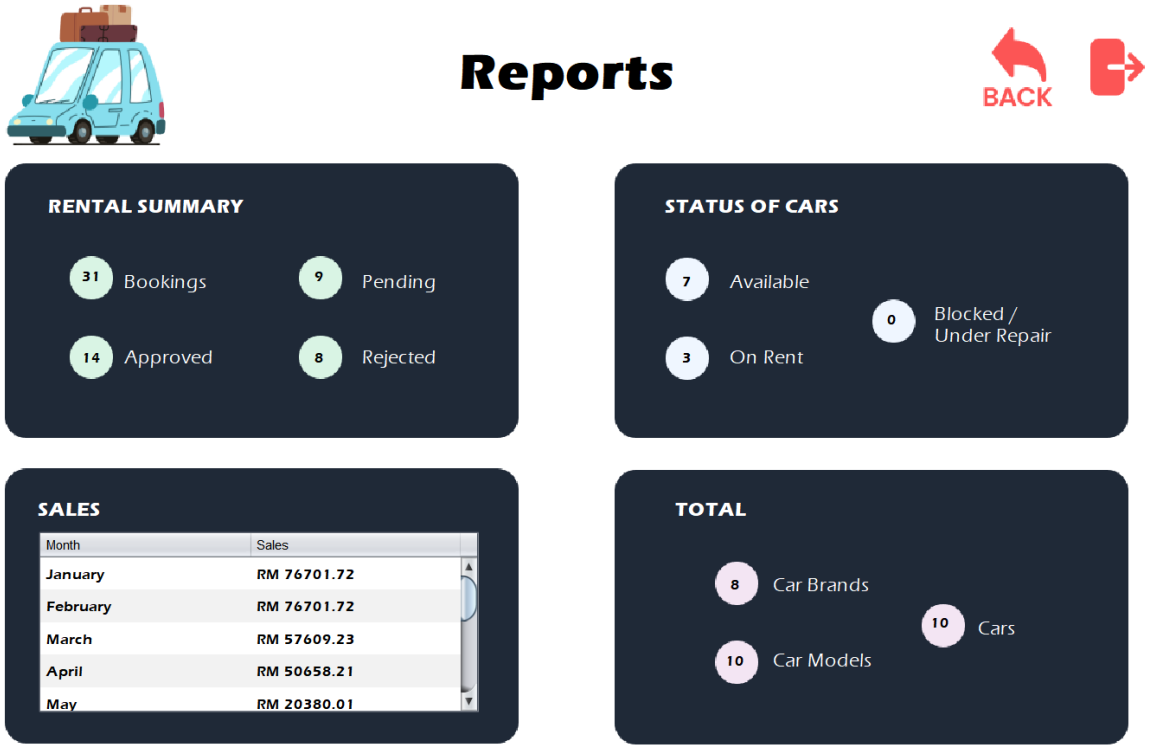


Figure 192 Reports

This is the “Report” page where admin can view then rental summary, status of car, monthly sales and total car brands, car models and car of the system have.

5.5 Default Admin

5.5.1 Default Admin Extra Function

Figure 193 Default Admin Extra Function

This page is “Admin Menu” page and this two extra functions are only assessable for default admin which are "Manage Administrator Registration” and “Login Record”.

5.5.2 Manage Administrator Registration

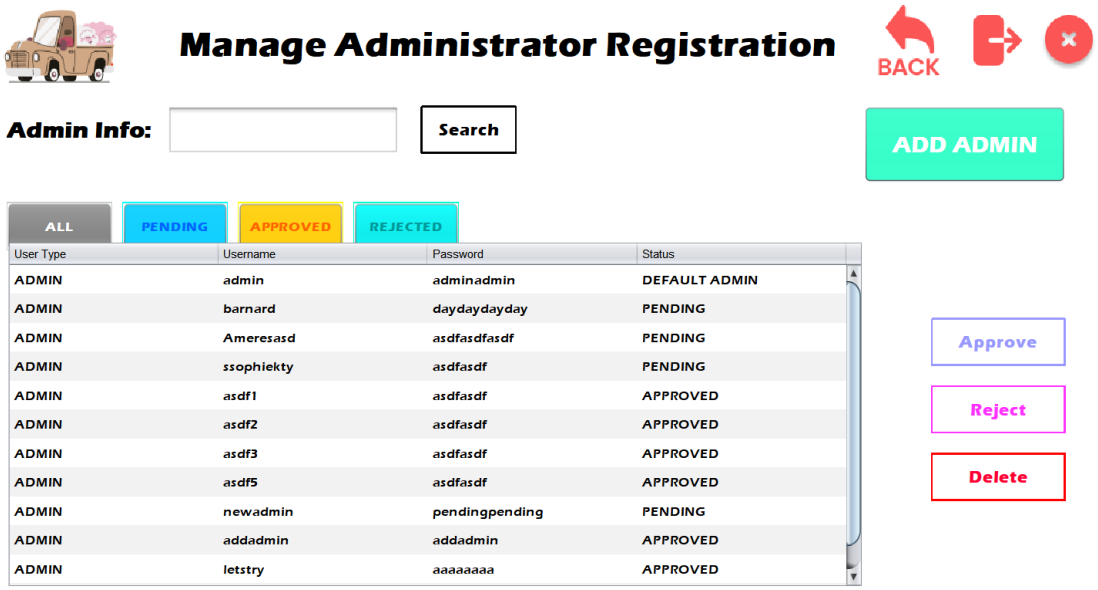


Figure 194 Manage Administrator Registration

This is "Manage Administrator Registration” page where default admin can approve, reject and delete the admin account.

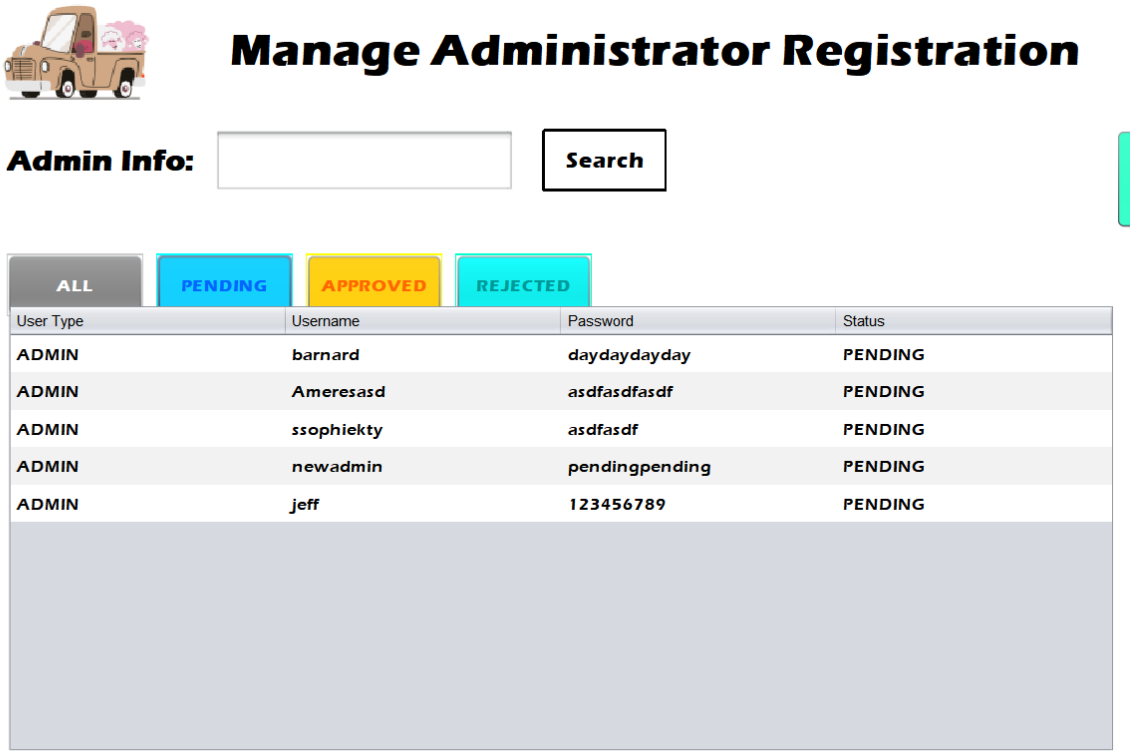


Figure 195 Manage Administrator Registration

Admin can click on the “PENDING” button to sort the admin account status and system will show all accounts are pending.

