mechanics need databases too

Prepared by Group 14: Heather Fillerup- Software Developer, Chris Nelson- Software Developer

CS 340-400: Spr 2020

Project- Step 2 Final

April 18, 2020

Summary

## Overview

Mahinui auto shop has seen record business in the last decade, repairing 50 or more cars on any given day. The owner, Brad, has finally decided to upgrade his repair order workflow from pen and paper being passed between his 10 mechanics to a website database. With more customers coming in by the day, keeping track of records has become a nightmare. Brad is looking to create a system for his mechanics to track and record the tasks involved with a car's repair, from diagnosis to customer pick up, and be able to view a display this progress on the homepage. The website will allow users to:

1. Search for car
   1. If car is not found
      1. Search for customer to add to car
         1. If customer is not found
            1. Add customer
      2. Add car
2. Add repair order to a car
3. Add work orders to repair orders
   1. First work order automatically added is the diagnosis task
      1. Followed by customer approval, order parts, repair, test drive and finally contact customer
   2. Add Mechanic to work order
4. Complete current work order and move on to next work order if needed
   1. Order parts task is only added if the repair order needs parts
   2. Provides an option to delete repair order's current work order or delete the entire repair order
5. View on the website homepage the following display of all of the cars currently being repaired at the shop and the current task being performed

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **EXAMPLE DISPLAY** | | | | | | |
| **Repair Order** | **Customer Name** | **Car Description** | **Current Task** | **Task Start Date** | **Mechanic** | **Details** |
| 1 | Jason Bateman | 2015 Honda Accord | Diagnosis | 03/1/2020 | Johnny |  |
| 2 | Charlize Theron | 2014 Toyota Civic | Customer Approval | 3/2/2020 | Ben |  |
| 3 | Ryan Reynolds | 2011 Honda Ridgeline | Order Parts | 3/3/2020 | Cameron | Delivery expected in two weeks |
| 4 | Scarlet Johansen | 2009 Toyota Front Runner | Repair | 3/4/2020 | Peter | Waiting for open bay |
| 5 | Jeff Bridges | 2014 Fiat 500 | Test Drive | 3/5/2020 | Frank |  |
| 6 | Tyler Perry | 2017 Kia Soul | Contact Customer | 3/6/2020 | Brian | Left message, waiting for call back |
| 7 | Brandon Fraser | 2019 Chevrolet Corvette | Diagnosis | 3/6/2020 | Ben |  |

## Programming Implementation and Assignments

For this project we will be implementing pair programming when possible. The goal is for the code in this project to be done together and we will switch back and forth between who is actively programming and who is giving feedback and checking for errors. This will allow us to learn from each other and help ensure that the code is of good quality.

## Database Outline

**customers**: records details about the customers who own the cars being repaired (Heather)

* id: int, auto\_increment, unique, not NULL, PK
* f\_name: varchar, not NULL
* l\_name: varchar, not NULL
* contact\_no: varchar, not NULL
* email\_address: varchar
* street\_address: varchar, not NULL
* city: varchar, not NULL
* state: varchar, not NULL
* zip\_code: int, not NULL
* relationship: a 1:M relationship between customers and cars is implemented with customer\_id as a FK inside of cars, where a customer can have 0 to many cars, and a car can only have one customer.

**cars:** records details about the car being repaired (Chris)

* id: int, auto\_increment, unique, not NULL, PK
* customer\_id: int, not NULL FK
* license\_plate: varchar, not NULL
* make: varchar, not NULL
* model: varchar, not NULL
* year: int, not NULL
* description: varchar
* relationship: a 1:M relationship between cars and repair\_orders is implemented with car\_id as a FK inside of repair\_orders, were a car can have 0 or more repair orders and a repair order can have only one car ; a 1:M relationship between customers and cars is implemented with customer\_id as a FK inside of cars, where a car requires one and only one customer and a customer can have 0 or more cars

**repair\_orders:** records details about the repair order being done on a car (Heather and Chris)

* id: int, auto\_ increment, unique, not NULL, PK
* car\_id: int, NOT NULL, FK
* date\_received: date
* date\_completed: date
* parts\_needed: Boolean, not NULL (defaults to false)
* relationship: a M:M relationship between repair\_orders and work\_tasks and a M:M relationship between repair\_orders and mechanics are both implemented with a composite table work\_orders; a 1:M relationship between cars and repair\_orders is implemented with car\_id as a FK inside of repair\_orders, where a repair order can have only 1 car, but a car can have 0 or more repairs

**work\_tasks:** records the types of tasks that can be added to repair orders, these tasks are associated to repair orders through work orders (Heather and Chris)

* id: int, auto\_ increment, unique, not NULL, PK
* category: varchar, not NULL (diagnosis, parts ordered, parts received, repair, test driven, customer paid, customer picked up)
* relationship: a M:M relationship between repair\_orders and work\_tasks and a M:M relationship between mechanics and work\_tasks are both implemented with a composite table work\_orders

**work\_orders**: composite table that records the tasks that have been added to the repair\_orders and also tracks the mechanic responsible for the work order

* repair\_order\_id, not NULL PK
* order\_task\_id, not NULL PK
* mechanic\_id: int, FK
* start\_date: date
* end\_date: date

**mechanics:** records details of the mechanic responsible for the work orders (Chris)

* id: int, auto\_ increment, unique, not NULL, PK
* f\_name: varchar, not NULL
* l\_name: varchar, not NULL
* relationship: a 1:M relationship between mechanics and work order is implemented with mechanic\_id as a FK inside of work\_orders, where a mechanic can have 0 or more work\_orders but a work order can only have one mechanic; a M:M relationship between repair\_orders and mechanics and a M:M relationship between mechanics and work\_tasks are both implemented with a composite table work\_orders;

## Schema

**customers**(

id,

f\_name,

l\_name,

contact\_no,

email\_address,

street\_address,

city,

state,

zip\_code)

**cars**(

id,

customer\_id,

license\_plate,

make,

model,

year,

description)

**repair\_orders**(

id,

car\_id,

date\_received,

date\_completed)

**work\_orders** (

repair\_order\_id,

order\_task\_id,

mechanic\_id

start\_date,

end\_date)

**work\_tasks**(

id,

category)

**mechanics**(

id,

f\_name,

l\_name,

rate)

## Changes from Original Design

* Removed Parts table since we only have to implement one M:M relationship
* Removed cost, hours and rate attributes to focus on the order tasks and mechanics for the repair order
* Removed redundant fields from statuses such as current\_status and complete, we can check if the current status is complete by looking at which statuses have a NULL end\_date
* Changed the name of the status table to work\_tasks for better clarification
* Changed the name of the repairs table to repair\_orders for better clarification
* Changed repairs\_stasuses relationship table to work\_orders and added it as a composite table for better clarification
* Made work\_orders a composite table with attributes moved from work\_tasks (mechanic\_id, start\_date and end\_date). This was to satisfy the requirement that when we delete our M:M task and repair orders relationship record, we cannot delete any record from the repair\_orders or work\_tasks tables.
* Updated the customers participation with cars, a customer can have 0 or more cars, this will allow a customer to be added to the database without requiring a car\_id
* Updated the cars participation with repair orders, a car can have 0 or more repairs, this will allow a car to be added to the database without requiring a repair
* Changed mechanics relationship with work\_tasks to work\_orders, this allows a record in work\_orders to be deleted without having to delete a record in work\_tasks. The initial relationship with work\_tasks was also incorrect, mechanics has a M:M relationship with both repair\_orders and work\_tasks.
* Added parts\_neeeded attribute to repair\_order, this will allow us to use logic to only add a Order Parts work\_order to a repair\_order when true
* Added pair programming to programming assignments because we want to work on everything together if possible.

## Review Feedback

* Reviewer 1
* Reviewer 2