mechanics need databases too

Prepared by: Heather Fillerup, Chris Nelson

April 6, 2020

Summary

## Overview

Mahinui auto shop has seen record business in the last decade, repairing 50 cars on any given day. The owner, Brad, has finally decided to upgrade his repair order workflow from pen and paper being passed between mechanics to web and database. With more customers coming in by the day, keeping track of records has become a nightmare. Brad is looking to create a system to track and record the progress of the car repair. The website will allow the mechanics to:

1. Search if a car already exists in the database, add one if not, and associate a car and a customer with a new repair order
2. Update status of the repair order until the car repair is complete or the customer decides to cancel
3. View a display containing all of the cars currently be repaired at the shop and the progress of each repair.

## Database outline

**customers**: records details about customers who own the car

* 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
* city: varchar
* state: varchar
* zip-code: int
* relationship: a 1:M relationship between customers and cars is implemented with customer\_id as a FK inside of cars

**cars:** records details about the car being repaired

* id: int, auto\_increment, unique, not NULL, PK
* customer\_id: int, not NULL FK (tracks primary owner)
* vin\_no: varchar, not NULL (license plate is easier to find on car, or last 8 of vin, vendor database use license plates APIs)
* make: varchar, not NULL
* model: varchar, not NULL
* year: int, not NULL
* description: varchar
* relationship: a 1:M relationship between cars and repairs is implemented with car\_id as a FK inside of repairs; a 1:M relationship between customers and cars is implemented with customer\_id as a FK inside of cars

**repairs:** records details about the repair order being done on a car

* id: int, auto\_ increment, unique, not NULL, PK
* car\_id: int, not NULL, FK
* current\_status (FK to status\_id? default to diagnosis status\_id and auto increment when completed)
* cost: decimal (calculate based on status total hours \* shop rate)
* date\_received: date
* relationship: a M:M relationship between repairs and statuses is implemented with a relationship table repairs\_statuses; a M:M relationship between parts and repairs is implemented with a repairs\_parts relationship; a 1:M relationship between cars and repairs is implemented with car\_id as a FK inside of repairs

**statuses:** records detail about the various statuses of the repair order

* id: int, auto\_ increment, unique, not NULL, PK
* category: varchar, not NULL (diagnosis, parts ordered, parts received, repair started, repair finished, test driven, customer paid, customer picked up)
* mechanic\_id: int, not NULL, FK
* start\_date: date
* end\_date: date
* complete: Boolean (easier to check for NULL end date than use this?)
* hours: int
* relationship: a M:M relationship between repairs and statuses is implemented with a relationship table repairs\_statuses; a 1:M relationship between mechanics and cars implemented with mechanic\_id as a FK inside of statuses

**parts:** records details about the parts ordered/used for a repair

* id: int, auto\_ increment, unique, not NULL, PK
* upc\_no: int (unless there can be leading zeroes or strings, then we must store as varchar)
* category: varchar (accessories, brakes, battery, air conditioning & heating, alternators & starters, motor oil, shocks & struts, lighting, bearings & seals, belts & hoses, chassis & steering, CV driveshaft & axle, engine cooling, engine, filters, gaskets, ignition, tire & wheel)
* description: varchar (needed? upc look up would give more details)
* base\_cost: decimal
* markup: decimal (percentage)
* relationship: a M:M relationship between parts and repairs is implemented with a repairs\_parts relationship

**mechanics:** records details of the mechanic working on the statuses

* id: int, auto\_ increment, unique, not NULL, PK
* f\_name: varchar, not NULL
* l\_name: varchar, not NULL
* rate: int (we could replace with a base shop rate if we aren't calculating shop profit)
* relationship: a 1:M relationship between mechanics and cars implemented with mechanic\_id as a FK inside of statuses; a M:M relationship between mechanics and certifications implemented with a mechanics\_certifications relationship

**certifications**: records details of the certifications a mechanic can have

* id: int, auto\_increment, unique, not NULL, PK
* category: varchar, not NULL(A1-A9)
* type: varchar, not NULL (engine repair, automatic trans/transaxle, manual drive train and axles, suspension and steering, brakes, electrical/electrical systems, heating & air conditioning, engine performance, light vehicle diesel engines)
* issued\_date: date
* expiration\_date: date (5 years from issued\_date)
* relationship: A M:M relationship between mechanics and certifications implemented with a mechanics\_certifications relationship