Case study for *Murach's ASP.NET Core MVC*

The case study described in this document is a series of assignments that let you apply the programming skills you learn in *Murach's ASP.NET Core MVC* by developing a website called SportsPro Technical Support. This website is designed for the technical support department of a hypothetical software company that develops software for sports leagues. The purpose of the website is to track technical support service calls (referred to as *incidents*) in a database that also stores information about the company's customers, software products, and technicians.

An introduction to the case study

This introduction describes the SportsPro Technical Support website and the database that it uses. In addition, it describes a starting point that your instructor may provide as well as some guidelines for completing each assignment.

The website

The SportsPro Technical Support website consists of web pages that support two types of users. First, it lets administrators manage the products, technicians, customers, incidents, and product registrations that are in the database. Second, it lets technicians update incidents that have been assigned to them.

Most assignments have you add one or more new pages to the SportsPro app or improve pages that you added in an earlier assignment. For example, assignment 4-1 has you add pages that let an administrator manage products. If you complete all of the assignments for this book, you will create a relatively realistic web app.

Note that the first number in an assignment refers to the chapter in the book. So, for example, you can complete assignments 4-1 through 4-4 after reading chapter 4. Similarly, you can complete assignment 7-1 after reading chapter 7.

The database

The database for the SportsPro website stores the data that's needed to track technical support incidents. For the completed website, the database should include tables for storing data about the company's products, technicians, customers, incidents, and registrations. If your instructor provides you with a starting point for the case study, you can use that starting point to create the database and populate it with some data. Otherwise, you can use EF Code First development to create or update each table as you complete each assignment. To do that, you may need to refer to the beginning of chapter 12 to master all the skills you need for using EF Code First development to configure and create a database.

The Incidents table contains one row for each technical support incident. Each row in the Incidents table is related to one row in the Customers table, which contains data about the company's customers; one row in the Products table, which contains data about the company's products; and one row in the Technicians table, which contains data about the company's technical support staff. In addition, each row in the Customers table is related to one row in the Countries table, which stores a list of available countries.

The Registrations table, on the other hand, is a linking table that creates a many-to-many relationship between the Customers and Products tables. As a result, each row relates one customer to one product. This allows one customer to register many products, and it allows one product to be registered by many customers.

A starting point

To make it easy to get started, your instructor may provide a version of the SportsPro app that includes some files that will help you get started. These files include the DB context and entity classes that you can use to create the database for the website and fill it with some starting data. If you run the SportsPro app provided by your instructor, it should display a page like the one shown in assignment 3-1.

Most of the assignments within this case study correspond to one of the links on the page shown in assignment 3-1. However, if you click on any of these links, they display an error message that indicates that the page can't be found. That's because you still need to implement these assignments.

If your instructor doesn't provide a version of the SportsPro app to you, you can complete assignment 3-1 to get your website started. Then, in later assignments, you can use EF Code First development to create the tables in the database.

The assignments

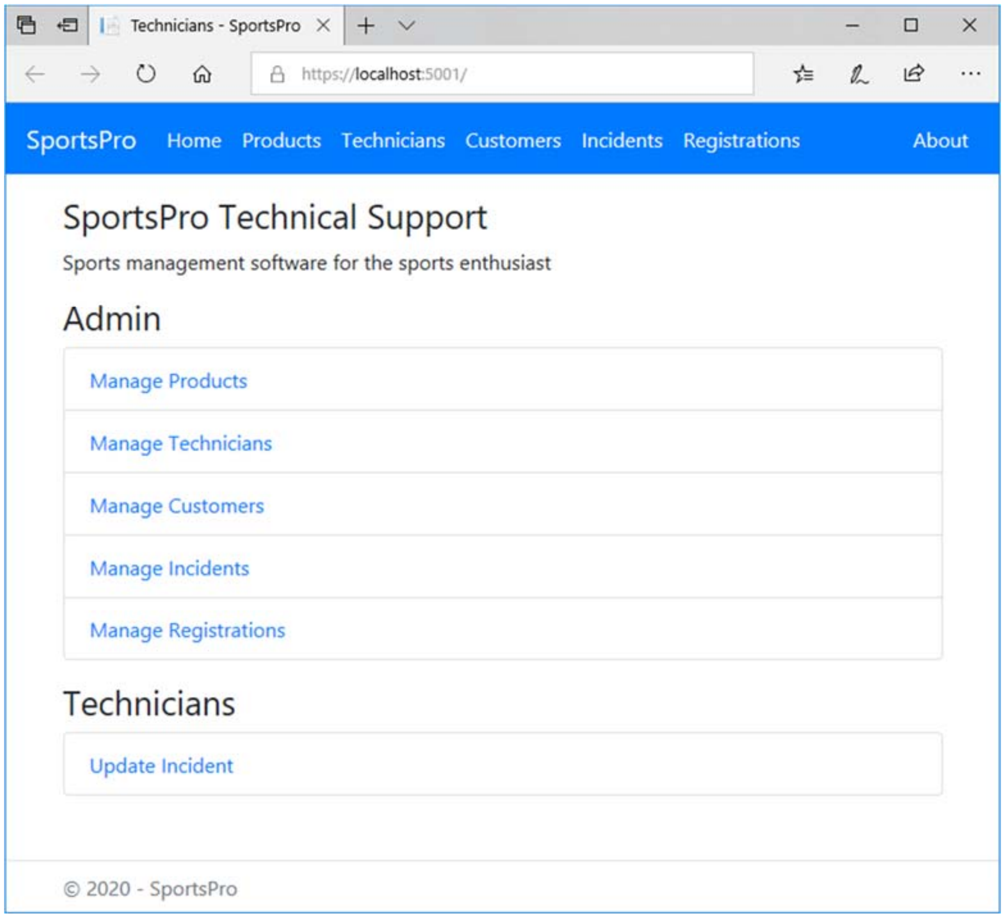
The description of each assignment usually includes one or more screen captures that show how the pages should appear when they're completed as well as specifications for how the assignment should be coded. This information is detailed enough for you to complete each assignment. However, you'll need to use your best judgment on how to code many of the details. To do that, write the code in the way that you think is best, based on the skills that were presented in the book and the guidelines provided to you by your instructor.

Unless you're instructed otherwise, you can implement each assignment using any programming techniques you wish. In some cases, however, the assignment's specifications will direct you to use a specific programming technique. For example, an assignment may direct you to use sessions. In that case, you should implement the assignment as directed.

Assignment 3-1: Set up the Home page

For this assignment, you'll create the Home page for the app. This Home page includes a navigation bar and some menu options for later assignments.

The Home page



SportsPro Technical Support

Sports management software for the sports enthusiast

Admin

Manage Products

Manage Technicians

Manage Customers

Manage Incidents

Manage Registrations

Technicians

Update Incident

© 2020 - SportsPro

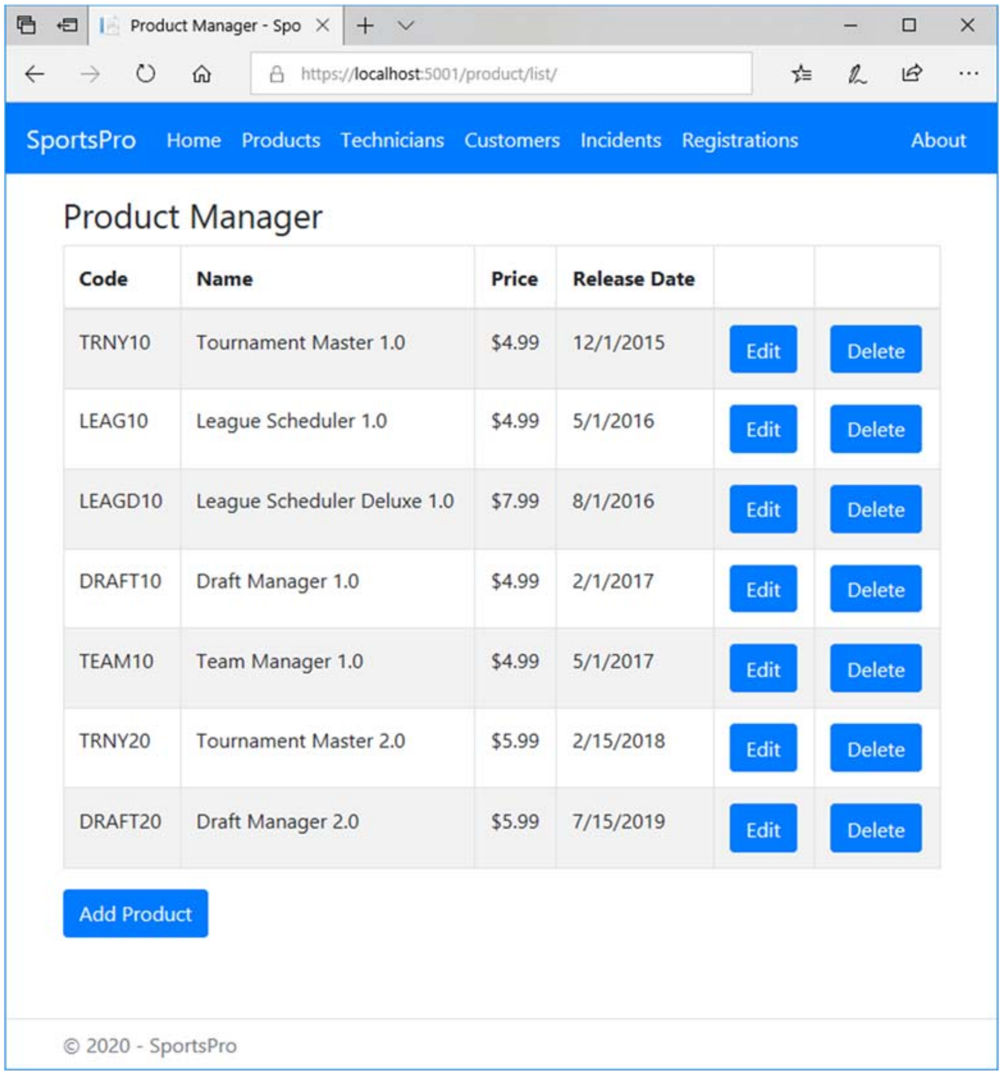
Specifications

- Create an app and that's based on the Model-View-Controller template. Then, edit the code that's provided by that template to create a Home page like the one shown above.
- Use a Razor layout to store elements that are common to all pages, such as the navigation bar and the footer.
- Use Bootstrap classes to format the HTML elements and to create the navigation bar.
- Mark all navigation bar items as active.

Assignment 4-1: Manage products

For this assignment, you'll modify the app so it lets an admin user view, edit, and delete existing products as well as add new products.

The Product Manager page



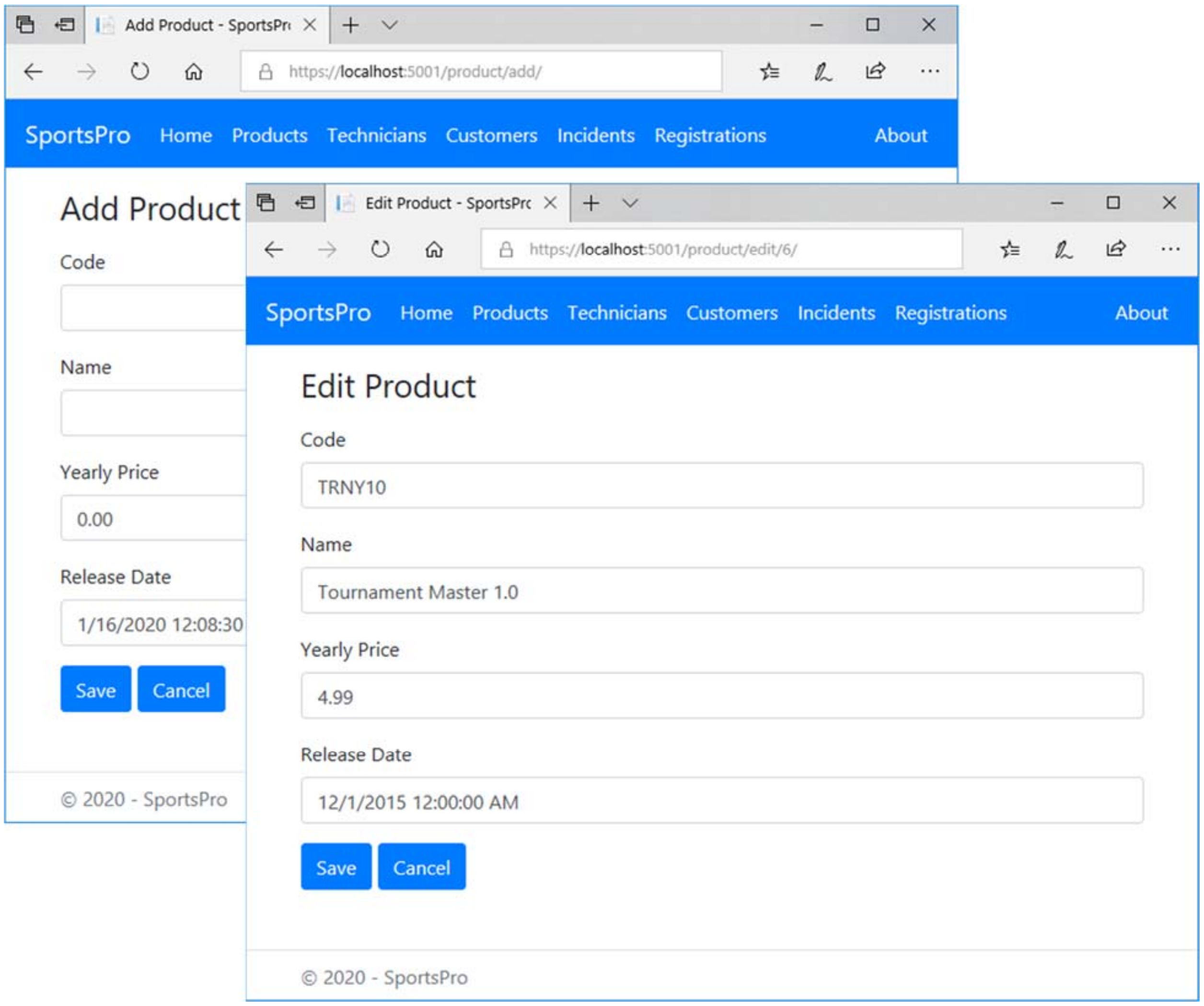
Product Manager

Code	Name	Price	Release Date		
TRNY10	Tournament Master 1.0	$4.99	12/1/2015	Edit	Delete
LEAG10	League Scheduler 1.0	$4.99	5/1/2016	Edit	Delete
LEAGD10	League Scheduler Deluxe 1.0	$7.99	8/1/2016	Edit	Delete
DRAFT10	Draft Manager 1.0	$4.99	2/1/2017	Edit	Delete
TEAM10	Team Manager 1.0	$4.99	5/1/2017	Edit	Delete
TRNY20	Tournament Master 2.0	$5.99	2/15/2018	Edit	Delete
DRAFT20	Draft Manager 2.0	$5.99	7/15/2019	Edit	Delete

Specifications

- When the user clicks the Manage Products link on the Home page or the Products link in the navbar, the app should display the Product Manager page.
- When the user clicks the Add Product button, the app should display the Add/Edit Product page with blank Code and Name fields but with the current date in the Release Date field.
- When the user clicks the Edit button for a product, the app displays the Add/Edit Product page with the current data for the product.
- When the user clicks the Delete button for a product, the app should display a Delete Product page that confirms the deletion.

The Add/Edit Product page



Add Product

Code

Name

Yearly Price

0.00

Release Date

1/16/2020 12:08:30

Edit Product

Code

TRNY10

Name

Tournament Master 1.0

Yearly Price

4.99

Release Date

12/1/2015 12:00:00 AM

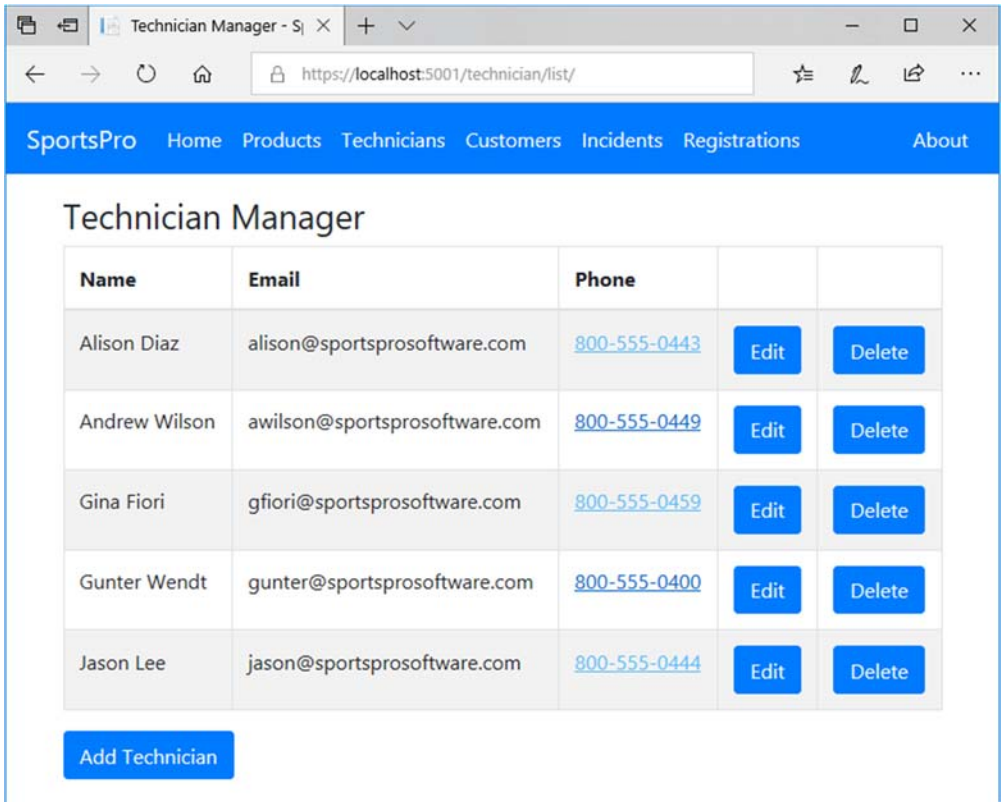
Specifications (continued)

- The app should display the Product Manager page when the user completes or cancels an add, edit, or delete operation.
- Validate the data the user enters in the Add/Edit Product page to be sure that the user enters a product code, name, and price. If this data isn't provided, the app should display a summary of the validation errors above the form.
- Use lowercase URLs with a trailing slash.
- Use the same Razor view file to add and edit a product.
- If you have trouble adding migrations or updating the database, refer to the EF PowerShell commands described in chapters 4 and 12.

Assignment 4-2: Manage technicians

For this assignment, you'll modify the app so it lets an admin user view, edit, and delete existing technicians as well as add new technicians.

