**Conestoga College - ACS & IT**

**Programming Microsoft Web Technologies – PROG2230**

**Final Exam Tues Dec 14, 2021 (9am-1pm)**

**Total marks: 27 Worth: 35%**

### Introduction

This Final is a hands-on exam. You are given some starting code (found in the *Final* folder in eConestoga) and you must follow the steps outlined in this document to complete the code for final submission. The instructions here are sometimes complemented by helpful comments in the starting code. Good luck, and feel free to submit multiple versions but, as always, only your last (i.e. most recent) solution will be graded.

### Please note

Your solution ***must only use techniques covered in class***. Also, I remind you to make sure you do your own work on this midterm and resist any urge to copy code from any other source - e.g. your classmates, the web, etc. Everyone’s solution will be run through [Moss](https://theory.stanford.edu/~aiken/moss/) to check for academic integrity violations. There is zero-tolerance for such violations and any encountered with be dealt with in accordance with [Conestoga’s policy](https://lib.conestogac.on.ca/academic-integrity/penalties).

### Some advice

It is always best to make small changes and test it - i.e. work in small *edit-run-test* cycles. The starting code is a runnable app so do everything you can to keep the app runnable as you progress through the steps below. It is ***far better*** to have something runnable at the end that might not be complete than something you thought was complete but doesn’t run!

### How will it be graded?

Accompanying the exam will be an Excel marking sheet that details how your grade will be calculated so you will want to make sure that you are doing everything as it is laid out there.

### What/how to submit?

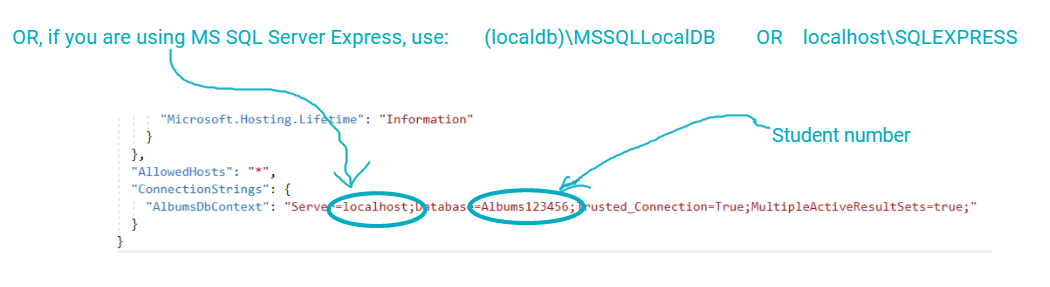
Zip up your entire solution into ***one zip file*** and submit that file to the eConestoga dropbox for the midterm.You can submit multiple solutions but only the latest (i.e. most recent) one will be looked at and evaluated.

