# CSC 540 - DATABASE MANAGEMENT CONCEPTS & SYSTEMS

## FALL 2022 PROJECT 1 - DATABASE APPLICATION DESIGN & IMPLEMENTATION

Draft Project Description - **Due Nov 2nd.**

AUTOR - Auto Repair and Service Management System

~~**Version: 9/12**~~
**Version2: 9/23:**
Notes about changes:
1. Some sentences have been revised or added in the description to improve clarity.
2. The structure of the document contains general information about entity types and attributes and relationships in the main description part of the document. The application flow elaborates on this, capturing some information about constraints and program behavior.

# Introduction

The goal of the project is to design a relational database application for supporting a chain of car repair and service centers by a company. The project should be carried **out in teams of four (4)** unless prior permission is obtained from the instructor**,** and team peer assessments will be part of the overall grade for each student. The project is expected to be executed in stages with interim deliverables submitted, as described in this document.

The description has several details that you should pay attention to. First, it describes the data and application requirements. It then gives information about deliverables and tentative deadlines (deadlines may change slightly) and “getting started” guidelines.

**Notes:**
1. **You should necessarily assume that this is an imperfect description and will be subject to updates**. Therefore, it is important that you read through the description in the coming days and ask questions to clarify any missing or ambiguous statements.
2. In addition to revisions to help clarify description, **supplementary information about application flow and a set of queries that must be implemented, final submission instructions** will be added to the description in the coming weeks.
3. **You are responsible for testing your application’s correctness using test data that you craft**. However, for the demonstration, all teams will use the same data to help keep uniformity in assessing team projects. We will provide this test data about a week prior to the demos.

# Project Specification

## Background and Overview

Downtown Auto Car Service and Repair Centers is a chain of car service centers owned by Downtown Auto Inc spread across the United States. They service and repair Honda, Nissan, Toyota, Lexus and Infiniti cars.

Your task is to design a database system, AUTOR, that allows Downtown Auto corporation and its customers to keep a record of and manage the following information for all its repair centers. This description provides an indication of what important information (italicized) will need to be stored in the database as attributes and relationships. Suggested attributes for the different entity types are listed in the application; however, these suggestions are by no means exhaustive so you may find that you need to introduce additional attributes beyond the ones explicitly mentioned in this document in some cases.

### Service Center details

Each service center is identified by *a globally unique ID, address, and a telephone number*. There might be multiple service centers in a given state in the country but each center is independently managed. Each center operates 5 days a week (M-F) from 8 AM to 8 PM. Some (not all) are also open on Saturdays from 9am - 1pm.

The general employee structure in each center has a *manager* who manages all employees, a *receptionist,* and several *mechanics*. For each employee, we store their locally unique *9-digit employee ID, name, address, email address, phone number, service center ID, and the role at the service center they work at*. Each employee is associated with only one service center. Each employee can only play one role at a time (for example, a mechanic cannot also be a receptionist). While the manager and receptionist are contract employees with fixed annual salary, mechanics are hourly paid workers. Each center has its own hourly rate for mechanics.

Each center offers a list of car services e.g. Oil Changes, Check Engine Light Diagnostics & Repair, Engine Repair, Catalytic Converter Repair, Alternator Repair & Replacement, Compressor Repair & Replacement. These services are loosely categorized into two broad categories: a *repair service* or a *maintenance service*. Some services like *Brake Repair* are both maintenance and repair services while others like *Oil Change* are exclusively maintenance services. Repair services are categorized into 6 main subcategories: (i.) *Engine Services* which include individual services like *Belt Replacement*, *Engine Repair*; (ii.) Exhaust Services which include individual services like *Catalytic Converter Repair*, *Muffler Repair*; (iii.) Electrical Services which include individual services like *Alternator Repair*, *Power Lock Repair*; (iv.) Transmission Services which include individual services like *Axle Repair*, *Transmission Flush* (v.) Tire Services which include individual services like *Tire Balancing*, *Wheel Alignment* and (vi.) Heating and Air Conditioner Services which include individual services like *Compressor Repair*. Individual services belong to only one of the 6 repair service subcategories (although as said before they could also be classified as a maintenance service).

Individual services have a globally unique number and name. The price for each service may vary in different stores; however, the duration for each service is the same in every store. For example, at

one center, brake repair in a Honda may cost $50 while at another store it may cost $70, but the service takes the same 3hrs at both centers. Each service also has a price and a time estimated for the service job, which is based on the car and the specific auto center. For example, at one center, brake repair in a Honda may cost $50 and take 3hrs while the brake repair for the Lexus may cost $95 and take 4hrs at this same location. At another center, the price estimates may be different.

Maintenance services are usually provided in “bundles" called schedules. Schedules include a set of individual services in a bundle that are completed together and are priced as a bundle. For example, a maintenance schedule may include *Oil Changes*, *Brake Repair*, *Check Engine Light Diagnostics*, etc. There are three bundles or schedules, Schedules A, B and C. Each schedule has a specific subset of services that are covered and there is a downward inclusion relationship so that Schedule B contains all the services in A and some extra ones and likewise Schedule C contains everything in B and some additional ones. As with the individual services, the pricing estimate is dependent on the type of car e.g. Honda vs Toyota and the particular center. Maintenance services are usually done in a rotational manner i.e. after a *Schedule A* maintenance service, the next maintenance will be *Schedule B*, then *Schedule C*. After *Schedule C*, it restarts at *Schedule A*.

The example individual services mentioned in the above paragraphs are a subset of the services. A complete list of in services and the list of services that comprise each of the scheduled maintenance services, will be provided later as a part of the demo data (however, you may consider the list of service categories listed here as complete).

Each mechanic works no more than (50hours a week). A mechanic may ask for and may be given time off, as long as there are always at least 3 mechanics present at any given time.

### Customer details

A *Customer* has *id* (an integer) that is unique with respect to a specific center, a *first* and *last name*. A customer is associated with at least one vehicle which is identified by globally unique *vin number* (8 alphanumeric characters), *car manufacture*r e.g. Honda, and *current mileage* (integer), *year, the last scheduled maintenance service class performed denoted by a single character ‘A’, ‘B’, or ‘C’*. For each customer service event, the services rendered, by which *mechanics* and *total amount charge*d and *total amount paid* are recorded. Each customer has a *status* field stating if their account is in good standing or not (i.e. has an outstanding balance).
Every customer is associated with one specific service store. A new customer can be added only by the receptionist.

Customers become “inactive” if their profiles no longer include any vehicles (this can happen as a result of deleting the cars on their profiles - see Application Flow Section below). Inactive customers will still be allowed to update their profiles by adding vehicles which changes their status to active. Customer has a *status* attribute recorded as a boolean field (*0* - inactive, *1* - active).

### Car Service Scheduling

A *Customer* can come in for either a scheduled maintenance or a specific repair service or both (for example if additional car components beyond those being serviced require repair). They initiate a request for services by adding the vin number of the car being repaired, the ids of any repair services they want and/or the scheduled maintenance service that they want. For example, a

request for a car repair service may include a *Schedule A* maintenance and additional repairs like *Muffler Repair* and *Catalytic Converter Repair* (more details are added in the application flow section). If the car has never been serviced at the center and therefore has no record in the center's database, the customer must first update their profile to include the car before requesting services for it. ~~is not on record at the shop, information about it like the Manufacturer, Current mileage and year are entered first. For cars that have a record in the shop, if based on records the scheduled maintenance selected is not the correct one that is next on the schedule, a prompt is used for the customer to confirm change~~. For each service event/visit, a customer selects what services they desire (it can be a combination of repair services and maintenance services). If the customer indicates that they want a maintenance service done, the system automatically selects the correct service schedule ('A', 'B', 'C') based on the car's maintenance history which is on record. Once all selected services are checked/validated, a cost estimate and numbered list of available "matching" time slots is given (up to 1 month ahead). "Matching time slots" means time slots sufficient for the estimated duration of the job and availability of mechanics. ~~Once the customer selects one of the available times, the relevant number of mechanics are denoted as "booked" for that period. In general, 1 mechanic is needed for a Schedule A service, 2 for Schedule B and 3 for Schedule C~~.