The Technician Manager page



Technician Manager

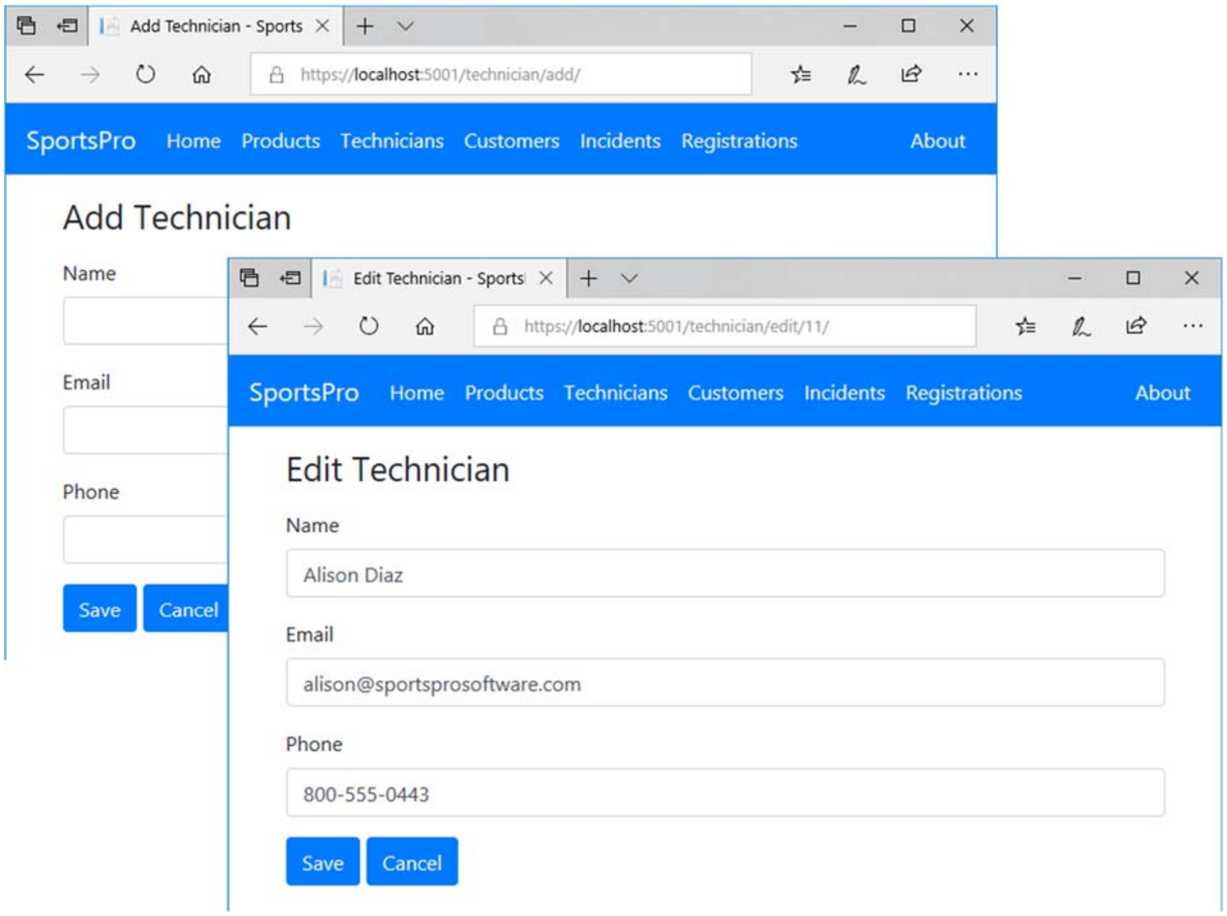
Name	Email	Phone		
Alison Diaz	alison@sportsprosoftware.com	800-555-0443	Edit	Delete
Andrew Wilson	awilson@sportsprosoftware.com	800-555-0449	Edit	Delete
Gina Fiori	gfiori@sportsprosoftware.com	800-555-0459	Edit	Delete
Gunter Wendt	gunter@sportsprosoftware.com	800-555-0400	Edit	Delete
Jason Lee	jason@sportsprosoftware.com	800-555-0444	Edit	Delete

Add Technician

Specifications

- When the user clicks the Manage Technicians link on the Home page or the Technicians link in the navbar, the app should display the Technician Manager page.
- When the user clicks the Add Technician button, the app should display the Add/Edit Technician page with blank fields.
- When the user clicks the Edit button for a product, the app should display the Add/Edit Technician page with the current data for the technician.
- When the user clicks the Delete button for a technician, the app should display a Delete Technician page that confirms the deletion.

The Add/Edit Technician page


Add Technician - Sports
https://localhost:5001/technician/add/
SportsPro Home Products Technicians Customers Incidents Registrations About
Save
Cancel
Edit Technician - Sports
https://localhost:5001/technician/edit/11/
SportsPro Home Products Technicians Customers Incidents Registrations About
Save
Cancel

Add Technician

Name

Email

Phone

Edit Technician

Name

Alison Diaz

Email

alison@sportsprosoftware.com

Phone

800-555-0443

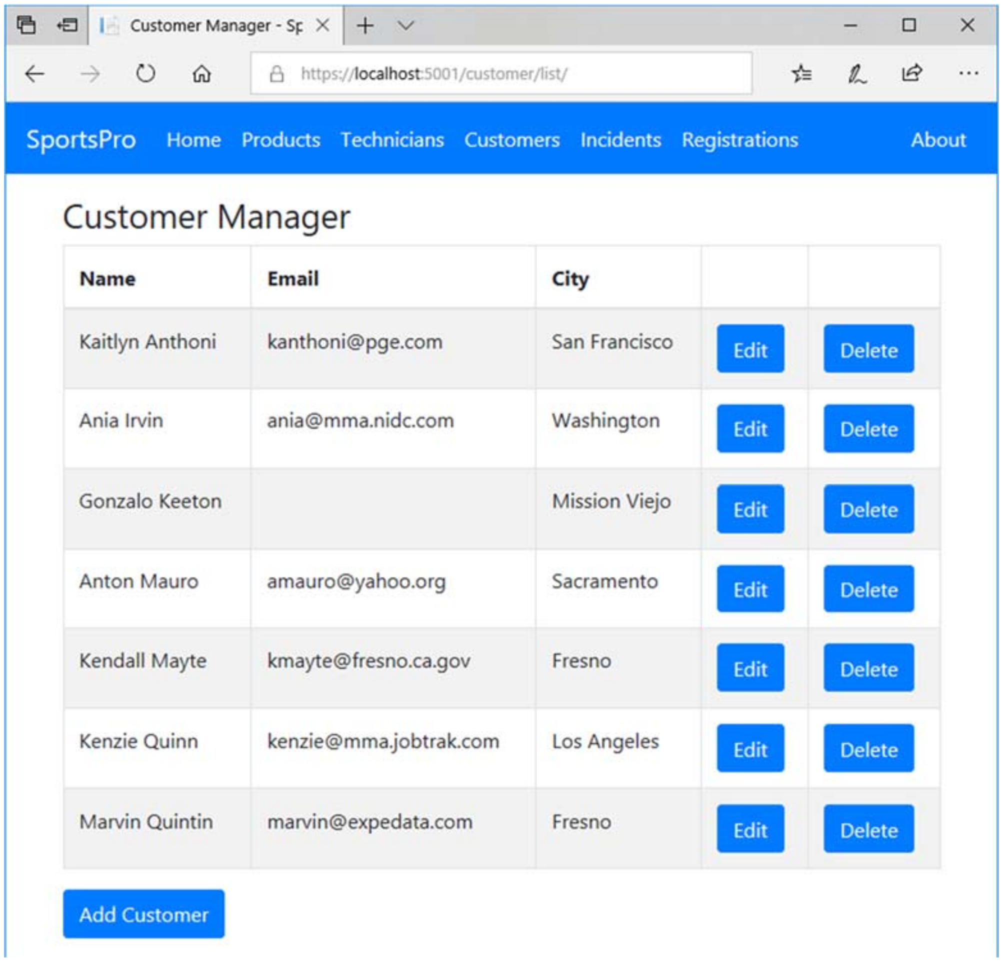
Specifications (continued)

- The app should display the Technician Manager page when the user completes or cancels an operation that adds, edits, or deletes a technician.
- Validate the data the user enters in the Add/Edit Technician page to be sure that the user enters values for all fields. If this data isn't provided, the app should display a summary of the validation errors above the form.
- Use lowercase URLs with a trailing slash.
- Use the same Razor view file to add and edit a technician.

