**Conestoga College - ACS & IT**

**Programming Microsoft Web Technologies – PROG2230**

**Midterm Wed Oct 20, 2021 (5-8pm)**

**Total marks: 40 Worth: 35%**

### Introduction

This midterm is a hands-on exam. You are given some starting code (found in the Midterm 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’s laid out there.

### What/how to submit?

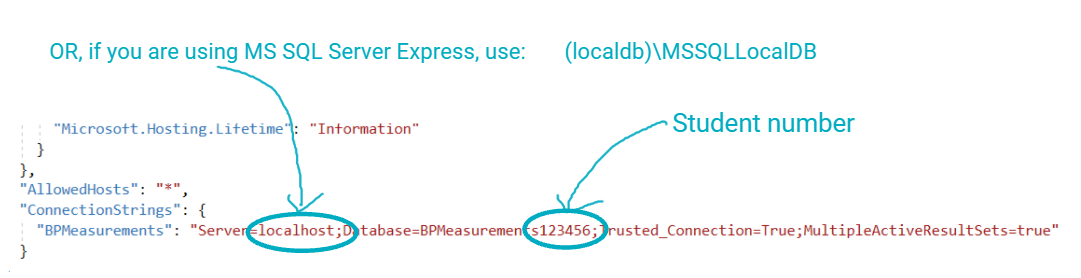
Zip up your entire solution into ***one zip file*** and submit that file to the eConestoga dropbox for the midterm.You can submit multiple solutions but only the latest (i.e. most recent) one will be looked at and evaluated.

### Setup

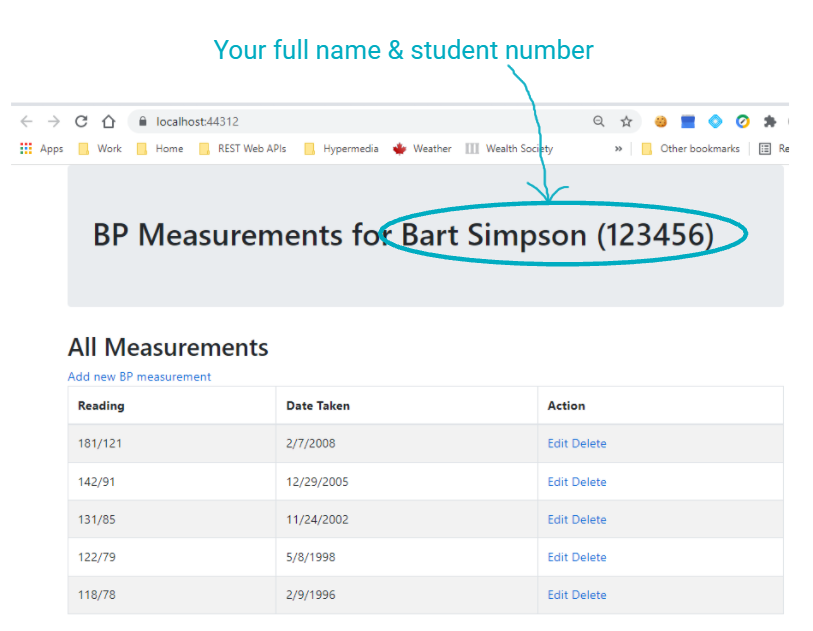
~~After some basic setup, the starting point of code contains enough code to display all 5 of the seeded BP measurements in the Db. Specifically, to get it running, complete the following steps:~~

* ~~Download and extract the “~~*~~midterm-start-code.zip~~*~~” file in the midterm folder in eConestoga~~
* ~~Put your full name (and student number) in title of Home/Index 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 BPApp, making is the startup project if necessary~~

~~For a student named Bart Simpson, with a student number 123456, you should have a connection string that looks like this:~~



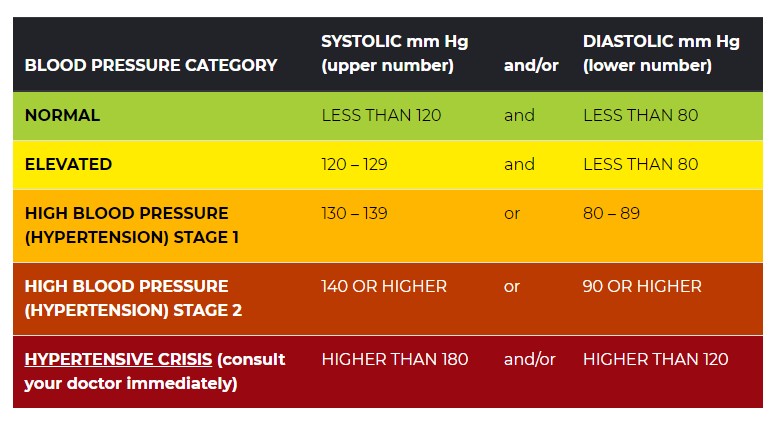
And, when you run the app, it should look like this..



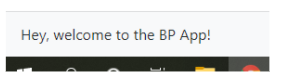
### What you need to do

~~The midterm is based on a BP (i.e. Blood Pressure) app that is very similar in structure to the database-driven web apps that have been covered in the textbook and in class. It allows users to view, add, edit, and delete BP measurements - i.e. the basic CRUD operations. To complete it you need to complete the following high-level steps~~:

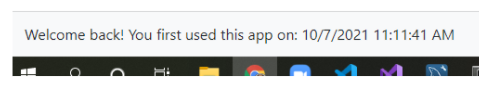
1. ~~Implement add new & edit existing BP measurements. To do this you will need to:~~
   1. ~~Implement GET & POST methods for both Add & Edit actions~~. ~~Empty method definitions for Edit are in place in BPMeasurementController.~~
   2. ~~Complete the BPMeasurement/Edit view~~
      1. ~~Add the missing input fields to bind to BPMeasurement properties~~
      2. ~~Note: Dates are tricky so what is provided in the starting code already includes a “datetime” input field in the BPMeasurement/Edit view for the MeasurementDate property (of type DateTime). Its functionality depends on the use of jQuery-ui’s datepicker component and that infrastructure is already set up for you.~~
2. ~~Implement Delete of existing BP measurements. To do this:~~
   1. ~~Implement GET & POST methods for the Delete action~~
   2. ~~Add a BPMeasurement/Delete view that allows the user to confirm that they want to in fact delete that measurement, or cancel by returning to the “all measurements” (i.e. Home) view~~.
3. Add validation:
   1. ~~All 3 properties should be required~~
   2. ~~Systolic & Diastolic should both be integers, and Systolic should be between 20 and 400,~~ ~~and Diastolic should be between 10 and 300~~
   3. ~~Add a placeholder for validation feedback to the user in the BPMeasurement/Edit view~~
4. ~~Next, you are going to update your data model slightly by introducing a new property - namely~~, ~~what position was the person in when the BP measurement was taken:~~ *~~Sitting~~*~~,~~ *~~Standing~~*~~, or~~ *~~Lying down~~*~~.To do this you will add a migration that:~~
   1. ~~Adds a Position class that has 2 properties:~~
      1. ~~A string ID (that will correspond to its primary key in the database)~~
      2. ~~A string name property~~
   2. ~~Make the collection of these Positions available as a new property on the~~ *~~BPContext~~* ~~class~~
   3. ~~And then seed the DB with instance for each of the 3 positions, namely:~~ *~~Standing~~*~~,~~ *~~Sitting~~*~~,~~ *~~Lying down~~*~~, using an ID of your choosing and update the seeding of the original data accordingly~~
   4. ~~Once done, using the package manager console:~~
      1. ~~Add a new migration~~
      2. ~~Then update the database~~
5. ~~Now that you have updated your data model, you need to update the views accordingly:~~
   1. *Home/Index* should show the *Position* in a new column:
   2. ~~And~~ *~~BPMeasurement/Edit~~* ~~(in both Add & Edit scenarios) should provide a dropdown of position options for the~~ *~~Position~~* ~~field~~
6. ~~Add a method or readonly (i.e. get only) property - your choice - that returns the category for a BP reading based on the following chart:~~



1. ~~Use the following as your categories:~~
   1. *~~Normal~~*
   2. *~~Elevated~~*
   3. *~~Hypertension Stage 1~~*
   4. *~~Hypertension Stage 2~~*
   5. *~~Hypertensive Crisis~~*
2. ~~And then, in the Home/Index (i.e. “all BP readings”) view, add a new column that uses your method or property to display the category for each reading in a new column~~
3. Next, use the existing CSS classes (already defined in the site.css file, which is already referenced in the \_Layout) to color code both the reading and category text fields in this view.
   1. Those CSS classes are:
      1. *bpNormal*
      2. *bpElevated*
      3. *bpHypertensionStage1*
      4. *bpHypertensionStage2*
      5. *bpHypertensiveCrisis*
   2. ~~Hint: You will notice that the CSS classes to style the text accordingly all begin with “bp” but then are suffixed by the BP reading category without any spaces. Thus, in your View, all you need to do is a little text manipulation of the category to assign the right CSS class to the two~~ “*td*” elements for reading and category.
4. ~~Next, using the~~ *~~TempData~~* ~~dictionary, show a “~~*~~last action message~~*~~” every time you are returning to the “all BP measurements” (i.e Home) view from having completed an Add/Edit/Delete action.~~
   1. ~~The message should appear below the jumbotron but above the “~~*~~All Measurements~~*~~” header - see the screenshot below.~~
   2. ~~For Edit and Delete actions, the message should include the BP measurement (i.e. systolic/diastolic) and the date it was taken, whereas for the Add, show just the measurement.~~
5. ~~And finally, use a Cookie to track when a user first visits the app. With this cookie you can then control a welcome message that shows up in the bottom left hand corner of at least the~~ *~~Home~~* ~~and~~ *~~BPInfo~~* ~~pages~~, ~~i.e. in the footer that is fixed to the bottom of the page. Namely:~~ 
   1. ~~If the user has never visited the app before the message should say something like:~~



* 1. ~~Whereas if the user has visited the app before, a welcome back message that displays the date they first used it, something like:~~



### ~~How it should look in the end~~

~~Putting all this together, the following represent some screenshots of a completed version:~~