### Setup

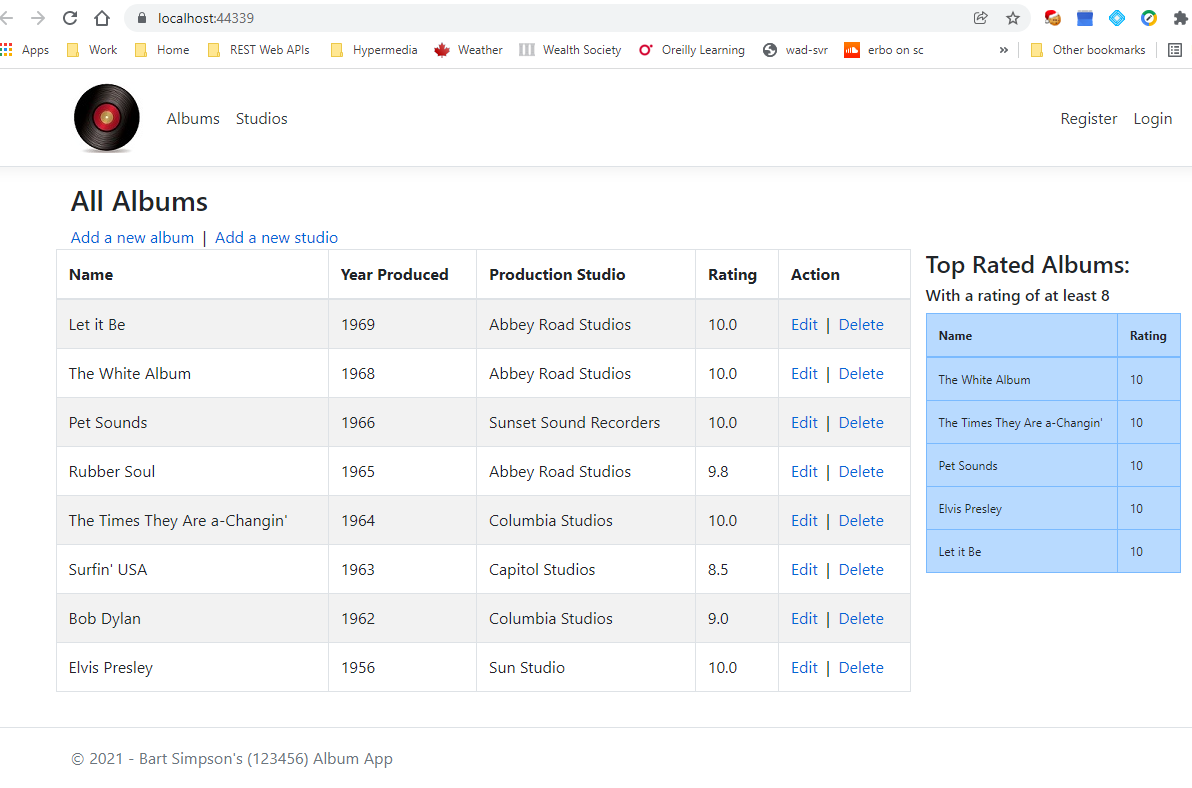
After some basic setup, the starting point of code contains enough code to display all the seeded Album and Studio data in the Db. Specifically, to get it running, complete the following steps:

* Download and extract the “*final-starting-code.zip*” file in the *Final* folder in eConestoga
* Put your full name (and student number) in the footer - i.e. defined in the *\_Layout.cshtml* view
* Add your student to the end of the DB name in the connection string (see image below) in *appsettings.json*
* Run *Update-Database* in pkg mgr console
* Then you should be able to run the *AlbumsApp*, making sure is the startup project if necessary

Thus, for a student named Bart Simpson (with student number: 123456) you should have a connection string that looks like this:



And, when you run the app, it should look like this..



### What you need to do

This Final exam is based on a basic Albums App - i.e. a way of managing record albums. Thus, the primary entity of interest for this app is an Album. It by no means captures everything about albums but it does try to capture that Albums are recorded in a Studio, but of course studios can record many albums (i.e. there is a 1-to-many relationship between Albums and Studios).

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

### Part 1 - Validation

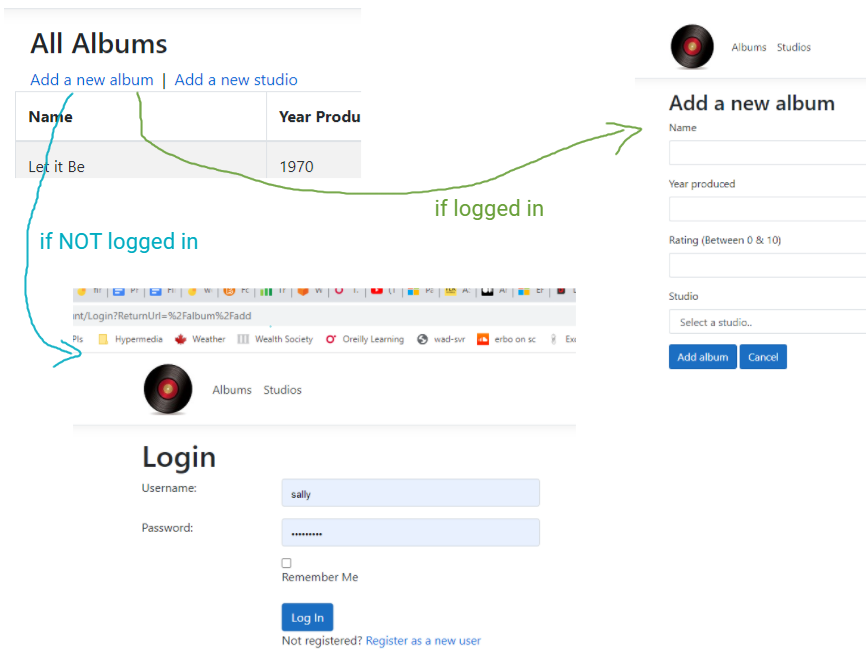
Add validation for adding/editing Studios, specifically:

1. The fields: Name, Website, Address, City, and Zip/postal code are all required
2. Name cannot be more 64 characters
3. The Website must be a valid URL
4. Zip code must be in a valid format. For simplicity, assume that a valid Zip code is in one in the format you see in the seeded data, namely a 5 digit US zip code - e.g. 90028
5. 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 2 - Authentication & authorization

Complete the authentication and authorization infrastructure that is already in place by taking the following steps:

1. Configure the following password criteria:
   1. Passwords must be at least 6 characters.
   2. Passwords must have at least 1 digit.
   3. Passwords must have at least one special - i.e. non-alphanumeric character.
2. Restrict the actions of adding/editing/deleting of both Albums & Studios to authenticated users.
   1. Note, to test this properly you will have to register a user.
   2. Also, to clarify what this means, if an anonymous user tries to add/edit/delete an album or studio, then they should be redirected to a login page, as such:

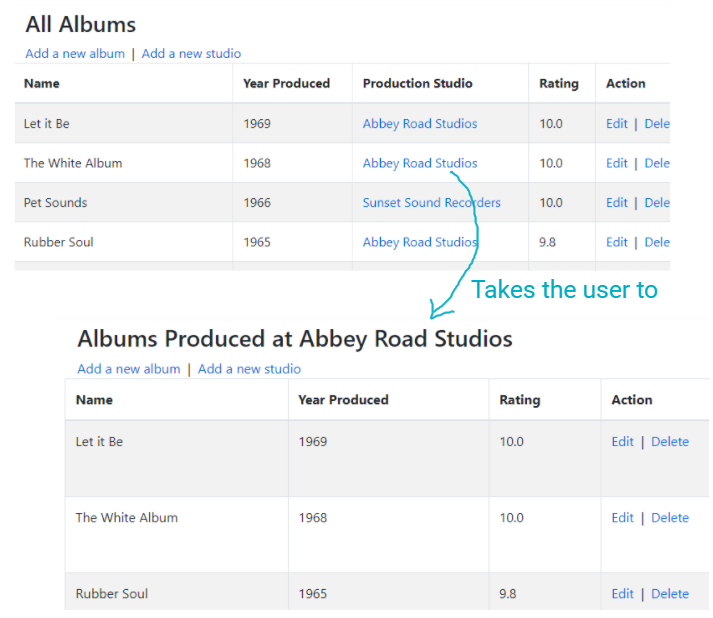


### Part 3 - Tag helpers, partial views, and view components

1. Add a simple partial view that consists of the links to add Albums & Studio
2. Then reference, i.e. make use of, that partial view everywhere where those links appear in the app
3. Update the existing *TopRatedAlbums* view component. Currently it is hard-coded to display 5 of the top-rated albums (i.e. albums with a rating of at least 8). So, your tasks here are to:
   1. Update it to accept a parameter the specifies the number of top-rated albums to display
   2. And update the reference where the view component is used to pass the number of albums to display to the view component

### Part 4 - EF Core

1. Implement and utilize a complete 1-to-many between Albums and Studios:
   1. Turn the studio name in All Albums view into a link that, when clicked, shows only the Albums recorded at that studio. This requires you to:
      1. Implement an Action method to handle such requests
      2. And a view to display the results, namely a table of only albums produced at the given studio, without a Studio column but instead the studio name in the page heading.
   2. In the end, this should look something like:



### Part 5 - Cookies & sessions

Implement a page counter using both sessions and cookies that, on the Album/List page states: "*You have visited this page a total of N times, and M of those visits are from this current session*", Where:

* N is a counter stored in a persistent cookie
* M is a counter stored in session

You are free to place it elsewhere but one possible location is after the table, for example:

