MIS 333K Homework 6 Instructions

Please follow these instructions in the following order:

# Create a new database on Azure for this homework. This assignment is unrelated to previous homeworks, so you need to have a blank database. You can use a different server or the same one; it really doesn’t matter.

# Create a new Visual Studio project for HW6. Name it LastName\_FirstName\_HW6. You should make an ASP.NET Web Application. Choose the empty template and the core references for MVC. (These are the same options you’ve picked since HW2.)

# Install Entity Framework and Bootstrap into your project.

# Add the appropriate connection string info to your web.config file.

# Add a DAL folder and put the appropriate AppDbContext file in it .

# Download the Customer.cs and Frequency.cs files from Canvas and add them to your models folder. You can do this by right-clicking on the folder and selecting “Add Existing Item…”

# Change the namespace of Customer.cs and Frequency.cs to reference your project name.

# Build your project by selecting Build 🡪Build Solution

# Enable migrations in your project by typing “enable-migrations” in the Package Manager Console window.

# The enable migrations command should have created a folder in your project called Migrations. Download the CustomerData.cs file from Canvas and add it to the Migrations folder. You can do this by right-clicking on the folder and selecting “Add Existing Item…

# Change the namespace and using statements in CustomerData.cs to reflect the name of your project.

# Replace the existing Seed method in the Migrations/Configuration.cs file with code to call the methods on the CustomerData class. Be sure to update the parameter to reference your project name

protected override void Seed(Homework\_6\_Example.DAL.AppDbContext context)

{

//call method to add frequencies

AddCustomers.AddFrequencies();

//call method to add customers

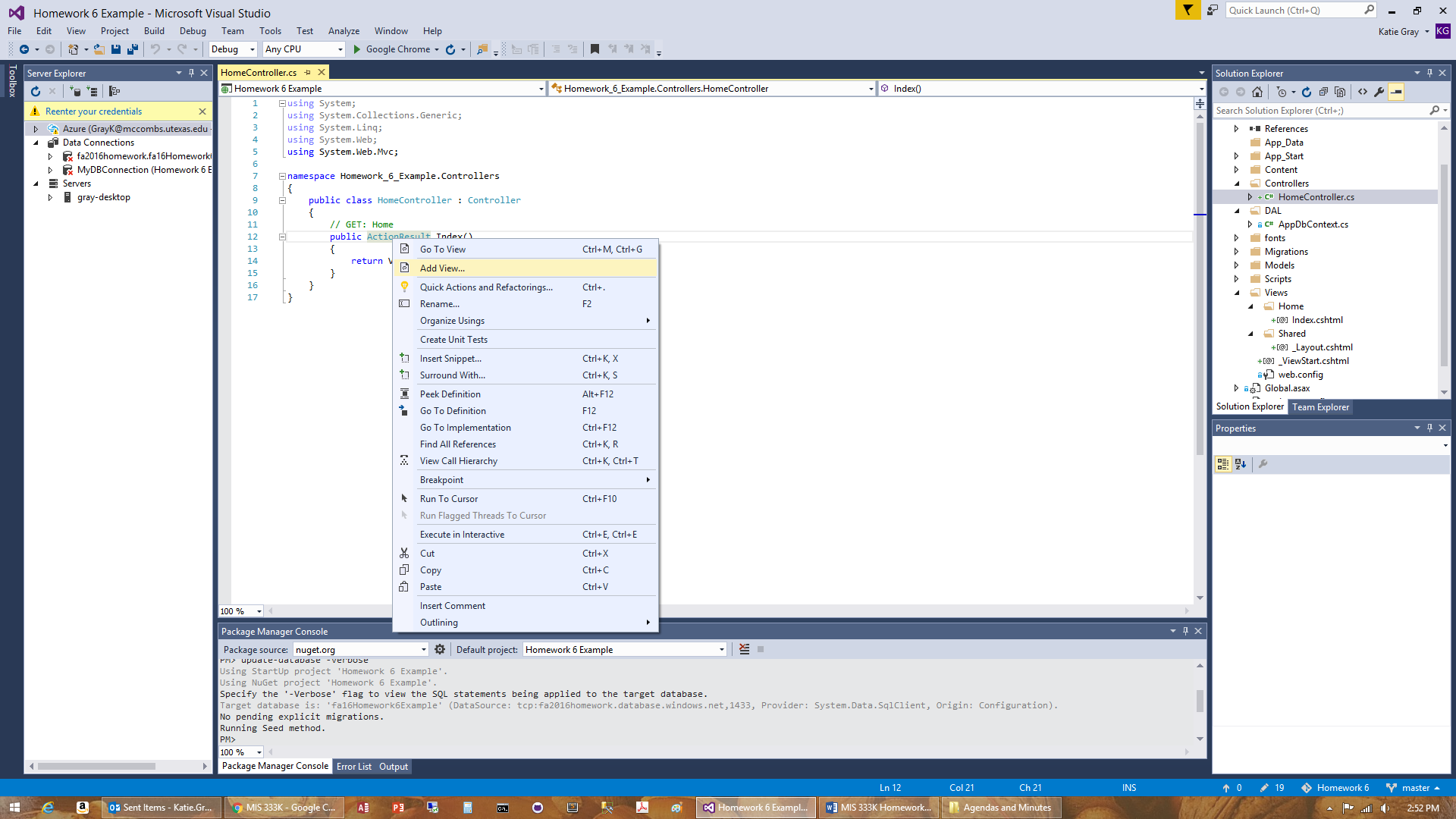
AddCustomers.SeedCustomers();