Assignment 4-3: Manage customers

For this assignment, you'll modify the app so it lets an admin user view, edit, and delete existing customers as well as add new customers.

The Customer Manager page



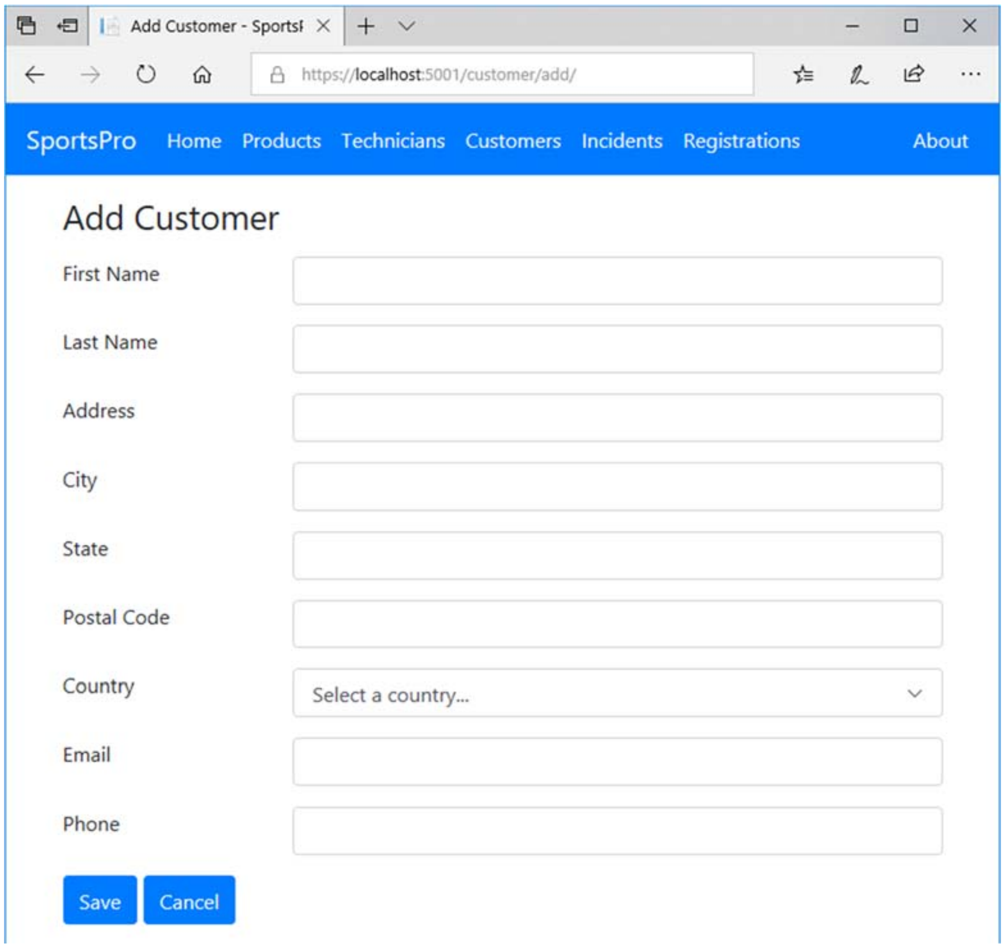
Customer Manager

Name	Email	City		
Kaitlyn Anthoni	kanthoni@pge.com	San Francisco	Edit	Delete
Ania Irvin	ania@mma.nidc.com	Washington	Edit	Delete
Gonzalo Keeton		Mission Viejo	Edit	Delete
Anton Mauro	amauro@yahoo.org	Sacramento	Edit	Delete
Kendall Mayte	kmayte@fresno.ca.gov	Fresno	Edit	Delete
Kenzie Quinn	kenzie@mma.jobtrak.com	Los Angeles	Edit	Delete
Marvin Quintin	marvin@expedata.com	Fresno	Edit	Delete

Specifications

- When the user clicks the Manage Customers link from the Home page or the Customers link from the navbar, the app should display the Customer Manager page.
- When the user clicks the Add Customer button, the app should display the Add/Edit Customer page with blank fields.
- When the user clicks the Edit button for a customer, the app displays the Add/Edit Customer page with the data for the selected customer.
- When the user clicks the Delete button for a customer, the app should display a Delete Customer page that confirms the deletion.

The Add/Edit Customer page



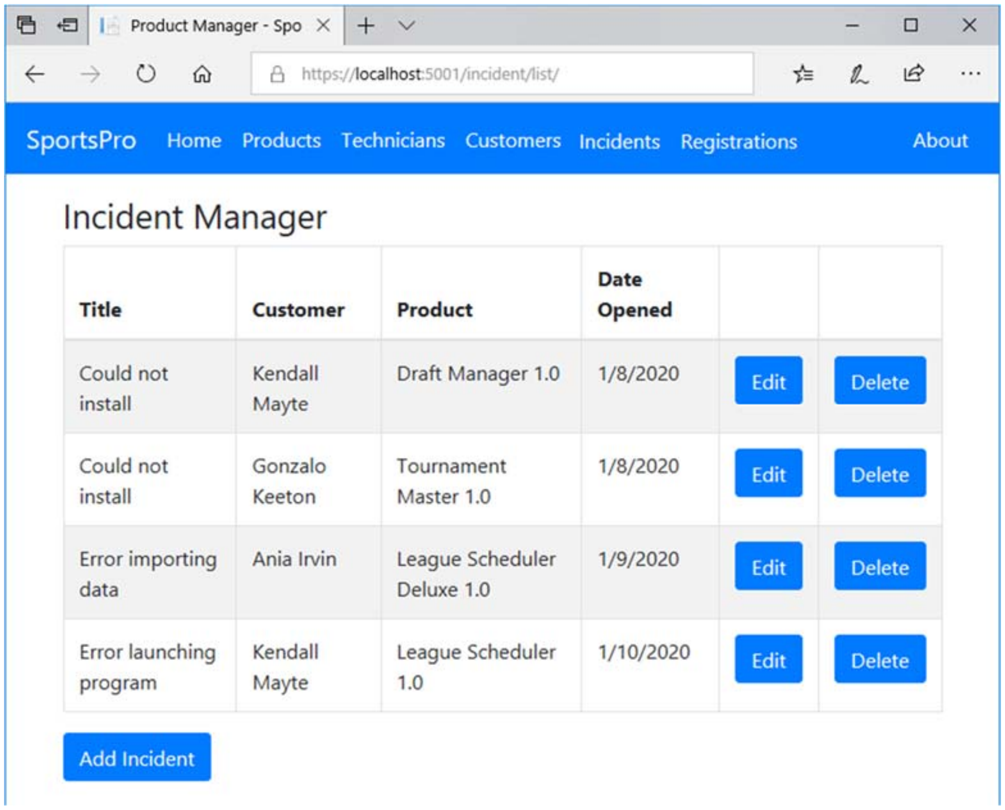
Specifications (continued)

- The app should display the Customer Manager page when the user completes or cancels an operation for adding, editing, or deleting a customer.
- Validate the data the user enters in the Add/Edit Customer page. To be valid…
    - The user must select a country from the drop-down list of countries.
    - The Email and Phone fields are optional.
    - All other fields are required.
- Use the same Razor view file to add and edit a customer.
- In the Country drop-down list, display all countries that are available from the database. If necessary, add some countries to the database.
- When the app displays the Edit Customer page, make sure to select the correct country for the specified customer.

Assignment 4-4: Manage incidents

For this assignment, you'll modify the app so it lets an admin user view, edit, and delete existing incidents as well as add new incidents.

The Incident Manager page



Incident Manager

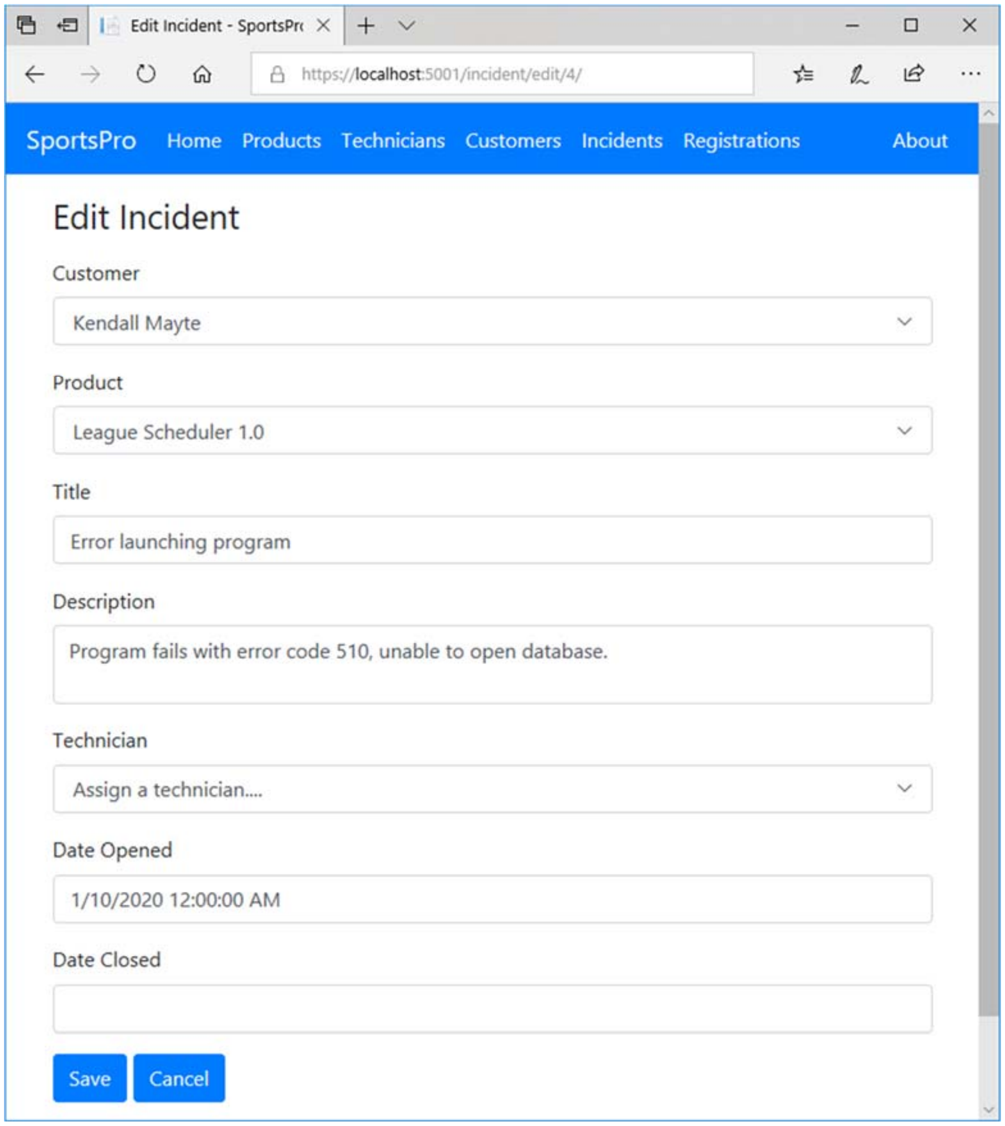
Title	Customer	Product	Date Opened		
Could not install	Kendall Mayte	Draft Manager 1.0	1/8/2020	Edit	Delete
Could not install	Gonzalo Keeton	Tournament Master 1.0	1/8/2020	Edit	Delete
Error importing data	Ania Irvin	League Scheduler Deluxe 1.0	1/9/2020	Edit	Delete
Error launching program	Kendall Mayte	League Scheduler 1.0	1/10/2020	Edit	Delete

Add Incident

Specifications (continued)

- When the user clicks the Manage Incidents link from the Home page or the Incidents link from the navbar, the app should display the Incident Manager page.
- When the user clicks the Add Incident button, the app should display the Add/Edit Incident page with blank fields.
- When the user clicks the Edit button for an incident, the app should display the Add/Edit Incident page with the data for the selected incident.
- When the user clicks the Delete button for an incident, the app should display a Delete Incident page that confirms the deletion.

The Add/Edit Incident page



Edit Incident

Customer: Kendall Mayte

Product: League Scheduler 1.0

Title: Error launching program

Description: Program fails with error code 510, unable to open database.

Technician: Assign a technician....

Date Opened: 1/10/2020 12:00:00 AM

Date Closed:

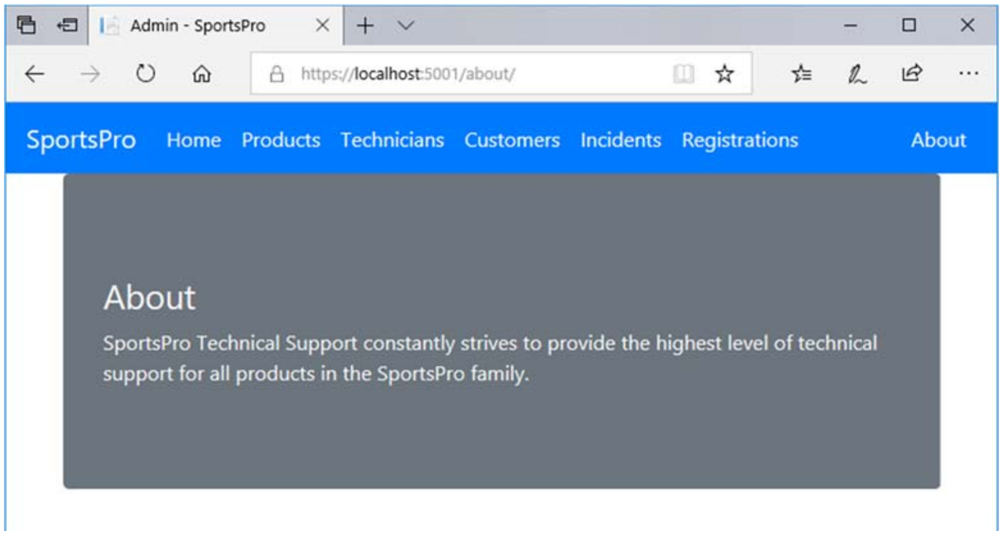
Specifications (continued)

- The app should display the Incident Manager page when the user completes or cancels an operation for adding, editing, or deleting an incident.
- Validate the data the user enters in the Add/Edit Customer page. To be valid…
    - The user must select a customer and a product from the drop-down lists.
    - The Technician and Data Opened fields are optional.
    - All other fields are required.
- Use the same Razor view file to add and edit a customer.
- To make the Technician field optional, make sure to specify a nullable int type (?int) for the TechnicianID property of the Incident entity class.

Assignment 6-1: Improve the URLs for the app

For this assignment, you'll modify the SportsPro website so it uses some skills described in chapter 6.

The About page



About

SportsPro Technical Support constantly strives to provide the highest level of technical support for all products in the SportsPro family.

Specifications

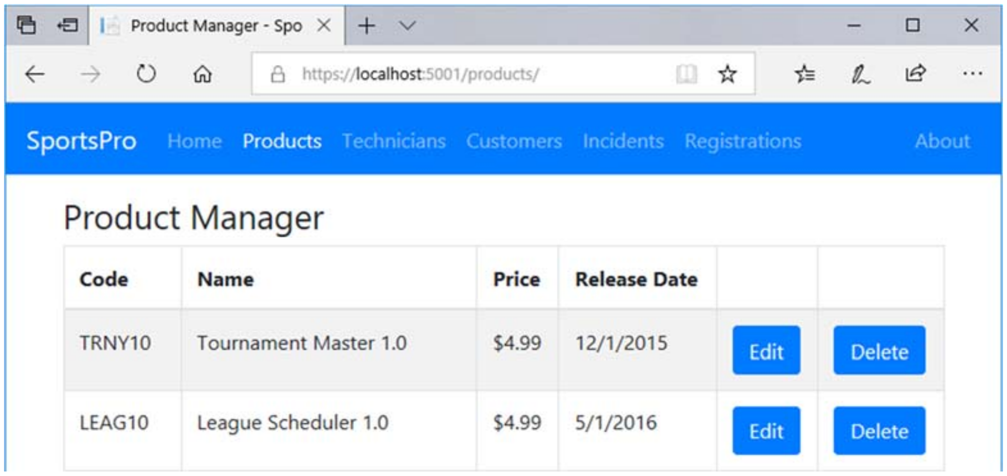
- Add an About page like the one shown above that's displayed by the About action of the Home controller.
- Use attribute routing to shorten some of the URLs for the app like this:

Request URL	Controller/Action
`/about`	`Home/About`
`/products`	`Product/List`
`/technicians`	`Technician/List`
`/customers`	`Customer/List`
`/incidents`	`Incident/List`

Assignment 7-1: Improve the Razor views

For this assignment, you'll modify the SportsPro website so it uses some skills described in chapter 7.

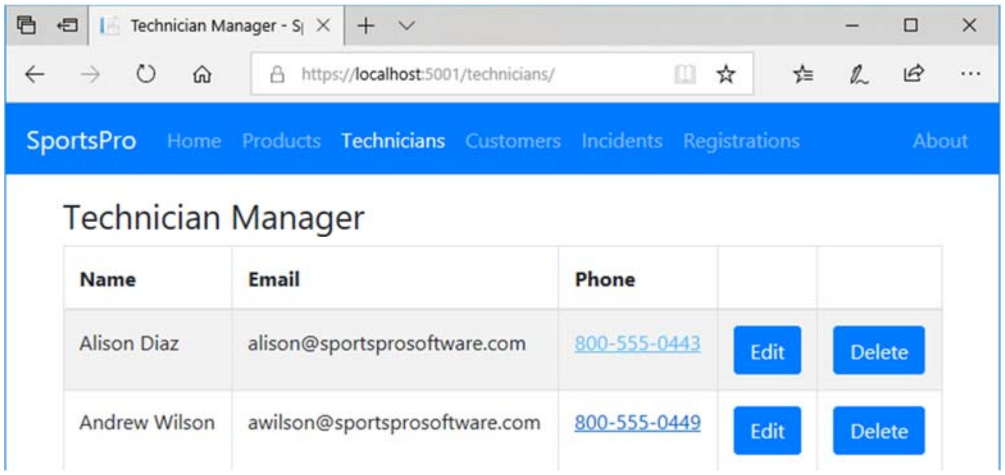
The navigation bar when the Product Manager page is displayed



Product Manager

Code	Name	Price	Release Date		
TRNY10	Tournament Master 1.0	$4.99	12/1/2015	Edit	Delete
LEAG10	League Scheduler 1.0	$4.99	5/1/2016	Edit	Delete

The navigation bar when the Technician Manager page is displayed



Technician Manager

Name	Email	Phone		
Alison Diaz	alison@sportsprosoftware.com	800-555-0443	Edit	Delete
Andrew Wilson	awilson@sportsprosoftware.com	800-555-0449	Edit	Delete

