**Conestoga College - ACS & IT**

**Programming Microsoft Web Technologies – PROG2230**

**Assignment #4: ASP.NET Core: More Validation & EF Core**

**Due: Friday Nov 26, 2021 (by Midnight)**

**Total marks: TBD Worth: 6%**

**Introduction**

The goals of this assignment are to:

* Apply some of the more recent validation techniques we learned in class.(ch11)
* Develop a web app from the database-first perspective.
* Apply LINQ and EF Core to a slightly richer data model than we have worked with previously and maximize the “*hypermedia design*” of your app to reflect the relationships between the entities in that data model.(ch12 one to many many to many)
* ~~Apply the "~~*~~soft delete pattern~~*~~” and use it to support an undo of a delete action~~.
* Apply a "*paging*" solution to the vendors by clustering them into alphabetic groups.

**The general idea**

~~With this assignment you are given some freedom to design this app as you see fit, based on the material you have learned so far. This is true both with the look and feel of the app and your code design choices. As for the look and feel, the only requirement is that your web UI must look professional and be styled consistently across all pages. You don't have to use Bootstrap but it is what we have explored the most in this course so it is likely the most obvious choice.~~

**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 assignment 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)~~.~~

**~~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 assignment.You can submit multiple solutions but only the latest (i.e. most recent) one will be looked at and evaluated.~~

**What do you need to build?**

~~You need to build some basic functionality around "~~*~~accounts payable~~*~~". In~~ [~~simplest terms~~](https://www.investopedia.com/terms/a/accountspayable.asp)~~, this database is about a company's way of recording the amounts due to its vendors or suppliers for goods or services received that have not yet been paid for. To do this, you are going to apply the database-first approach and use EF Core to generate your C# entity classes from the "~~*~~ap~~*~~" database. The SQL file needed to set up the schema and insert some initial data is provided in the assignment folder. This script is a slightly modified version of the AP (Accounts Payable~~) ~~database from the book~~ [~~Murach's MySQL by Joel Murach~~](https://www.murach.com/shop/murach-s-mysql-3rd-edition-detail)~~, ported from MySQL to SQL Server's T-SQL. See Appendix A for a diagram of this database’s schema.~~

~~Once you have your code~~ generated from this database ~~you need to develop an app that supports the following basic features:~~

* ~~Displays the vendors in alphabetic groups, e.g. “~~*~~A - E~~*~~” represents all the vendors whose name starts with a letter in the range A to E~~
* Add/edit a vendor with some validation requirements
* ~~Soft delete a vendor~~
* ~~Undo the action of deleting a vendor~~
* View some more details about a vendor including its invoices and be able to return to the previously viewed alphabetic group of vendors
* For each invoice of a vendor, be able to see both the total and the line items of the invoice
* And finally, be able to add a new line item to an invoice

To achieve this, there are 3 main pages that you need to develop:

* ~~The Vendors page(patienList)~~
* The Add/Edit Vendor page(patienRecord)
* The Vendor's Invoices page(patienPrescriptionList)

~~To flesh out the details of what you need to build, let’s consider each of these in turn.~~

The Vendors page:

This page should show:

* ~~A link to take the user to the form to add a new vendor~~
* ~~A group of links to display vendors from other alphabetic groups~~
* ~~The link for the current group should be highlighted appropriately~~
* ~~A table of the vendors with the columns:~~
* ~~Name~~
* ~~Address~~
* ~~Actions. This means three links~~:
* ~~An “~~*~~Invoices~~*~~” link to take the user to page showing all the invoices for that vendor~~
* An “*Edit*” link to take the user to the Add/Edit view to edit/update the current vendor
* ~~A “~~*~~Delete~~*~~” link that deletes the current vendor. This result in two things:~~
* ~~The vendor in question being marked for “soft delete” in the database~~
* ~~A return to the previously viewed alphabetic group of vendors with a link to optionally undo that delete action~~
* ~~An~~ *~~undo~~* ~~link, after a delete action, and clicking it should undo the previous delete action. A few considerations about this~~:
* ~~This should lead to the vendor appearing in the table again~~
* ~~The undo action should not remain available indefinitely. You could either:~~
* ~~Have it go away upon the next page refresh~~
* ~~Or fade away after some period of time - just how Gmail gets rid if its undo action links after 6 seconds~~
* ~~This could be done easily by using jQuery’s “fadeOut”, or some other similar animation.~~

Here is a potential mockup of the Vendors page:



And after a delete, it should be something like this:



The Add/Edit Vendor page:

~~This page should:~~

* ~~Show:~~
* ~~A header describing what action is being completed~~ - ~~Add or Edit~~
* ~~Labels and input elements for each field to be entered for a vendor~~
* ~~A button that enables the successful completion of the Add or Edit action~~
* ~~If successful the user should be returned to the previous alphabetic group they were viewing~~
* ~~Otherwise the validation errors should be highlighted - please the section below for all the validation requirements for a vendor~~
* ~~A button that cancels the action and returns to the previous alphabetic group they were viewing~~

~~Here is a potential mockup of the Add/Edit Vendor page, shown here in “Add” mode:~~



The Vendor's Invoices page:

This page should:

* ~~Show:~~
* ~~A heading the references that the user is viewing the invoices for a given company~~
* ~~The company’s address~~
* ~~Their payment terms~~
* ~~Have a link that returns to the previously view alphabetic group of vendors~~
* ~~Their invoices, and by default showing the~~ details - i.e. line items ~~and total for the first invoice~~
* ~~Thereafter it show the same invoice details for whatever invoice is clicked~~
* For the current invoice, the user should also be able to add a new line item by entering the description and amount, which should show the new line item in the table and the total adjusted appropriately but otherwise stay on the same view.

Here is a potential mockup of the Vendors page:



Putting some of this together, the following diagram highlights the main page flows:



**Validation requirements for adding/editing a Vendor**

~~When adding/editing a vendor you need to implement the following validation requirements:~~

* ~~Name, Address 1, City, State, Zip code~~, ~~phone, default terms, and account number are all required~~
* ~~Name cannot contain special characters~~
* ~~State must be either a valid State name or a valid 2 letter state code - but case doesn’t matter~~
* ~~Vendor contact's phone number must be:~~
* ~~In a valid 10 digit phone number format~~
* ~~And the number must be unique within the database~~
* ~~Zip code must be in a valid format~~
* ~~In addition to these value-based validation requirements you need to support the following UI related validation requirements~~:
* ~~If there are validation errors, then:~~
* ~~There must be a generic message above the form indicating that there are errors in the form~~
* ~~The fields in error highlight must highlighted in 3 respects:~~
* ~~It must have a red border~~
* ~~A different background colour - i.e. different from the valid or empty fields~~
* ~~And a field-specific error message, in red, after (i.e. to the right of) the field in error~~
* If the phone number the user has entered for the vendor is not unique, then they should be informed of that conflict immediately - i.e. they should not have to wait to submit the add/edit button.

**And finally**

~~Please do not modify the schema of the provided database, or even its name for that matter. Why? Two reasons:~~

* ~~This is meant to at least simulate the scenario wherein you have to work with a given database, like it or not, and you aren't able to modify its schema.~~
* ~~When grading, I want to be able to simply run your solution against my instance of the database~~.

**Appendix A**

The following is a schema diagram for the “ap” database derived from the Murach MySQL book.