With respect to scheduling, time is managed in terms of slots. Each day has 9 time slots (8am - 9am is slot 1, 9am - 10am is slot 2, and so on). Therefore a service that lasts 2 hours takes two time slots. The lunch period is the 12 - 1pm time slot and no work is done during that period. However, a service may span before and after lunch. In other words, a 2hr service that starts at 11am will end at 2pm (skipping the lunch hours). Each service event (all services scheduled by a customer in a single visit to the interface) is handled by a single mechanic. Scheduling will need to avoid double-booking or overbooking a mechanic i.e. > 50hrs, or booking them while they are on vacation. Also, scheduling should not book on a Saturday if the store is not open on that day.

(**A Suggestion**: To avoid having to keep track of the actual days of the week, we will use a numbering scheme to encode days. Each month will have week ids 1 through 4, and each day of the week will correspond to an integer id as well e.g. Sunday = 1, Monday = 2, …. Saturday = 7). Therefore, we can easily identify Saturdays by day id.)

**However, you are free to use actual dates and times to keep track of timeslots. The main thing is to be able to identify the lunch hours and also Saturday hours. If you choose to do this, you will need to adjust the parameters in some of the mechanic role procedures that use time slots. (see role discussions in the later part of the document.)**

For each service event, an *invoice* with unique *invoice id*, *customer id*, *vin of car serviced*, *date of service*, *services provided*, *cost for each service*, *mechanics* that provided the service and *total bill* (sum of costs) is stored. An invoice also has a *status* attribute recorded as a boolean field (*0* - unpaid, *1* - paid). By default when an invoice is created the *status* is unpaid (*0*).

# High Level Application Flow Requirements

The application should support the correct flow i.e. appropriate sequence of application steps and at each step the appropriate function options, for the appropriate role. Below we describe some of these functions. Some of these functions may require the use of advanced features like Triggers, Stored Procedures, Views, etc. The application can be driven by text-based menus which present numbers next to menu options and users can just enter the appropriate number for the menu option they desire. (GUIs are not required but are welcome if you wish to). Below is the description of an example flow. In the coming weeks, we will be providing you a supplementary document providing similar information to the one below to encourage a uniform application flow across projects.

You are also encouraged to include reasonable error handling code e.g. prompt for retry if user enters incorrect input.

| Display | Menu | Input | Output |
|---|---|---|---|
| Display the menu | 1. Login<br>2. Sign Up<br>3. Exit | Enter Choice (1-3) | Go to the appropriate page.<br><br>If exit is chosen, terminate the program. |

Login

| Display | Menu | Input | Output |
|---|---|---|---|
| Ask user to input the following details in the order shown below, followed by the menu.<br><br>A. User ID<br>B. Password | 1. Sign-In<br>2. Go Back | Enter Choice (1-2) | If the user chooses 1, validate credentials and recognize if user is a Manager, Receptionist, or Customer to go to the correct **Landing page**.<br>Print "Login Incorrect" for invalid credentials and ask to enter again.<br><br>If the user chooses 2, go back to the **Home page** |

The following gives an overview of the functionality required for the different roles.

For simplicity, in all the cases where accounts are created, the user's last name will be used as default password. We will also not bother with a password resetting feature.

# Admin/Any User

This refers to the role that initializes the system with global data items (i.e. data items that apply to all stores) and makes general updates e.g, adding a new store to the chain of stores). You do not need to explicitly create the admin account with any credentials (in other words, the admin functions can just be accessible directly without logging in first). However, there will be nothing wrong with creating one if you so choose.

- **System Set Up:** This is a procedure that initializes all the general information about the services that is universal to all stores. It loads information about all individual services offered by the chain, all service categories, service schedule bundles for Schedule (A, B, C), and the duration of services (not prices that are defined by the manager of every store). These data are applicable to all stores.

Then, it loads general information for all existing stores i.e. for each store, *store id* (either auto generated or given), *address* and *store manager* information (*firstname*, *lastname*, *username*, *password*, *salary*, and *employeeid*) and the store's *minimum* and *maximum hourly wage* for mechanics. (Additional details about each store are added at the store level by the manager after the store has been set up.)

All of this general chain information will be provided in a csv or spreadsheet format which can be read by the procedure and loaded into your database. The file format will be given.

- **Add New Store**: This menu should allow admin to enter information about a new store to include the *store id* (either auto generated or given), *address* and *store manager* information (*firstname*, *lastname*, *username*, *password*, *salary*, and *employeeid*) and the store's *minimum* and *maximum hourly wage* for mechanics. This automatically creates the managers account for that store.

- **Add New Service:** Allows admin to add information about a new service they are offering. Although service categories and maintenance schedule bundles will remain fixed i.e. will remain as initialized, individual services can be added. A new service will be assigned a service category when added.

# Manager

When logged in as manager, the manager will be able to add other store employees i.e. receptionist and mechanics along with their relevant information

- **Setup Store**: (these can be done in any order. However, only after the store has all necessary components including minimum number of mechanics hired (3), is it considered *available* i.e. operational and can accept customer requests to schedule services. While the process is still going on, store is still in *pending* mode

    - **Add employees**. Add information for employees including role, salary (or hourly wage). There should be only one receptionist per store. Once these employees are added, accounts are automatically created for them and they should be able to login. Salaries must fall within the min and max salaries set up for the stores.

- **Select Operational Hrs**: Indicate if the store is open on Saturdays or not.
- **Set Up Service Prices**: Assign prices for each of the individual services available.

- **Add New Employee:** A manager should be able to add additional employees later also after initial setup.

# Receptionist

The account for the receptionist can be created only by the manager. When logged in as a receptionist, the homepage displays the following options:

- **Add New Customer Profile:** Adds basic information for a customer for the current service center which must include exactly one vehicle. This generates a customer account that a customer can log into and update.

- **Find Customers with Pending Invoices:** Finds customers with pending invoices and displays some basic information about each customer and pending invoice.

# Customer

The account for customers shall be created from the application (only receptionists can create customer accounts). When logged in as a customer, the homepage displays the following options:

- **View and Update Profile:** The logged in customer should be able to update their profile information limited to adding and deleting cars to/from their profile.

  - **View Profile:** The logged in customer should be able to view their profile information.
  - **Add Car to Profile:** A customer should be able to register a car with a service center. They should be allowed to only register cars that belong to the list of manufacturers that Downtown Auto works on. If the number of cars increases to at least 1 the customer's status becomes active.
  - **Delete Car from Profile:** When a customer sells a car they can delete a car from their profile. If the number of cars drops to zero, the customer's status becomes inactive.

- **View and Schedule Service**: This will have the following sub-menus:

  - **View Service History**: A customer should be able to see service history for their car using the car's VIN number
  - **Schedule Service**: This subflow should allow a customer to enter the list of service codes (either one after the other or as a set). These indicate the services the customer is interested in. This can include both a maintenance service and some repair services. This procedure will randomly select mechanics from the list that will be available at that date and time. Scheduling is done in blocks of hour multiples i.e. 8am – 9am for a 1hr service or 8am – 10am for a 2hr service (except lunch hour).

- **Invoices**:

- **View Invoice Details**: This should display a numbered list of invoices and status e.g. *1.) Invoiceid1, paid*. When a customer clicks on the number, they can see the details of the invoice.

- **Pay Invoices**: A customer should be able to pay for the invoice that is uniquely identified by *invoiceid*. If the invoice is unpaid, the system updates the invoice status to "paid", otherwise if the invoice was already paid the system displays the message telling that it was already paid and doesn't update the invoice status.

# Mechanic

The account for mechanics can be created only by the manager. When logged in as a mechanic, the homepage displays the following options:

- **View Schedule**: This will show the list of time slots that mechanic is booked for the service
- **Request TimeOff:** This procedure allows a mechanic to request some time off. It takes as parameter the time slots they want to be off (indicated by week, day, time slot start and end slot ids). It returns with a status of 1, if approved and 0 if not approved. It is automatically approved if the mechanic is not already assigned a job during that period and if it will not cause the number of working mechanics to go below the threshold.
- **Request Swap:** one mechanic can request another to swap periods in their schedules. The swap duration doesn't have to be exactly the same as long the swap doesn't create any conflict for either of the mechanics. Basically, this is a procedure that takes in a *timeslot range* that the requesting mechanic is offering to swap, (identified by day id, weekid, begin slot id, end slot id), the id of the mechanic that is being requested to swap and that time slot range of the requested mechanic that is of interest. The source and destination time slot ranges do not have to be equal. For example, mechanic A who has been assigned a 1 hour service timeslot can request another mechanic B who has an assigned 2 hour service timeslot to swap. The time slot request will be rejected if it fails a number of basic validity checks. If the swap request passes validity checks, then it is considered to be in a *pending state* (you can use integer 0) and placed in on an approval list of the requested mechanic to approve/accept swap request. The validity checks are as follows:
  - The procedure must verify that indeed the time slot range is assigned to the mechanic being requested, otherwise reject. For example, it should check that timeslot is arranged to Mechanic A if Mechanic A was requested (not B, or C).
  - The swap request is also rejected if the reassignment of slots to the mechanic being requested causes him to either be overbooked (>50hrs) or double-booked (he already has an assignment for the time being offered in the swap).
- **Accept//Reject Swap:** Each mechanic sees a numbered list of their pending swap requests which include the name of the requesting mechanic and timeslot range being requested for swap. The mechanic can pick a swap request from the list and approve it or reject it. If approved, the request is considered to be in "*accept status*" (integer 1) and the schedules of both mechanics are updated to reflect the change. If rejected, the request is considered to be in "*reject status*" (integer 2) and their schedules remain unchanged.

## Sample Queries (TBA)

You will be given a list of queries to implement ahead of the demo. The queries will be numbered. You will simply need a screen with the same numbering so that selecting the numbering executes the query and shows the result.
The queries will be provided soon.

# Project Deliverables

## Milestone 0 - Questions on the Forum - on or before Sep 26th

Post any questions for clarification of the project description on the forum

## Milestone 1 - Report, Due October 3rd **9AM**

For the first milestone, you should:

1. Fill in the form for deciding team members as soon as possible.
2. A partial ER-Diagram covering the entity and relationship types and constraints that you have identified so far relevant to Employees, Stores and Services.
3. A translation of your partial E-R model into SQL.
4. Include a table that has two columns. One column has a phrase in the description document that you have identified as a constraint and the other column says how your model so far has captured the constraint e.g. foreign key. You do not need to include those that you have not captured yet since some constraints will require all of SQL to capture.
5. Also include up to 5 functional dependencies that you have identified so far.

## Milestone 2 - Final Report. Nov 2nd.

Each team will have to book a timeslot to demo the application to the Professor or Teaching Assistants. The details about the demo will be conveyed later. The following need to be submitted. Submission instructions for final report will be given later.

# Grading Rubric (TBA)

| Prelim Report<br><br>**20** | Final Report<br><br>**40** | DEMO<br><br>**40** | Peer Review Weight Factor Per Total Score Range |
|---|---|---|---|
| Partial ER **(8)**<br><br>Constraints:<br>- Table and<br>- SQL **(10)**<br>- FDs **(2)** | Discussion **(10)***<br><br>Extended E-R Model **(8)**<br><br>Extended FD list **(4)**<br><br>Final Constraint Model - SQL **(18)** | Application Flow<br><br>Role Validation<br><br>Demo SQL Queries | 70 - 84% ⇒ **.**9 of score<br><br>**>** 84% ⇒ 1 of score<br><br>**<** 70% ⇒ **.**7 of score |

*Footnotes
- Discussion - explanation of extensions to preliminary report submission (what new entities, relationships constraints captured). Justification of design (BCNF or 3NF).
-

# Getting Started

I would recommend using an IDE(like Eclipse or IntelliJ) for developing the project and some form of version control like GitHub for sharing project amongst team members. (You can create private repositories on NCSU Github). Using Oracle's SQL-Developer would also help you in writing long queries, triggers and procedures because of advanced features like debugging and static analysis of query.

The following link will help you to connect to the database using the JDBC Driver: Creating a connection using JDBC. Students are encouraged to read more about proper handling of connection and JDBC Best Practices.

Note that points will be deducted for **improper handling** of connection in the java application.