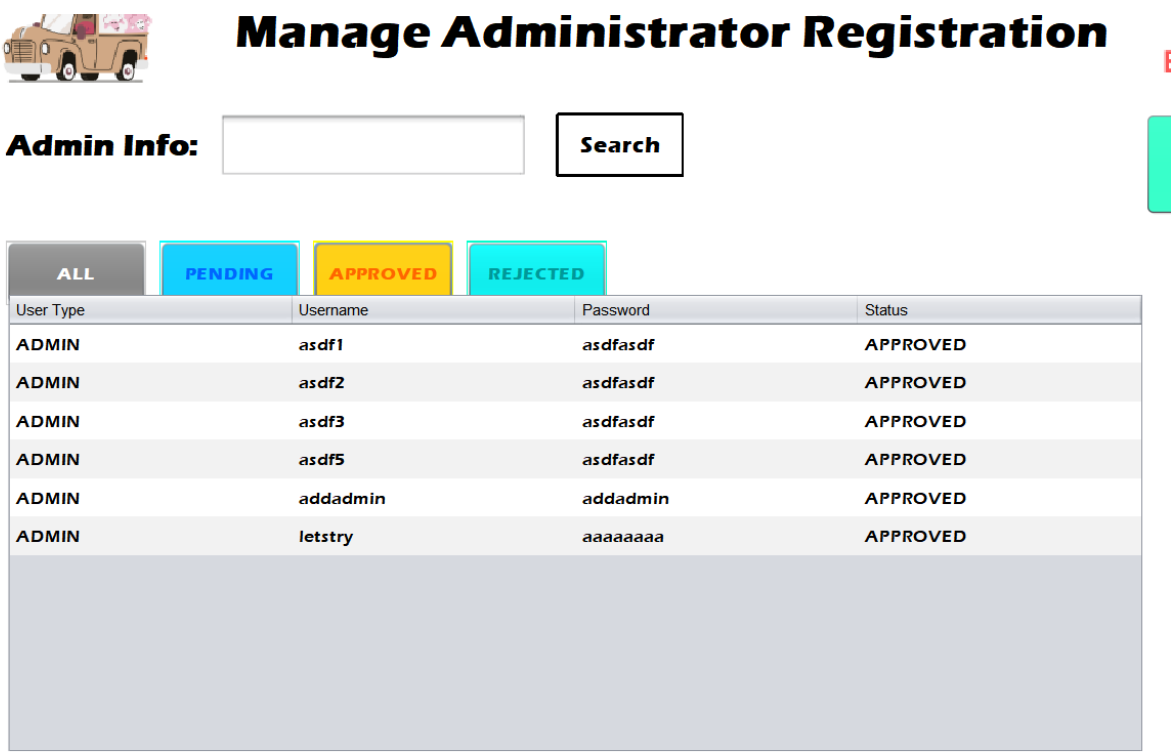


Figure 196 Manage Administrator Registration

Admin can click on the “APPROVED” button to sort the admin account status and system will show all accounts are approved.

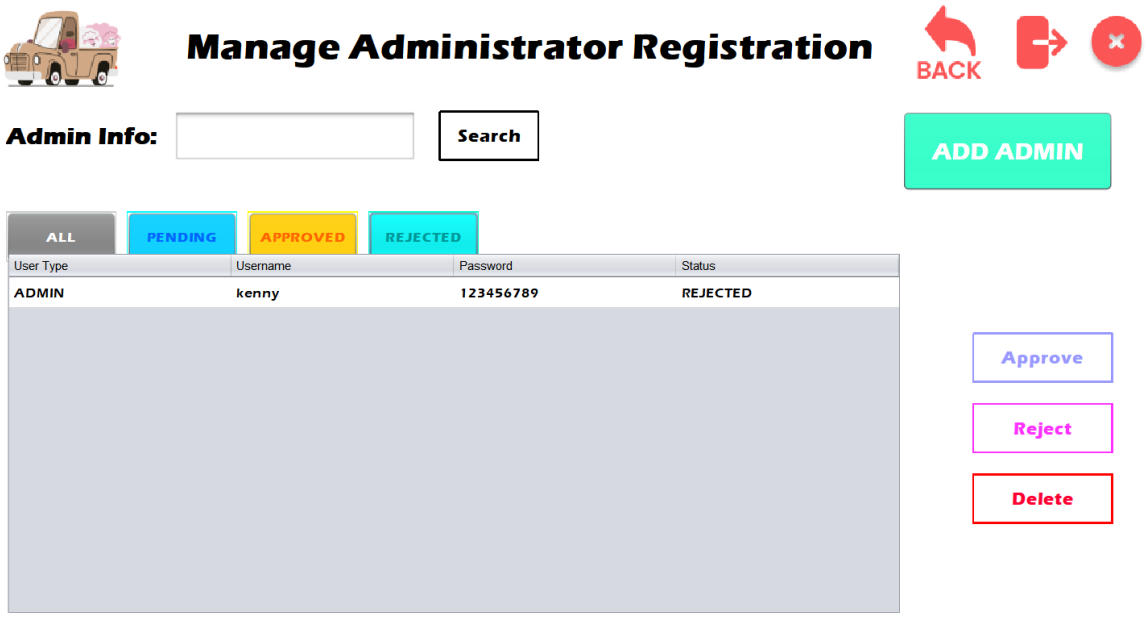


Figure 197 Manage Administrator Registration

Admin can click on the “REJECTED” button to sort the admin account status and system will show all accounts are rejected.

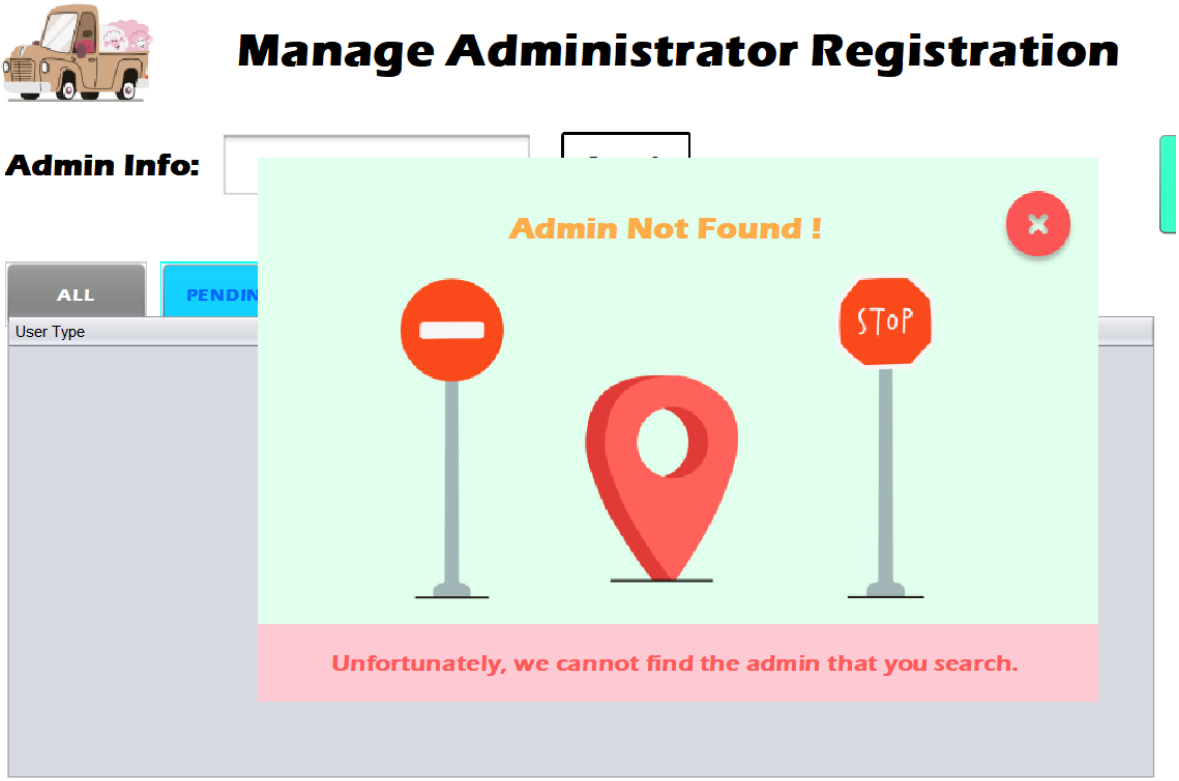


Figure 198 Manage Administrator Registration

This message will show if there are no admin account found according to the search.

