CSCI C400
Small Project#1 – Cars and Car Values Mini System

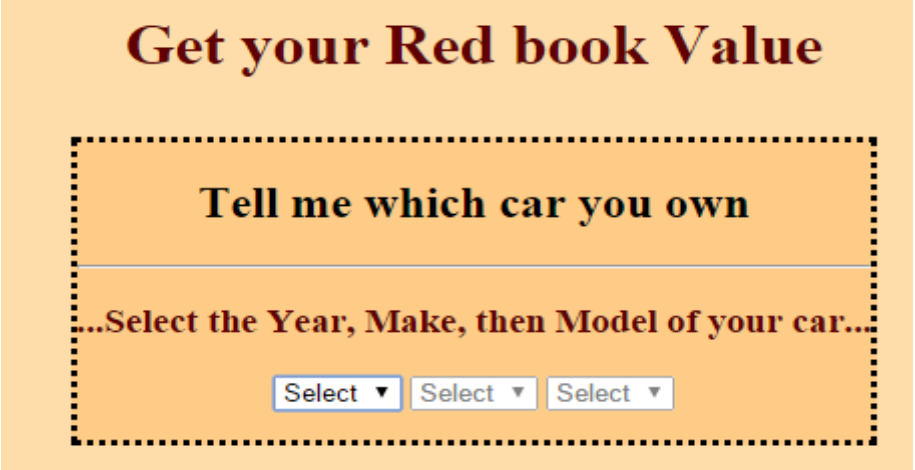
Group Assignment – You must work with your group members to complete this assignment

In this mini project you will be creating a mini vehicle valuation system using the PHP programming language. This is an open ended project. I'm relying on your creativity and imaginations to create the best possible web solution for this problem. However, minimum requirements include the following:

- The system will allow users to register and login to the system. Here's a sample registration/login page below. Make sure to implement basic error checking / validation code. For example, valid password must be at least 6 characters. Passwords must be encrypted/hashed using the one-way function we discussed in class. These must be stored securely in a database. Once users are logged in and authenticated, they will be able to store their history. Keep in mind that only registered users can use the services available in this site. Information about users, which includes userID, first name, last name, email, password, etc. must be stored in a database table (ex: car_owners).

  Images below are just sample views – You can design your own look and feel and your own forms

- The most important operation users are able to do on the site is to value their cars. You might create a database table with car values (carID, make, model, year, and base-price). For simplicity, you don't really have to add too many cars. You may assume that your system stores information about perhaps 20 cars. As I said, you can spend more time in creating a drop-down menu as shown below, or you may use a simple web form. The more you spend time and efforts on this project, the more points/extra points you will receive.



- Speaking of database tables, you should now link the two tables that you have created, *car_owners* and cars and implement the appropriate relationship between them (it is a many-to-many) relationship. Thus, a third table, perhaps named *users_cars*, need to be created. In addition to the PK from each table, this new table may contain fields that represent the price if the car is to be sold to or bought from a private owner, the suggested retail price if the car to be bought from a dealership, and the certified pre-owned car price. (see item below)
- When you display the final output (car details and price details) and when you store the user's history which includes the cars that the user has priced, you must also display car value if sold to a private owner, suggested retail price (15% more than suggested retail), and certified pre-owned value (10% over private owner value). Remember, your system must provide a way for users to display the cars they have valuated (users history)
- The final value of the car is based on the condition of the car. For instance, you may assume that the car price stored in the database table represents a base price for a car with a "Fair" condition. However, the user should be able to choose a different condition and the final car value should increase. You must have the following car conditions (Excellent, Very Good, Good, and Fair). For instance, you may assume that a car with Good condition is 5% more than the base price, and so on.
- Base price adjustment: The mileage determines how much the base price of a car will be reduced. You may follow a simple calculations procedure as follows:
    - 0-10k miles: final price will be similar to the base price stored about the car in the database.
    - 11k – 40k: 5% less
    - And so on
- Options: provide a list of at least 5 desirable options, each of which will add a fixed value ($50) to the car price. For simplicity, you may assume that all options are uniform to all cars.

Be creative and above all, enjoy the experience……

Submission Guidelines

A readme file needs to be included with your project. Your project readme file must include a complete description of your project's design and implementation. Things to consider when writing the readme file:

- Any design and implementation assumptions.

- Any challenges that you faced in the design or implementation of your program.
- Which requirements you didn't do and which requirements you added to the system
- Known bugs (if I find bugs in your code that you did not report, I will assume you didn't test your code well enough to find the bug)
- Any specific procedures to testing the system.

Submit your code and the readme file in a zip archive that's named *GroupName_Project1.zip*