**Conestoga College - ACS & IT**

**Programming Microsoft Web Technologies – PROG2230**

**Assignment #1: Introductory Data-driven ASP.NET Core MVC Web Apps**

**Due: Friday Oct 1, 2021 (by Midnight)**

**Total marks: 40 Worth: 6%**

### Introduction

The goal of this assignment is to build a basic database-driven ASP.NET Core MVC web app as a way of getting an overview of ASP.NET Core before we start diving deeper into its various topics like routing, binding, etc. This app will look and behave very similarly to the *Movies List web app* that is covered in Chapter 4 of our [Murach book](https://www.murach.com/shop-books/web-development-books/murach-s-asp-net-core-mvc-detail), as such it will very much ***be your guiding example***! As part of this you will:

* Develop a first basic web app in part I, using the Entity Framework (EF) Core code first approach.
* And then in part II, introduce a change to the data model (and therefore database) as part of an improvement to the web app so that you get a start down the path of using EF Core’s support for migrations.

### First, a word of caution:

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. Not only is this the only way to learn how to program but also 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). I also remind you that if you are not typing syntactically correct code in yourself you are not learning to program! Finally, I’ll note that these assignments are very representative of what will be expected of you on exams so it is very much in your interest to ensure that you are capable of doing them on your own.

### Hints

The general idea is that there will be time to work on these questions in class and, if necessary, I can offer hints if I see that you are struggling with certain parts. Also, you should try to pace out your work on this assignment over the coming weeks.

### How will it be graded?

I’ll say from the outset that there is absolutely no reason to not get 100% - the emphasis here is simple: get your hands dirty coding to solve basic problems and, if needed, I will do what I can to help steer you towards a working solution.

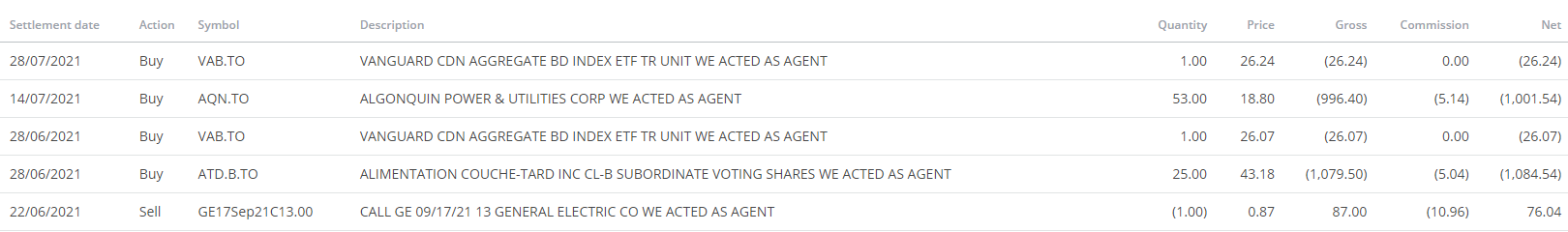
Accompanying each assignment will be an Excel marking sheet that details how your grade will be calculated so you obviously want to make sure that you are doing everything as it’s laid out there.

### 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 will you build?

Suppose an investor wants an app to keep track of the stock transactions that he or she makes in their personal investment portfolio. Here is a screenshot of a fragment of my own Questrade account that displays a few sample transactions:



So, for this assignment, you need to develop an ASP.NET Core MVC web app that:

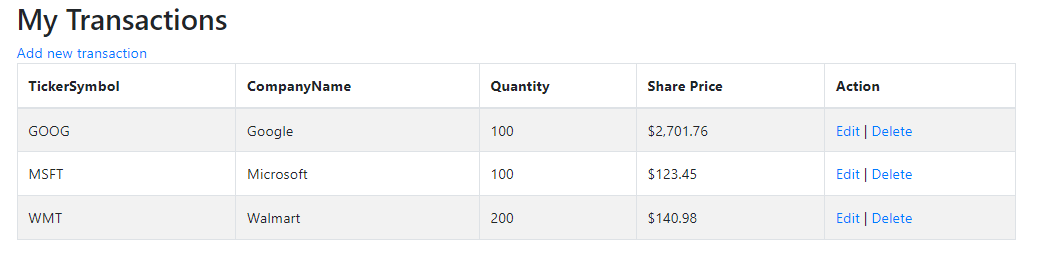
* Stores transactions (e.g. the investor bought 100 shares of MSFT at $216.34 per share) in a MS SQL Server (Express or Developer edition) database
* The home/main page of the app will display all the stored transactions in an HTML table with the following columns:
  + Initially (i.e. Part I) just:
    - Ticker symbol, company name, quantity, share price, and an action
  + And then with a migration (i.e. Part II) you will add:
    - Transaction type, commission fee, gross value, and net value
  + Please see the screenshots at the end of the assignment to get a better sense of how this should look
* Provide a link on that main page that takes to user to a new page with a form to add/record a new transaction
  + On the page the user should be able to enter valid fields for a transaction and add it to the database and then get returned to the main all transactions page
* Each transaction in the HTML table, on the main page, will be accompanied by two links:
  + An Edit link to edit the recorded transaction, which takes you to a form filled in with the transaction’s current values and a button to update the transaction if altered.
  + A Delete link to delete the recorded transaction, which takes you to a page that the user can use to confirm they want to delete it.
  + Both these pages should do the desired edit/delete action in the database and then return the user to the main all transactions page
* All the pages: Edit, Add, Delete should also have a cancel button that takes you back to the main “all transactions” page
* Your web app should also use the basics of ASP.NET Core’s validation capabilities to ensure that:
  + All the fields Ticker symbol, company name, Transaction type, quantity, share price are required
  + Quantity must be an integer value greater than zero
  + Share price must be a double value, also greater than zero
* Also, if the user attempts to add/edit a transaction with invalid fields then the add/edit attempt should be abandoned and appropriate and informative error messages, in red, should be returned to the user.
* Also, seed your database with at least the following two transactions:
  + Buy 100 shares of Microsoft (MSFT) at $123.45 per share
  + Sell 100 shares of Google (GOOG) at $2,701.76 per share
* Database naming convention: because I want to be able to easily run each of your solutions, it is very important that you abide by the following naming convention for your database so that your databases don’t clash on my machine when grading, namely:
  + ***TransactionRecord<YourDomainLoginName>***
  + For example, if your domain login name is *SJones1234*, then you should name your database:
    - ***TransactionRecordSJones1234***
  + Note: If I am unable to run your web app because of a violation of this naming convention ***you will lose marks***!
* About the Gross & Net value columns:
  + They are computed columns - i.e. computed based on values of some of the other properties on the model classes, namely:
    - Gross value is just *Quantity* multiplied by *SharePrice*
    - Net value is gross value plus or minus (depending on the transaction type) the *CommissionFee*
      * The screenshot below should help
    - These are best implemented as methods on the model class
  + In typical financial or accounting style, the values should include parentheses if they are “negative”, or a debit
    - Again, the screenshot below should help