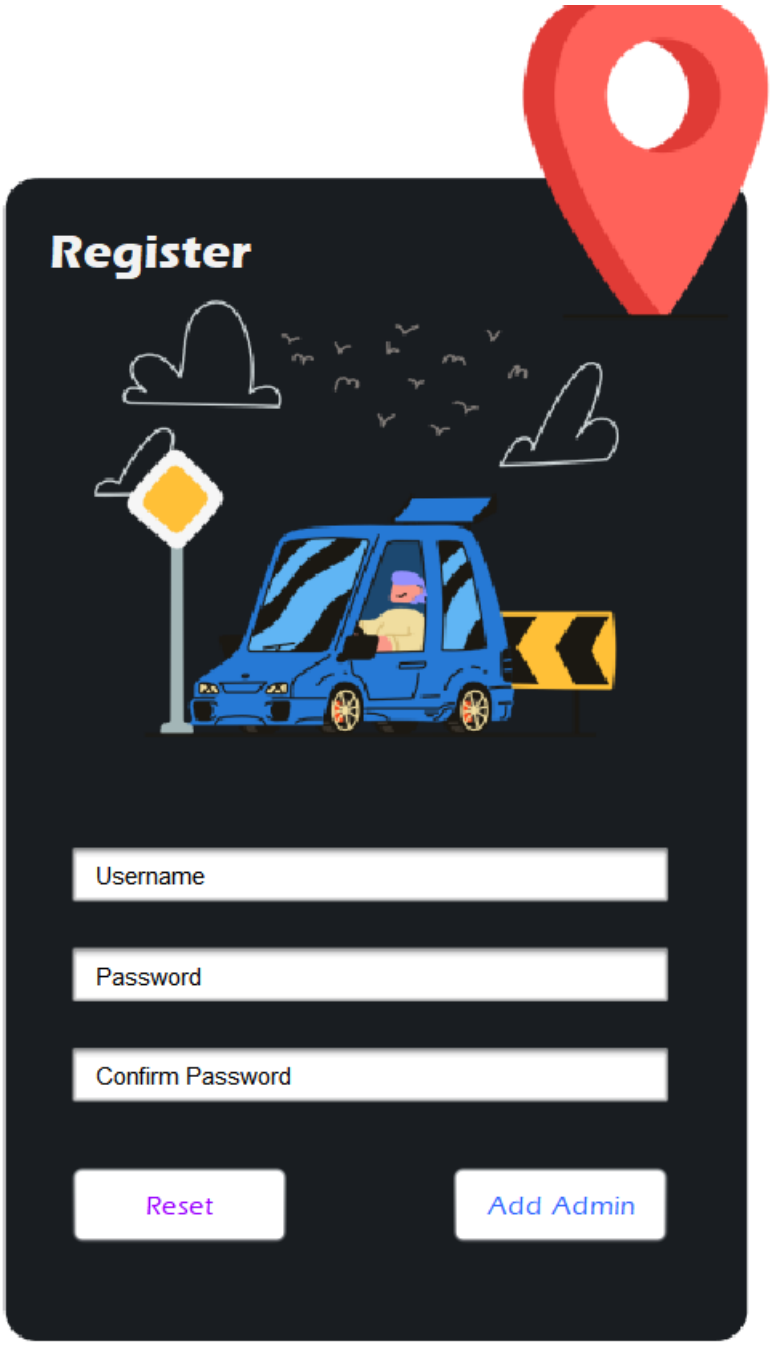


Figure 199 Manage Administrator Registration

This page will show if admin click on “Add Admin” button to add a new admin account.

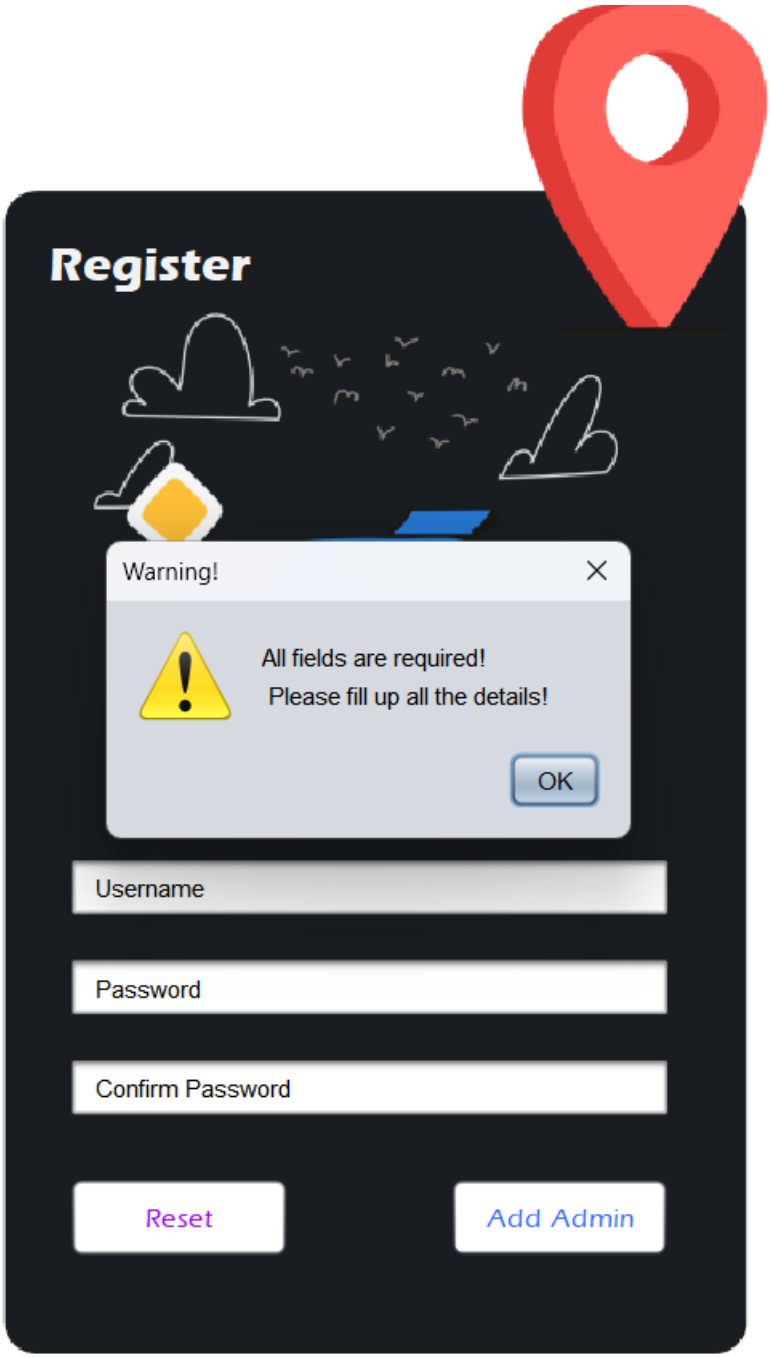


Figure 200 Manage Administrator Registration

This warning notice will show if the admin information on the Registration form is incomplete. This notification instructs admin to enter all information.