# }

# Use the Package Manager Console window to add a migration and update the database.

# Add a controller to the Controllers folder. Select empty controller. Name the controller HomeController.

# Add a view from the Index action on the HomeController.



# Select the following options for the new View

# 

# Add a using statement to the HomeController.cs file to give the controller access to your AppDbContext file.

# Add a using statement to the HomeController.cs file to give the controller access to your Models folder.

# Create an instance of the AppDbContext file in HomeController.cs to use in your Controller actions.

# Add the following code to the Index action: return View(db.Customers.ToList());

The completed method should look like this:

// GET: Home

public ActionResult Index()

{

return View(db.Customers.ToList());

}

# **CHECKPOINT: Run the project to make sure everything is working. This should give you a list of all 250 customers.**

# Update the Views/Home/Index.cshtml file to display frequency data for the customer. (Hint: Copy the code in the view for another field and change the field name.)

# Remove the code in Views/Home/Index.cshtml that creates a link to add a new customer. We don’t want the users adding any additional customers.

# Remove the code in Views/Home/Index.cshtml that creates links for users to view details, edit and delete customers. Again, we don’t want the users modifying the data in the database.

# **CHECKPOINT: Run the project again to make sure everything is okay. Do NOT continue until everything above this step is working correctly.**

# Basic Search

## Add a textbox on the view to allow the user to enter the search string for the quick search. The code below adds the textbox for the quick search and three buttons (Search, Advanced Search, and Show All):

@using (Html.BeginForm())

{

<p class="form-group">

Search: @Html.TextBox("SearchString",null,new { @class = "form-control"}) <br />

<input type="submit" value="Search" class="btn btn-default"/>

@Html.ActionLink("Advanced Search", "DetailedSearch",null, new { @class="btn btn-default"})

@Html.ActionLink("Show All", "Index", null, new { @class="btn btn-default"})

</p>

}

## Modify your Index Action method on the HomeController to search for customers with the name specified in the box. You will be replacing the existing code on this method.

### Add a parameter to the method signature for SearchString. This will be a String parameter, so you don’t need to include the ? to make it nullable.

### Add code that puts the total customer count in the ViewBag. (Hint: the List object has a .Count method)

### Create a list of customers called SelectedCustomers that will contain the list of customers to display on the form.

### List<Customer> SelectedCustomers = new List<Customer>();

### Check to see if SearchString is null. If it is, they didn’t search for anything (or it is the first time they opened the page). If SearchString is null, display all the records. If the search string is not null, use LINQ to limit the list to customers whose first or last name contain the requested string

### SelectedCustomers = db.Customers.Where(c => c.FirstName.Contains(SearchString) || c.LastName.Contains(SearchString)).ToList();

### Add the total customer count to the selected records count for the view bag so the results page will show “Showing X of Y Customers”

### Order the records to display sorted by last name, first name and average sales

### SelectedCustomers.OrderBy(c => c.LastName).ThenBy(c=> c.FirstName).ThenBy(c=> c.AverageSale);

### Modify the return statement to send SelectedCustomers to the view

### return View(SelectedCustomers);

## Modify the Index view to display the “Displaying X out of Y Customers” that we added to the ViewBag above.

# **CHECKPOINT: Run the project to make sure search is working. Make up a few sample cases to see if you are getting the correct results. Do NOT continue until everything above this step is working correctly.**

# Advanced Search:

## Add a new method to your Home Controller called DetailedSearch.

## Make a new view called “DetailedSearch.cshtml” by right-clicking on the method signature in the controller and selecting “Add View.” This view should be empty with no model. This view should be in your Views 🡪Home folder.

## Add in a statement to create the form that will capture the search criteria to include in the HTTP Get request. (See lines 42-51 in the search handout for help)

## Add controls to the view to capture the selected search criteria. You will need:

### Textbox for name. This textbox will search both first and last names.

### Drop-down list for frequency

### Radio buttons for gender

### Textbox for sales amount

### Radio buttons for sort order

## Add an action method called SearchResults on your HomeController to respond to the GET request generated by the DetailedSearch view. This method should send the user back to the Index page with a filtered list of customers.

### The method signature will need to include parameters for each of the search criteria. Make sure each parameter matches the data type of the control that sends it. See the table on line 57 of the search handout for help.

### Follow the directions on lines 58-102 on the Search handout to use LINQ to filter the data set of customers to only display the customers that meet the criteria.

# Test the website with various search criteria to see if it is working

# Publish your website to Azure.

# Zip your VS project file and submit on Canvas. Include a link to the Azure site in the comments.