​​**Conestoga College - ACS & IT**

**Programming Microsoft Web Technologies – PROG2230**

**Practice Review for Final**

### 

### **Starting point**

~~Create a new ASP.NET CORE Web App (Model-View-Controller) Project named~~ *~~PROG2230\_AutoParts\_FinalReview~~*~~, and make sure you pick AUTHENTICATION TYPE as~~ **~~NONE!~~** ~~We will be manually adding authentication manually. Note: on the final you will be given some starting code in a project to build on but for review, starting from scratch is good practice.~~

### **What you need to do**

~~This Example is based on a basic Auto Parts Store - i.e. a way of managing auto parts for vehicles. Cars can have many parts (i.e. there is a 1-to-many relationship between Cars and Parts).~~

~~The following are the steps you need to take to complete this review~~…

~~Create Models, Views and Controllers for the auto part store to manage car parts. You will need to create 3 models:~~

* ~~Manufacturer – Name, Address, Website, City, Zip/Postal Code~~
* ~~Car – Make, Model [we don’t have to use 3NF here for time purposes], Year~~
* ~~Part – Part Name, Items In Stock, Price~~

### Part 1 - EF Core

~~Implement and utilize a code-first DB between Manufacturer, Car and Parts. Seed your database with 2 Car Manufacturers, a car for each and 2 parts for each car.~~

~~Add a migration, update your database, and define or scaffold views and controllers for each:~~

* ~~Make the car manufacturer the index page~~
* ~~Make the link of the car manufacturer goto a list of cars for that manufacturer~~
* ~~Make the link of the car go to a list of parts for that car only~~

### Part 2 - Validation

~~Add validation for adding/editing Manufacturer, specifically~~:

* ~~The fields: Website, Address, City, and Zip/postal code are all required~~
* ~~Name cannot be more 64 characters~~
* ~~The Website must be a valid URL~~
* ~~Zip/postal code must be in a valid format. For simplicity, assume that a valid US Zip or UK Postal code is in one of the following two formats:~~
  + ~~A UK postal code of the format: 2 capital letters, a digit, a space, a digit, and 2 capital letters - e.g. NW8 9AY~~
  + ~~Or 5 digits US zip code - e.g. 90028~~
* ~~In addition to these value-based validation requirements, when there are validation errors, ensure that a generic message above the form appears that indicates that there are errors in the form~~.

### **Part 3 - Authentication**

Add authentication to your project. At the BARE MINIMUM, add Register and Login to your project (Think AS5!)

* ~~Configure the following password criteria:~~
  + ~~Passwords must be at least 6 characters.~~
  + ~~Passwords must have at least 1 digit.~~
  + ~~Passwords must have at least one special - i.e. non-alphanumeric character.~~
* Restrict the actions of adding/editing/deleting of Manufacturers, Cars and Parts to authenticated users.
  + If an anonymous user tries to add/edit/delete a car or part, then they should be redirected to a login page.

### ~~Part 4 - Tag helpers, partial views, components~~

* ~~Add a simple partial view that consists of the links to add Cars~~
* ~~Then reference, i.e. make use of, that partial view everywhere where those links appear in the program~~

### Part 5 - Cookies & sessions

~~Save an audit of all the modifications (add/edit/delete actions) of your application entities (i.e. Manufacturers, Cars, and Parts) in the current session and a way to view the history of modifications in the current session and a link to that view in the footer. That is:~~

* ~~Add code to read and write action messages to the current session~~
* ~~Define an ActionHistoryController with an Index action that can be completed to read the session history and pass that off to an ActionHistory/Index view to display actions in the history - a simple unordered list should suffice.~~
* ~~And finally, in the footer, add a link to take the user to a view of all the actions taken in the session~~