Using Entity Framework (EF) Core to develop data-driven web apps is very elegant and powerful but it also comes with its own subtleties and challenges, but a key aspect of it is its support for continually evolving your data model through the use of migrations. As such, just as the authors of the Murach book do in chapter 4, in this assignment you will complete the app in 2 parts so that you can evolve the app using a migration. That is, to get full marks on this assignment you need to include an initial migration and a second migration that adds the transaction type. Thus we break the assignment down into 2 parts...

### Part I:

Get a basic version of the app working that focuses first on only the following fields: ticker symbol, company name, quantity, and share price

Here is a screenshot of the main “all transactions” page of the app after this initial step:



### Part II:

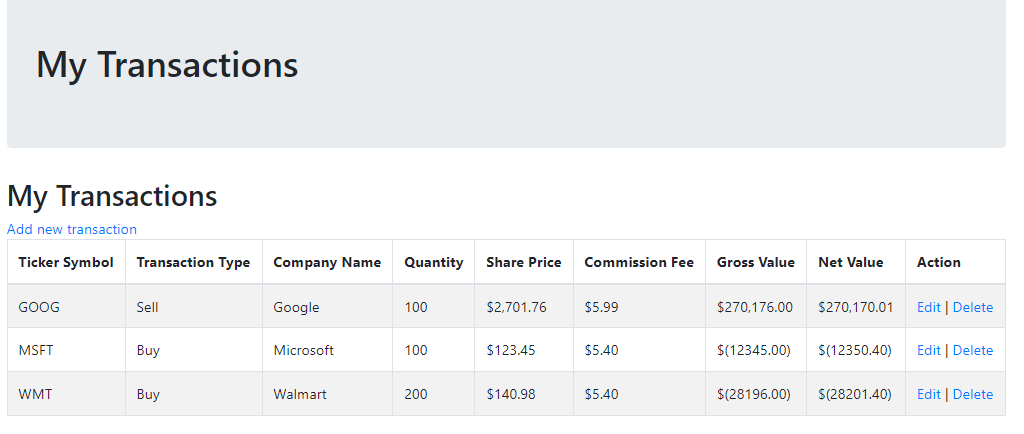
Just as the authors of the Murach book do in chapter 4 with the *Movies List web app*, wherein they add a migration that supports a movie *Genre*, in this part you will add a ***TransactionType*** class to your model and update the database accordingly.

The transaction type should include:

* A primary key (not an auto-generated one)
* A name - e.g. Buy or Sell
* A commission fee - e.g. I used:
  + $5.99 to sell shares
  + $5.40 to buy shares

### Screenshots of the Final Web App

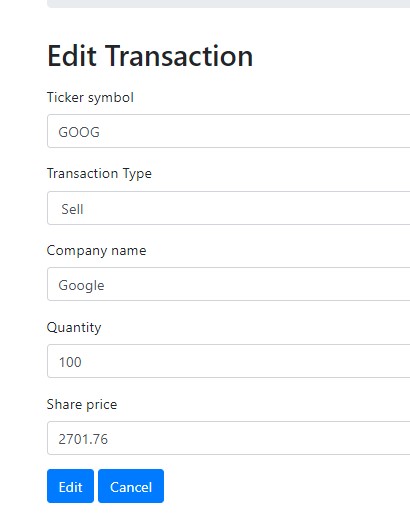
The following are screenshots of the final Web App to help give you guidance. First, the main “all transactions” page:



The “Add a new transaction” page:



The “Edit a transaction” page:



The “Delete a transaction confirmation” page:



A page highlighting some validation errors:

