ECS506U Software Engineering

Group Project 2017

**Team 9**

**Assumptions**

**Authentication**

* The system will allow the administrators to add new administrators
* The day-to-day users of the system will be the ‘Mechanic’
* A logged in user (administrator) cannot delete themselves (system would be inaccessible if all admins could be deleted.

**Customer Account**

* There are no differences in the system requirements between an individual and a business customer. The module was not designed to have two subclasses of the ‘Customer’ class (‘Individual’ and ‘Business’), but it is using instead a variable in the ‘Customer’ class, identifying the type of customer. This improves the efficiency of the whole system by reducing the required memory allocation while it preserves maintainability as it is possible to create the two subclasses if they are required in the future.
* The auto-increment artificial primary key used in the customer database does not have a natural uniqueness. Neither the first name and last name of a customer. Therefore, it was assumed that a customer with the same combination of phone and email cannot be inserted in the system twice. This was implemented by making the columns of ‘PHONE’ and ‘EMAIL’ unique to constraint them.
* When deleting a customer from the system all the records referencing the customer are also deleted. However, this is with one exception that the customer does not have any bills unsettled at the time the user attempted to delete. The system will refuse to delete the customer if after searching all the bookings it finds any with ‘unsettled’ status.

**Vehicles**

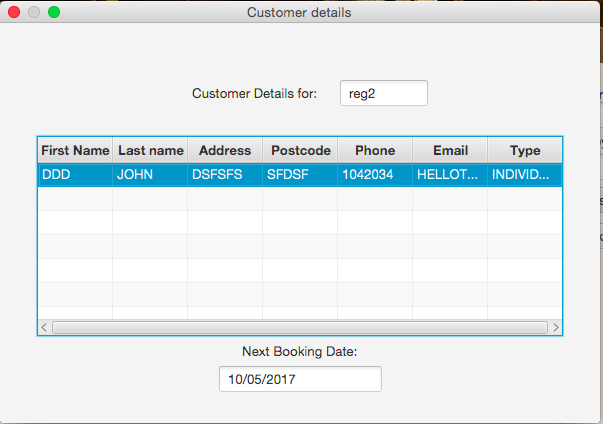
There were a few ambiguities I came across when designing the system which resulted in me designing it in the way I have. The ambiguities came from some of the requirements of my module.

6. Associated with each vehicle will be lists of parts used, past and future booking dates, and the total cost per booking (warranty and non-warranty). This must tally against the customer account.

It made no sense to store the parts used in my module as there was the parts module dedicated to this and as we were designing a fully integrated system there was no point in having information being repeated across several modules. Likewise, the past and future booking dates seemed irrelevant to store in my module as there was the bookings module dedicated to this purpose. My module enabled the bookings module to check if a vehicle was under warranty when helping to prepare the costs of each booking.

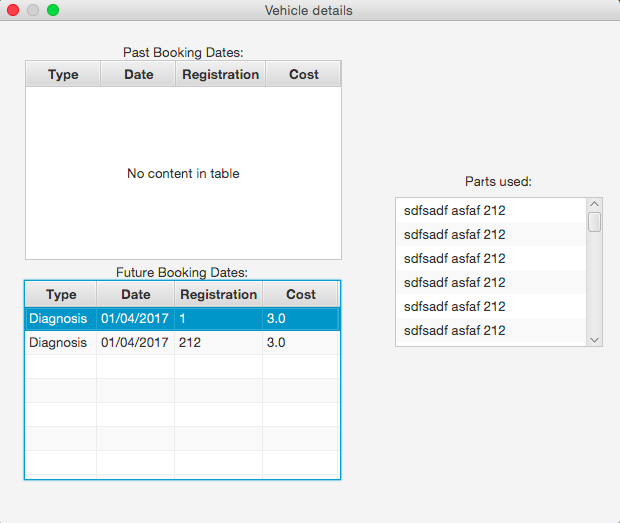
1. The user should be able to query the system to get a list of vehicles with customer details and next booking date. They should be able to add, edit and delete directly from this list.

Although the Customers module stores all relevant customer information I thought I would include this section anyways in my module to fulfill the requirement. It allows the user to select a vehicle to get the owner’s details and when the next booking date is. The user can obtain this information by clicking the booking details button which opens a pop up window with the relevant customer and booking information.



1. For each vehicle, the system user should be able to access lists of past and future booking dates, vehicles and parts used per vehicle etc.

Again, this requirement seemed ambiguous as the bookings module held the past and future booking dates and the parts module held the information regarding parts used per vehicle so seemed irrelevant to my module however I implemented it anyways to fulfill the requirement, Athanasios also had implemented this function so I used his CustomerBillsController as a template so is not fully my own work.



**Diagnosis and Repair**

* A booking can either be for diagnosis or repair.
* Mileage is not taken when making a diagnosis booking, but instead are taken during the diagnosis and added to a future repair for that vehicle.
* Parts are not used during a diagnosis booking.
* Mechanics have an hourly rate, and so bookings have at least an hour-long duration.
* Bookings involve only one mechanic.
* Bookings are completed in a day, or to be clearer a single booking is not spread over multiple dates.

**Part**

4. The system must track the stock levels of parts currently available in the garage. Assume a limited list of ten distinct parts that can be installed on any vehicle (of whatever model and type).

For this requirement I assumed that a single vehicle can only have up to 10 parts installed on it in total. If 10 are already installed and the user tries to install another then a pop up message appears informing the user that no more parts can be installed on this vehicle and the part is not added.

5. The system must be able to track part withdrawals for repairs and part additions for new stock items.

For this requirement I assumed that to track a part addition for new stock items there should be functionality to view deliveries. A delivery is created when stock items are added to an existing part or a new part is created (with a stock level greater than 0) and contain the part delivered, the amount delivered and the date delivered.

8. The system user should be able to query the system to get a list of parts used to repair a vehicle along with the vehicle and customer details. They should be able to add, edit and delete parts directly from this list.

For this requirement I assumed that the list of parts used should be displayed by booking and by vehicle, you can search all used parts by vehicle registration or repair id. You can also select a repair from the repair table view and get directed to the used parts table to view all used parts associated with the selected repair.

9. For each vehicle, the system user should be able to access lists of past and future booking dates, customer name and type of booking.

This requirement is already met in other parts of the system, it is possible to do this from the vehicles module. I found that there was no easy way to fulfil this requirement from my module, if I added this functionality it would make the GUI overcomplicated and I would be repeating functionality from others modules. Therefore, I did not include this under the parts tab.

10. When deleting a booking a confirmation shall be required.

This requirement seemed to have no place in the parts module. Users are unable to delete bookings from Parts and there is no requirement for them to do so. I also spoke to a TA about this and they agreed that the requirement had no place in the Parts module.

12. The user should be able to search for a part used by partial or full vehicle registration number or customer surname and firstname.

For this requirement I used a “LIKE” query so that it returns used parts where the associated vehicle has a registration that contains the string searched for.

**Specialist Repair**

* There are 10 repair centres CURRENTLY in use by the system, however this is not the limit.
* Specialist repairs can be done in 1 day (sent and returned date can be the same).
* The id for each part is the same for that particular part from stock.
* When sending a part in for repair, the vehicle the part is for must be recorded.
* The cost of parts is the cost of the new part for stock. The cost of the vehicle is given by the user.