Web Programming VI (420-H60-HR)
Assignment 2 – Authentication and Starting ReST

Date Assigned:	Friday, September 27
Due Date:	**Thursday, Nov 1, 23h50**(intermediate milestones as lab marks)

Learning Objectives

Upon successful completion of this assignment, the student will be able to:

- Add Authentication and Authorization to a project using identity
- Set up and use ReST Web Services

Setup

You will have to make a copy of your first assignment and the database it uses. Create a copy of your assignment 1 folder and call it *initials*H60A02. Do not change the name of your solution; leave it as *initials*H60Store.

Copy the database; call the new database H60Assignment2DB_*yourinitials* (Use Tasks-> Copy in SQL Management Studio). Open the solution in your assignment 2 folder. Change the references in the assignment to point to the new database. Change the name of the generated context model to also reflect the new database.

Make sure that your assignment still works. Once you have it working, add a second **project** to the solution. This project is called *initials*H60Customer (once again, naming is important). Add a **third project** to the **solution.** This project is called *initials*H60Services.

You are going to be creating a non-.NET project as well in assignment 3. If you want to do that now, please create a folder called *yourinitials*H60Manager. This must be at the same level as your solution (NOT THE PROJECTS, THE SOLUTION). This will be a JavaScript application and will be detailed as part of assignment 3.

This is not just a coding assignment, you should also demonstrate good design and **robustness**.

Part A: Adding Authorization and Authentication (Store Project)

1. Using the techniques from class add Microsoft Identity database and to the Store app. Make sure you add roles so that a role can be selected on registration. The roles can be added manually to the database, but must be selectable when a new account is registered. There are three roles: **clerk**, **manager** and **customer**. DO NOT HARD CODE the roles in the solution, read them from the database. Managers can create any types of account. Anyone can create Customer accounts.

2. Add login functionality to the Store application so that the no screens can be reached unless the user is Authorized (has logged in). The manager can access all screens in the store. The clerk can access all screens in the store EXCEPT the screen to change the buyprice and the sellprice. The customer cannot access any screens in the store application.

3. If a non-authenticated user attempts to access a page, display the Login page. If an authenticated user attempts to access a page to which they are not authorized, display the Access Forbidden page.

4. We will be adding more functionality here so be careful with your design so that it can accommodate future functionality.

Part B: Convert Database Functions (Store, Service Project)

1. Convert the database functionality from the previous assignment to do with Products and Categories to ReST services. Update the functionality in the Store (should be in the model) to use these new services and ***not directly access anything in the database***.
2. Update the Services project to provide the interface for Products and Categories CRUD via ReST.

Part C: Adding Customers (Services, Store Project)

1. Add REST functions to the Services project to allow the following functionality
   a) Perform CRUD operations on the Customer Table
   b) Remember, deletes cannot be done if a customer has a shopping cart or an order (you must maintain referential integrity).

2. In the Store project, provide an interface to use the rest functions to access the database and perform the following operations:
   a) Get a list of all customers (in alphabetical order).
   b) Add, Update and Delete Customers (if able to delete). Remember all customers must be linked to an account with the proper role to login. How you do that is up to you.
   c) Perform validation on the customer data for the first and last name (letters, apostrophe, dash and space only), email address format, phone number format and province (only Ontario, Quebec, New Brunswick and Manitoba accepted)
   d) Handle all errors returned from database/web service.
   e) Note: You should have server side validation on all your APIs. Don't assume all client side validation is performed.

Part D: Customers Searching Products (Customer Project)

1. Using the ReST functions from Part B, add functionality to the Customer project so that a customer can view/search the products and categories. These views should never display the buyprice from the store, but only the sellprice. The forms/views can look the similar to the ones from the Store, but should look more appealing to the customer.

   Understand for your design that you will be adding functionality to select one or more of these products and add them to your cart in Assignment 3. Think of this while designing the screens.

You will need to provide Views, Model and Controllers. I expect that all database access is through the ReST web services in the Services project and accessed) via skinny controllers and the design patterns taught in the course.

Final Five

Completing the above receives a maximum of 95%. The final 5% is available if you complete at least one of the following extra pieces of functionality.

1. Perform marked deletions rather than deletions on products and product categories. In theory, products could not be deleted after they become part of an order as the person should be able to look up that order and see the product. Careful if you choose this as it can get complex.
2. Provide photos of your products so that the user can see the products when browsing or in the cart/order. (Can not get marks for this if already received in A01).
3. Provide customer creation and account creation from the same screen. Instead of using the Register part of Identity to register a user and the Customer create part to create a Customer and then having one reference the other, do everything on one screen.

Marking

Web Programming VI (420-H60-HR) Assignment 2 (eStore)		**Mark**	
		0%	
	Mark	**Out Of**	**Comments**
Part A Authorization/Authentication			
Database updated with tables		4	
Roles implemented		2	
Authorization implemented		2	
Jumping to Login on non-Authenticated		2	
Displaying Access Denied on attempt		2	
Part B Repository Pattern			
Create ReST Services for all Store and Customer methods		4	
Create DB Context based methods for Services		4	
Proper Repository pattern design and implementation		4	
Part C Adding Customers			
ReST functions for Customer Table		5	
Store Get List		4	
Store Create/update/delete customer		4	
Store link customer and identity login		4	
Part D Customer Project Functions			

Customer Login		4	
Customer view/search products		6	
Functionality			
Interface design and flow Store - Professional		10	Basic (i.e. scaffolded or similar) will get you at most 4/10
Interface design and flow Customer - Professional		10	Basic (i.e. scaffolded or similar) will get you at most 4/10
Design			
Coding style/standards/consistency, minimal hardcoding, organization		6	
Code design/efficiency/abstractions. Follow design patterns and methods taught		8	
Project Organization and packaging, README, works with little effort		10	
Final Five			
Completed One of the Following: - Writing Tests - Photos of products - Customer/account creation in one screen	0	5	
Total	0	100	

To submit

When you have completed the assignment, submit it your github classroom repository.

It is up to you to ensure that your packaging works for me. Please include a README.txt to provide simple steps on how to get your project installed and running.
My expectation is that I can specify my own database server, database name and credentials.
If you have provided accounts to test with, please provide the account name and credentials.
Your goal is that I can get your projects up and running in a few minutes by following your simple instructions. It cannot depend on an existing database or any other co-incidental existing material external to your package, your packaging should be complete and flexible.