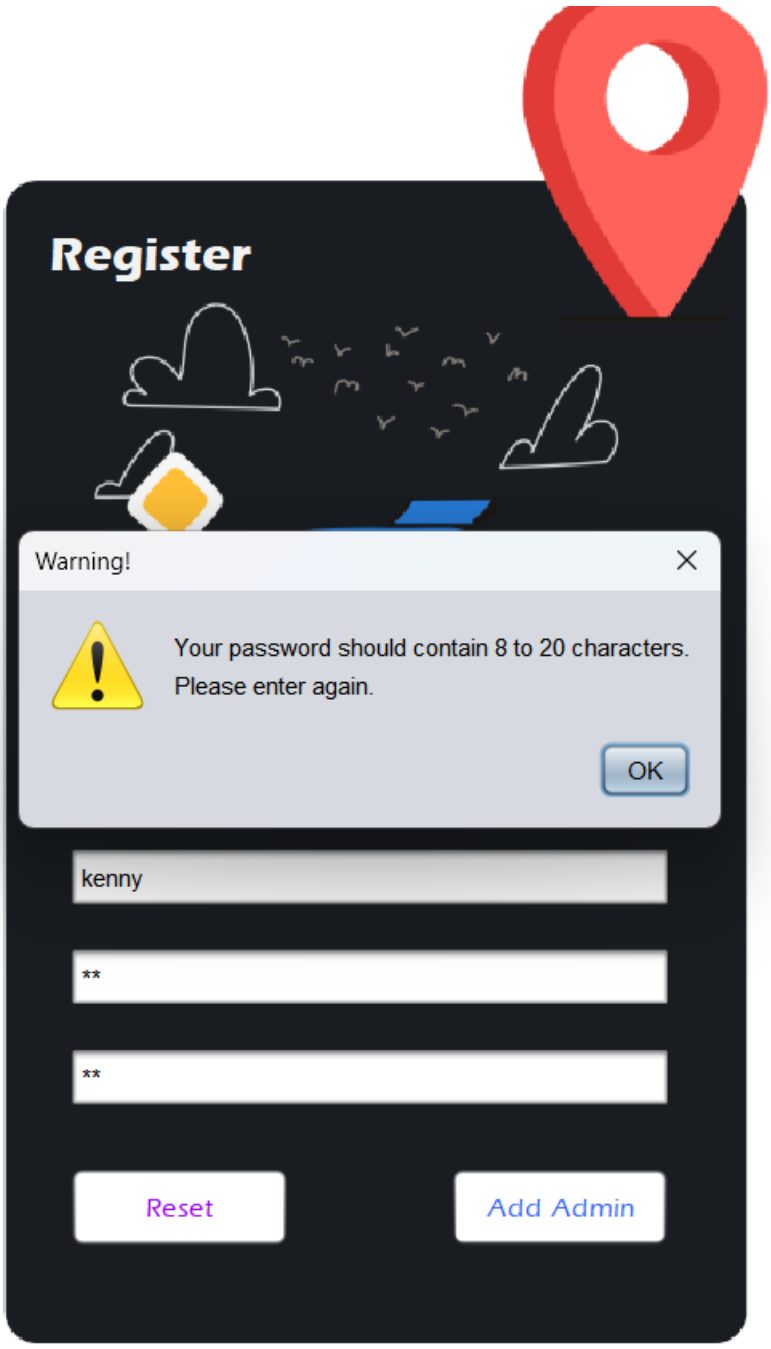


Figure 201 Manage Administrator Registration

This warning notice will show if the password used in the registration form does not include 8 to 20 characters. This notice requests that admin give a valid password.

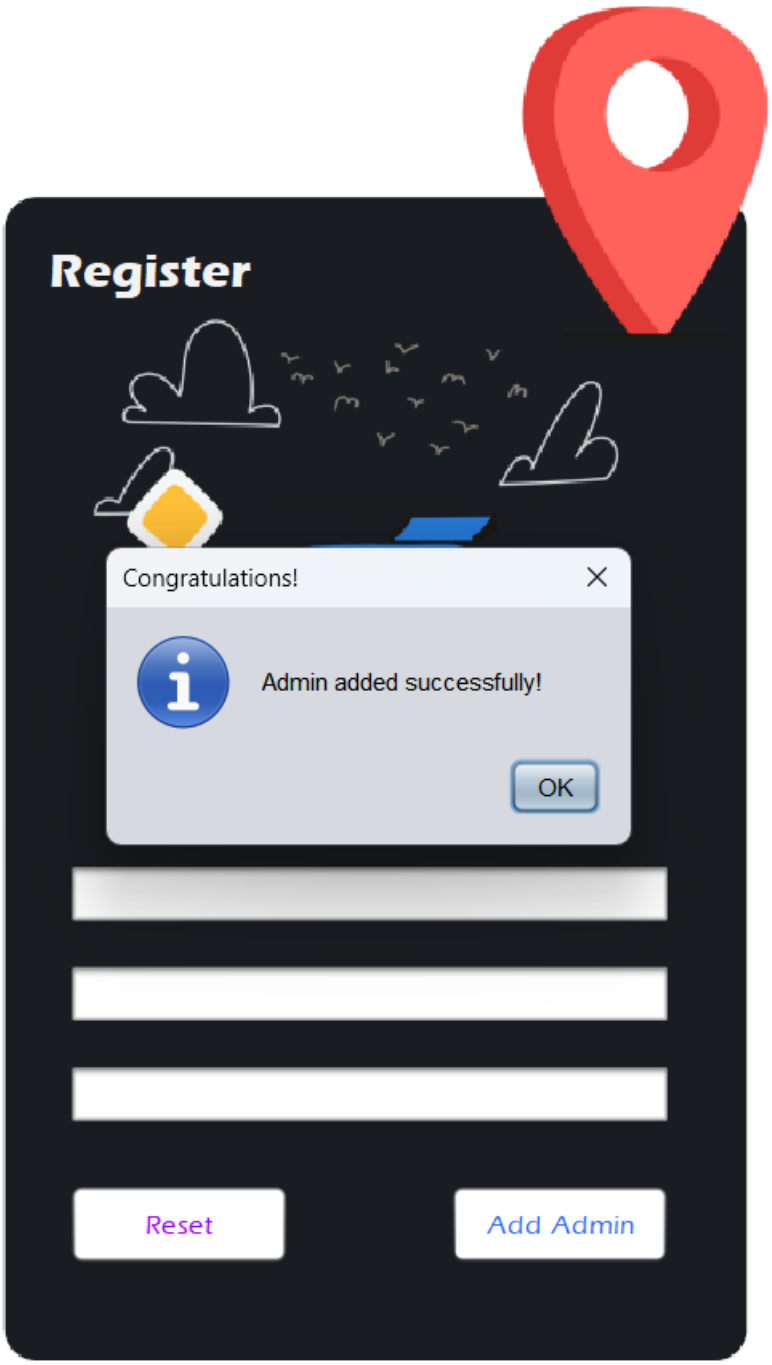


Figure 202 Manage Administrator Registration

This message will appear if admin information is valid, click "Register" to proceed to the "Login page."

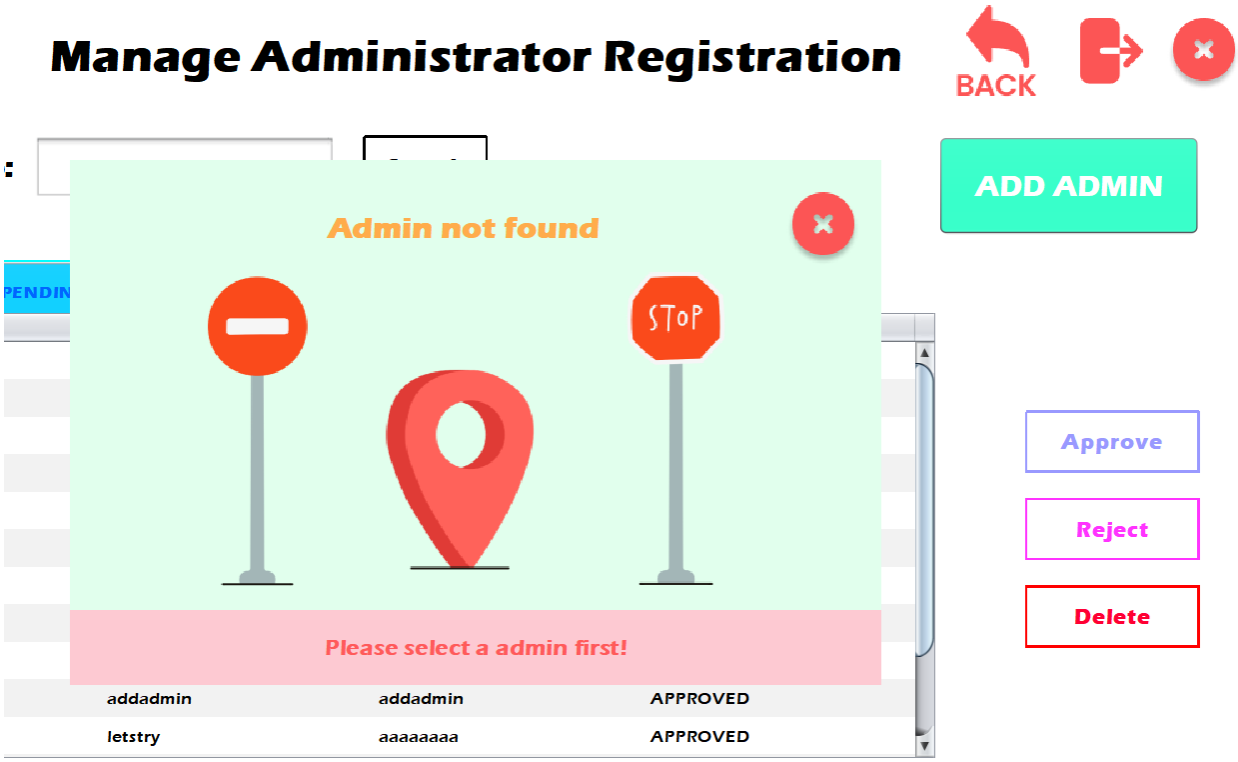


Figure 203 Manage Administrator Registration

This message will show if admin click on “Approve” button, “Reject” button or “Delete” button without select any admin account.

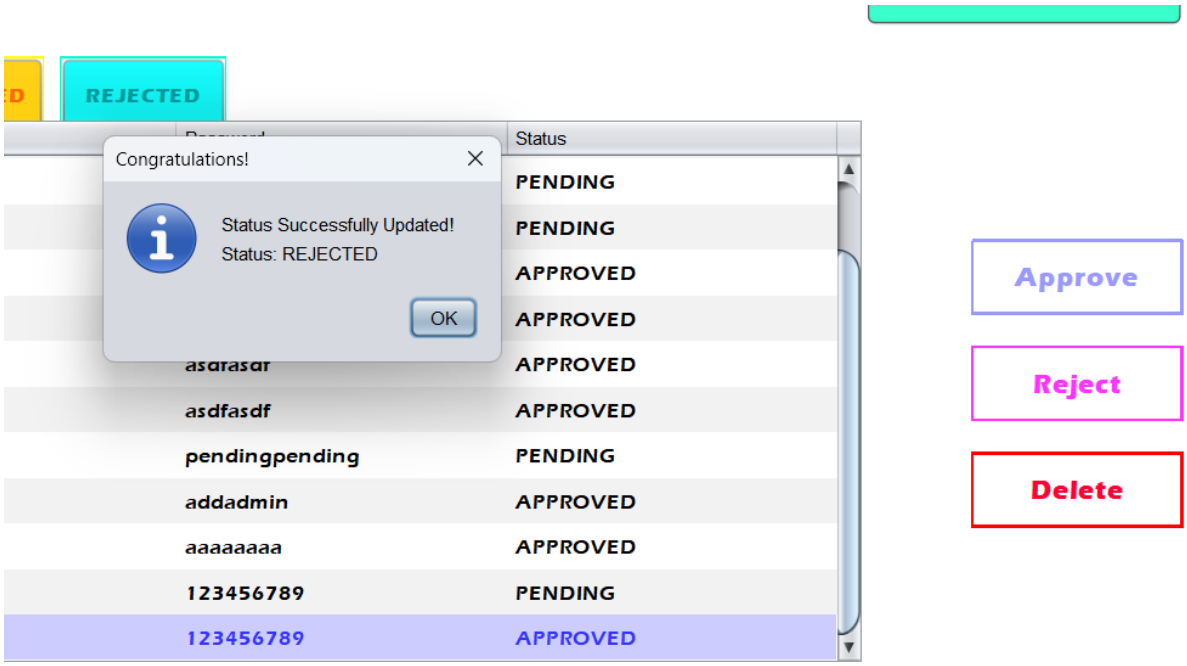


Figure 204 Manage Administrator Registration

This message will show if admin selected an account and click on “Reject” button.

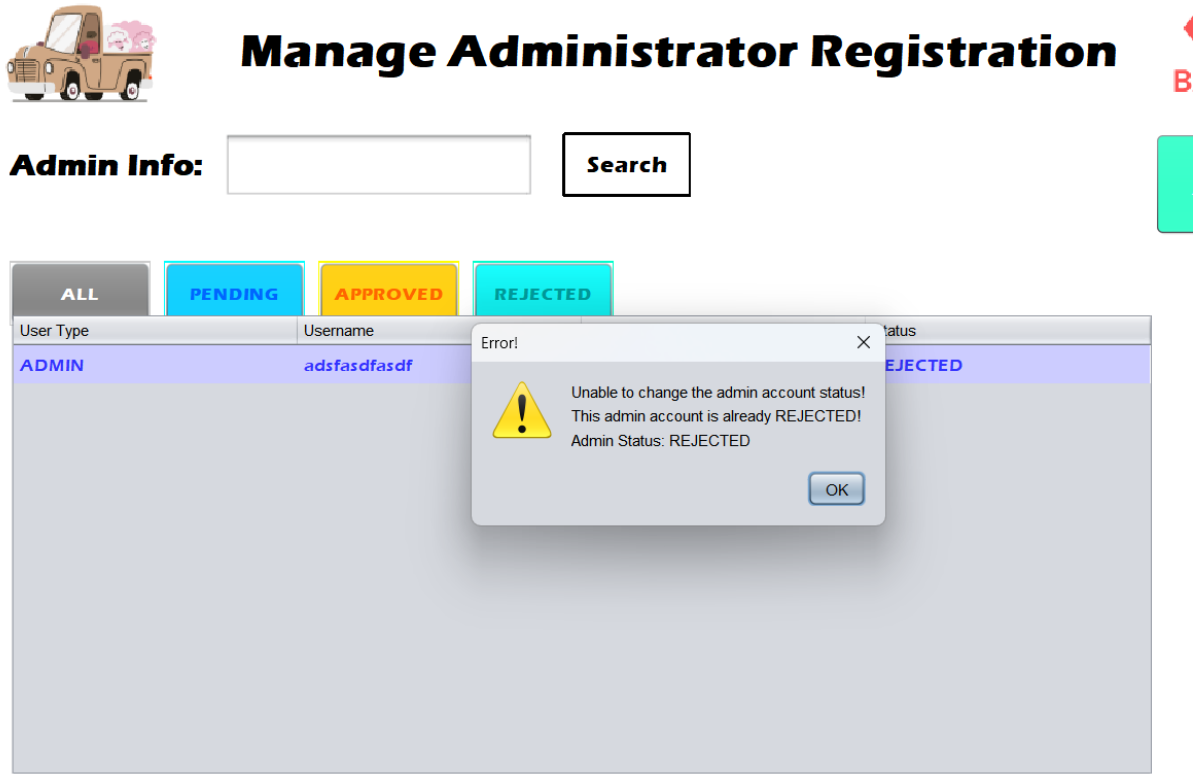


Figure 205 Manage Administrator Registration

This warning message will show is admin wanted to reject the admin account but the admi account are already rejected.

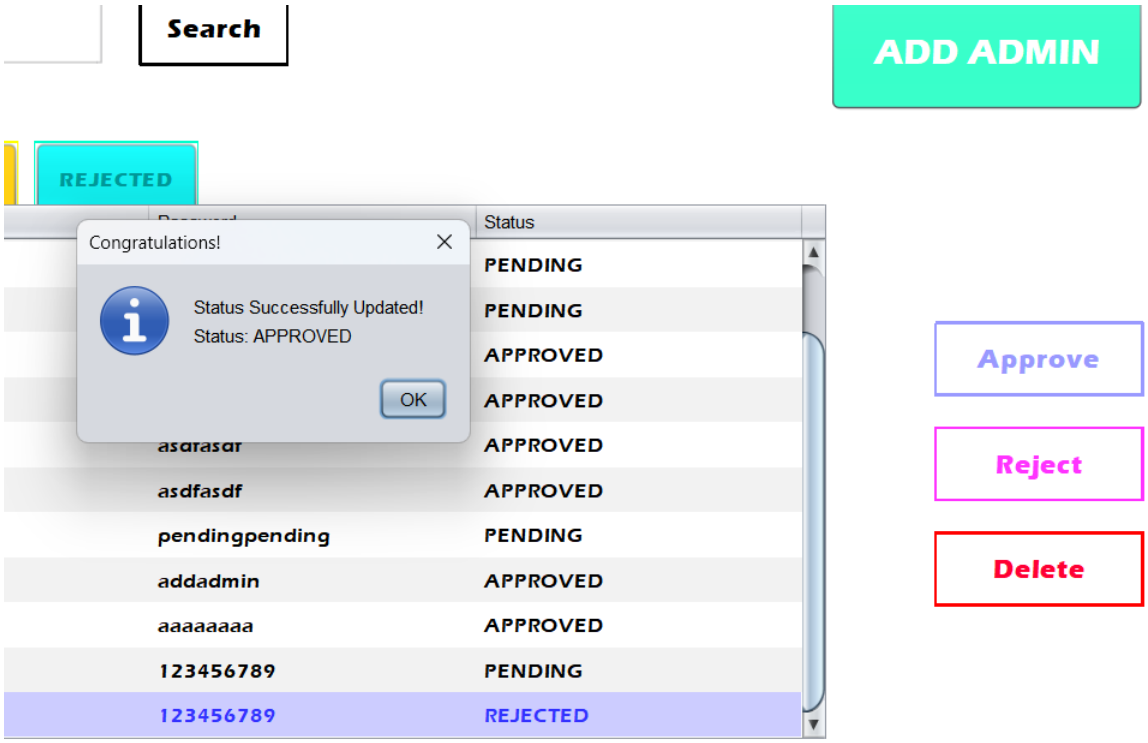


Figure 206 Manage Administrator Registration

This message will show if admin selected an account and click on “Approve” button.

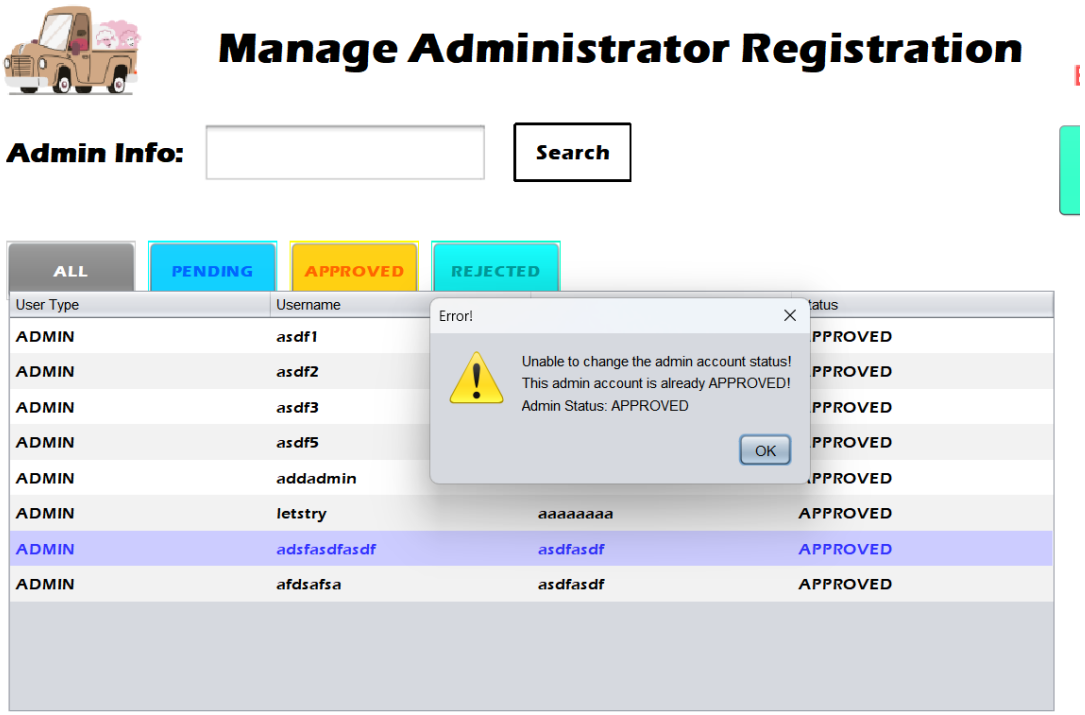


Figure 207 Manage Administrator Registration

This warning message will show is admin wanted to approve the admin account but the admi account are already approved.

Figure 208 Manage Administrator Registration

This message will show for admin to confirm their decision if admin click on “Delete” button. Press “Yes” to delete the admin account from text file otherwise click “No” to cancel the process.

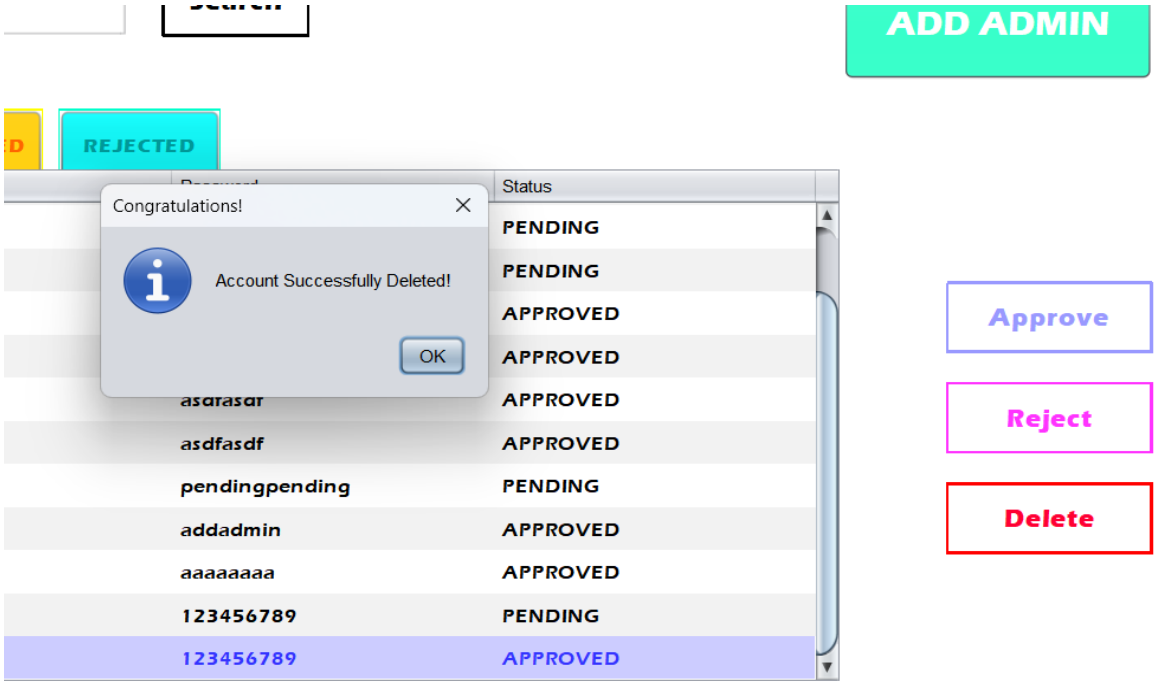


Figure 209 Manage Administrator Registration

This message will show if admin delete the account successfully.

5.5.3 Login Record

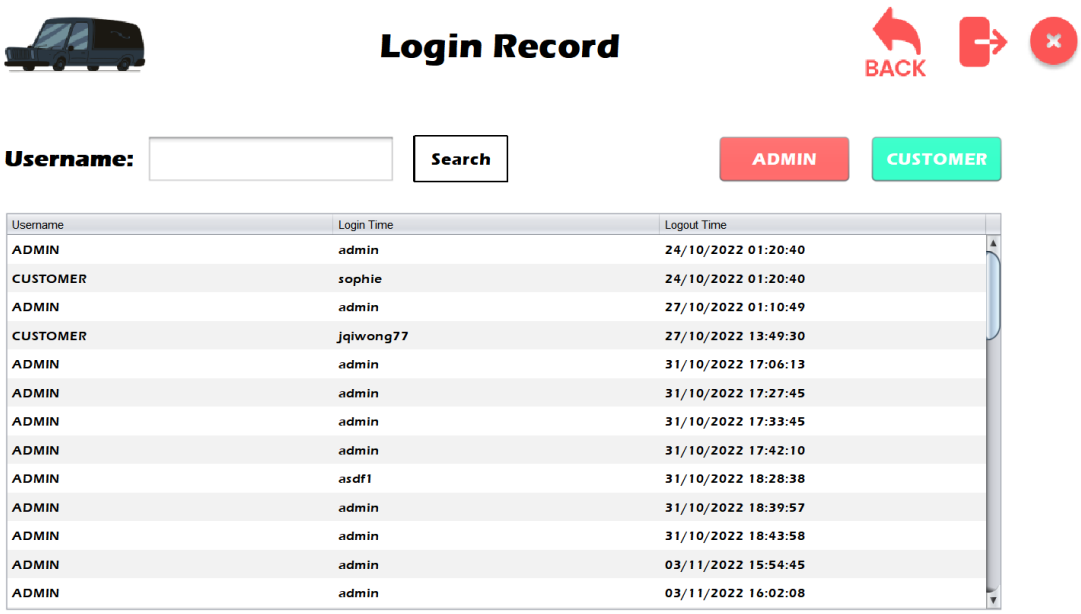


Figure 210 Login Record

This is the "Login Record" page where admin able to check the login and logout time for admin and customers.

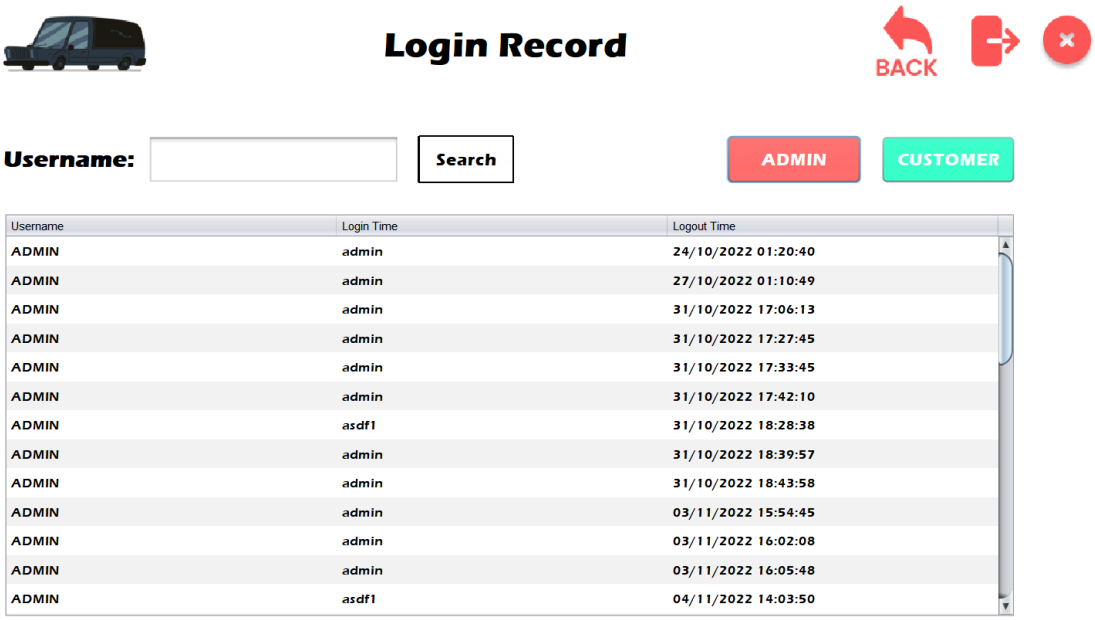


Figure 211 Login Record

Admin can click on the “ADMIN” button to sort the account type and system will show all accounts are admin.

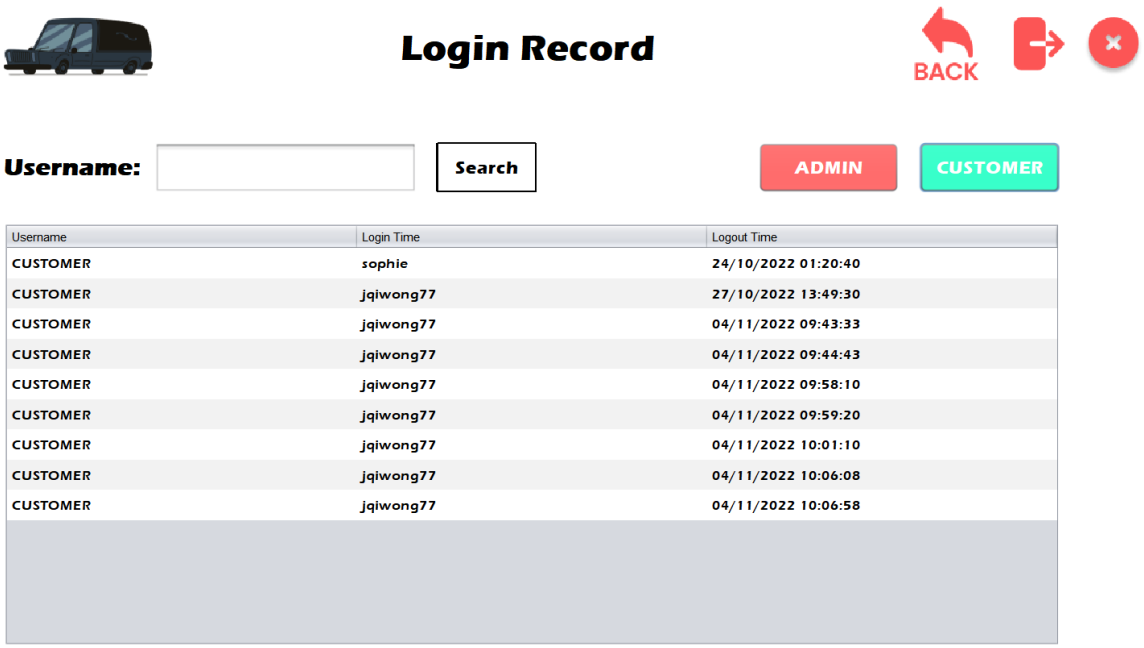


Figure 212 Login Record

Admin can click on the "CUSTOMER" button to sort the account type and system will show all accounts are customers.

5.6 Back, Logout and Exit buttons

Figure 213 Back, Logout and Exit buttons

The "Back" button are for user to go back to the previous page. Besides, the second buttons is "Logout" button where for user to logout their account and it will also store into loginRevord.txt file. The last button is "Exit" button where user can turn off the system.

```
ADMIN//admin//24/10/2022 01:20:40//24/10/2022 01:20:43
CUSTOMER//sophie//24/10/2022 01:20:40//24/10/2022 01:20:43
ADMIN//admin//27/10/2022 01:10:49//27/10/2022 01:20:43
CUSTOMER//jqiwong77//27/10/2022 13:49:30//27/10/2022 14:20:43
ADMIN//admin//31/10/2022 17:06:13//31/10/2022 17:25:45
ADMIN//admin//31/10/2022 17:27:45//31/10/2022 17:32:45
ADMIN//admin//31/10/2022 17:33:45//31/10/2022 17:43:45
ADMIN//admin//07/11/2022 16:31:27//
```

Figure 214 Back, Logout and Exit buttons

This is the page to store login date time and logout date time for admin and customer.

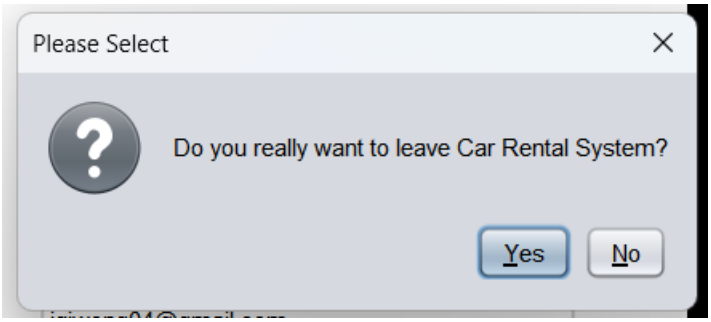


Figure 215 Back, Logout and Exit buttons

This message will show if users are going to logout the system. Press “Yes” to logout the system otherwise press “No” to continue using the system. After logout will go back to “Home” page

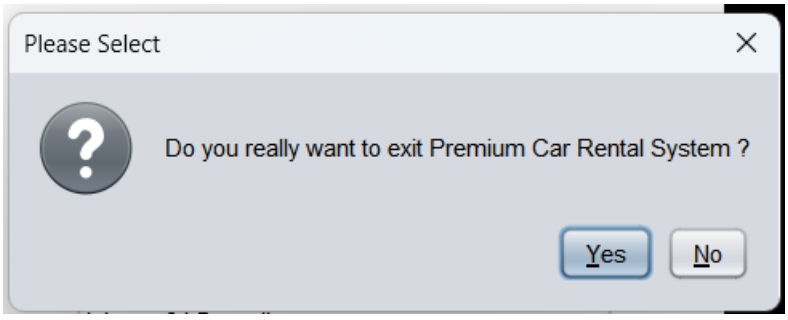


Figure 216 Back, Logout and Exit buttons

This message will show if users are going to leave the system. Press “Yes” to turn off the system otherwise press “No” to continue using the system.

6.0 Assumptions

i. The car rental system's equipment and materials will be acquired and made available for use.
ii. Everyone who was involved at the beginning of the task will stay involved until the project is completed. However, it is vital to note that personnel may quit the project at any moment throughout its life cycle.
iii. Members of the developer team would have fully access to the resources they need to fulfill their tasks on time. For example, latest software and equipment, as well as electric power throughout working hours.
iv. All equipment will be in good enough shape to be utilized during the project's lifecycle but as the project progresses, there may be problems about the resources malfunctioning or being harmed.
v. Assume that the username of the customer and admin must not be the same.
vi. The IC number of the customers must be unique.
vii. The customer must make payment online when they register for car rental.
viii. The customer must make full payment when they collect the car for car rental.
ix. The admin accounts can only be managed by the default admin.
x. The login record of the system can only be viewed by the default admin.
xi. The customer can only cancel the bookings 2 days before the pickup date.
xii. The customer must return car within 24 hours after the return date.
xiii. Customers who return after 24 hours of the return date will need to pay for fines.
xiv. The fine calculated will be 3 times of the normal price.

7.0 Conclusion

JAVA is a programming language designed to run Java applications on any machine that supports Java without the need for recompilation. It is, as everyone knows, one of the most popular and in-demand programming languages to learn, and it was among the first to define high-level threading tools. A Java project is required for ambitious developers. This effort aids developers in developing a real-world projects in order to hone their skills and adapt academic knowledge into practical experience. Java offers major benefits as both a commercial and a teaching language. The Java project has rigorous compile-time error checking, which is generally associated with Pascal. This allows educators to expose students to GUI programming fundamental ideas utilized in contemporary software. Overall, the java project provides a full design for the expanded language.

8.0 References

JavaTpoint. (2021). *Exception handling in java: Java exceptions - javatpoint*. www.javatpoint.com. Retrieved November 7, 2022, from https://www.javatpoint.com/exception-handling-in-java#:~:text=What%20is%20Exception%20in%20Java,which%20is%20thrown%20at%20runtime.

GeeksforGeeks. (2018, August 31). *Oops: Generalization as extension and restriction using Java*. GeeksforGeeks. Retrieved November 6, 2022, from https://www.geeksforgeeks.org/oops-generalization-as-extension-and-restriction-using-java/

GeeksforGeeks. (2022, July 15). *Packages in Java*. GeeksforGeeks. Retrieved November 6, 2022, from https://www.geeksforgeeks.org/packages-in-java/

GeeksforGeeks. (2022, July 5). *Getter and setter in Java*. GeeksforGeeks. Retrieved November 7, 2022, from https://www.geeksforgeeks.org/getter-and-setter-in-java/

GeeksforGeeks. (2022, June 27). *Generalization and specialization in Java*. GeeksforGeeks. Retrieved November 6, 2022, from https://www.geeksforgeeks.org/generalization-and-specialization-in-java/

Great Learning Team. (2022, October 31). *Oops concepts in java with examples: 2023*. Great Learning Blog: Free Resources what Matters to shape your Career! Retrieved November 6, 2022, from https://www.mygreatlearning.com/blog/oops-concepts-in-java/#:~:text=the%20data%20structure.-,What%20is%20OOPs%20in%20java%3F,real%2Dworld%20entities%20in%20programs.

Java - Encapsulation. Tutorials Point. (2022). Retrieved November 6, 2022, from https://www.tutorialspoint.com/java/java_encapsulation.htm

Java tutorial. (n.d.). Retrieved December 3, 2022, from https://www.w3schools.com/java

javaTpoint. (2021). *Multi-catch in Java*. Tutorials Point. Retrieved November 7, 2022, from https://www.tutorialspoint.com/multi-catch-in-java

Kai Tödter. (n.d.). *JCALENDAR*. toedter.com. Retrieved November 8, 2022, from https://toedter.com/jcalendar/

Loops in java: Java for loop - javatpoint. www.javatpoint.com. (n.d.). Retrieved December 3, 2022, from https://www.javatpoint.com/java-for-loop

Oracle. (2020, June 24). *Class Frame*. Frame (java platform SE 7). Retrieved November 6, 2022, from https://docs.oracle.com/javase/7/docs/api/java/awt/Frame.html

Oracle. (2022, October 11). *Java™ Platform, standard edition 8API specification*. Overview (Java Platform SE 8). Retrieved November 6, 2022, from https://docs.oracle.com/javase/8/docs/api/overview-summary.html

Oracle. (2022, October 11). *Package java.io*. java.io (Java Platform SE 8). Retrieved November 6, 2022, from https://docs.oracle.com/javase/8/docs/api/java/io/package-summary.html

Oracle. (2022, October 11). *Package java.text*. java.text (Java Platform SE 8). Retrieved November 6, 2022, from https://docs.oracle.com/javase/8/docs/api/java/text/package-summary.html

Oracle. (2022, October 11). *Package java.time*. java.time (Java Platform SE 8). Retrieved November 6, 2022, from https://docs.oracle.com/javase/8/docs/api/java/time/package-summary.html

Oracle. (2022, October 11). *Package java.time.format*. java.time.format (Java Platform SE 8). Retrieved November 6, 2022, from https://docs.oracle.com/javase/8/docs/api/java/time/format/package-summary.html

Oracle. (2022, October 11). *Package java.util*. java.util (Java Platform SE 8). Retrieved November 6, 2022, from https://docs.oracle.com/javase/8/docs/api/java/util/package-summary.html

Oracle. (2022, October 11). *Package java.util.logging*. java.util.logging (Java Platform SE 8). Retrieved November 6, 2022, from https://docs.oracle.com/javase/8/docs/api/java/util/logging/package-summary.html

Oracle. (2022, October 11). *Package javax.swing*. javax.swing (Java Platform SE 8). Retrieved November 6, 2022, from https://docs.oracle.com/javase/8/docs/api/javax/swing/package-summary.html

Oracle. (2022, October 11). *Package javax.swing.table*. javax.swing.table (Java Platform SE 8). Retrieved November 6, 2022, from https://docs.oracle.com/javase/8/docs/api/javax/swing/table/package-summary.html

Programiz. (n.d.). *Java constructors*. Programiz. Retrieved November 6, 2022, from https://www.programiz.com/java-programming/constructors

Team, E. (2022, November 25). *Project assumptions: 30 examples and how to manage them*. ProjectPractical.com. Retrieved December 2, 2022, from https://www.projectpractical.com/project-assumptions-examples-and-how-to-manage-them/

Tutorials Point. (n.d.). *Java - files and I/O*. Tutorials Point. Retrieved November 8, 2022, from https://www.tutorialspoint.com/java/java_files_io.htm