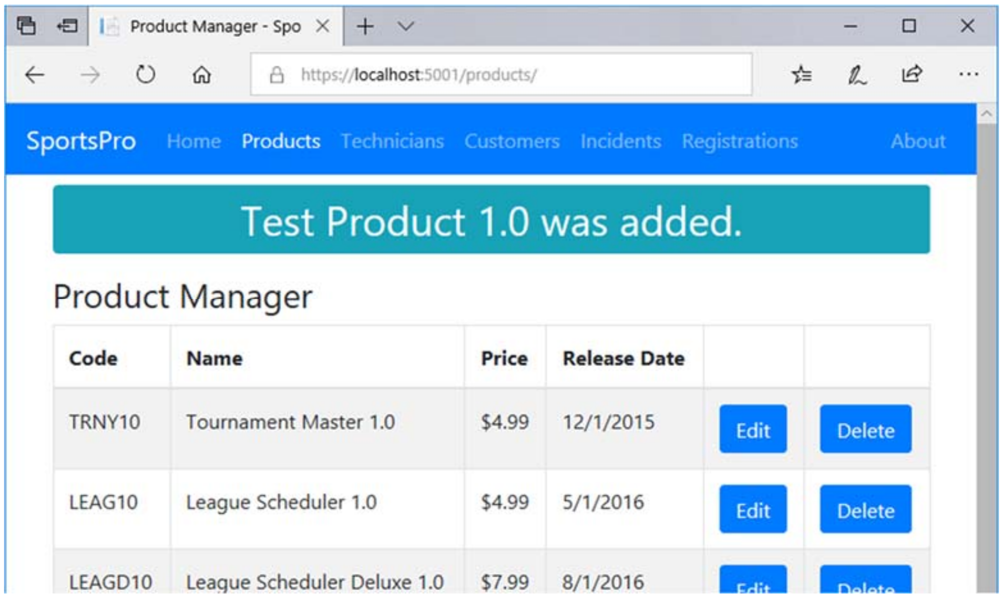
Specifications

- Get the navigation links to work correctly. For example, the Products link should be active for the Product controller, the Technicians link should be active for the Technicians controller, and so on.
- Convert the Razor foreach loops that display the items for the <select> elements so they use the asp-items tag helper instead. To include an extra item as the first item in the drop-down list, you can still include an <option> element like this one:
  `<option value="">Select a product...</option>`
- Remove the jQuery libraries for validation from the layout for the app and only include them in views where they're necessary such as the views for the Add/Edit pages.

Assignment 8-1: Use TempData to display messages

For this assignment, you'll modify the SportsPro website so it uses TempData to display messages when an operation is successful.

The Product Manager page after a new product has been added



Test Product 1.0 was added.

Product Manager

Code	Name	Price	Release Date		
TRNY10	Tournament Master 1.0	$4.99	12/1/2015	Edit	Delete
LEAG10	League Scheduler 1.0	$4.99	5/1/2016	Edit	Delete
LEAGD10	League Scheduler Deluxe 1.0	$7.99	8/1/2016	Edit	Delete

Specifications

- In the Product controller, edit the return types of the action methods so an action method that returns a view specifies ViewResult as the return type and an action method that redirects specifies RedirectToActionResult as the return type.
- In the Product controller, use TempData to store a success message after each successful add, edit, or delete operation. Then, display the message on the Product Manager page. To do that, you can add some code to the layout that displays a TempData message across the top of the page if the message exists.

Assignment 8-2: Use a view model with the Incidents Manager

For this assignment, you'll modify the SportsPro website so it uses a view model to pass data to the List Manager page and its Add/Edit Incident page.

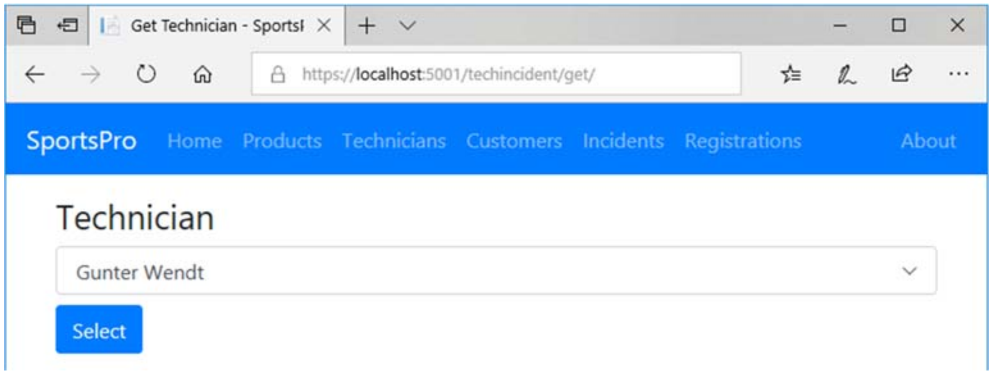
Specifications

- Use a view model to pass data to the Incident Manager page. This view model should store a list of incidents and a string that specifies the filtering for the page.
- Use a view model to pass data to the Add/Edit Incident page. This view model should store a list of customers, a list of products, a list of technicians, the current incident, and a string that specifies whether the page is for an Add or Edit operation.

Assignment 9-1: Update incidents

For this assignment, you'll add some pages that let technicians view all of their assigned and open incidents. Then, the technician can edit an incident's description and specify a close date for the incident.

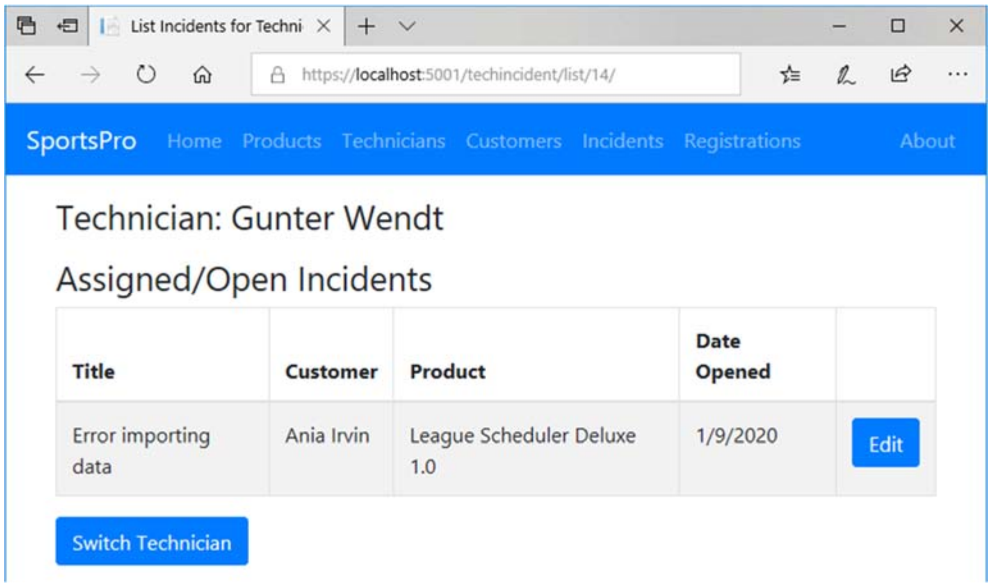
The Get Technician page



Technician

Gunter Wendt

The Incidents by Technician page



Technician: Gunter Wendt

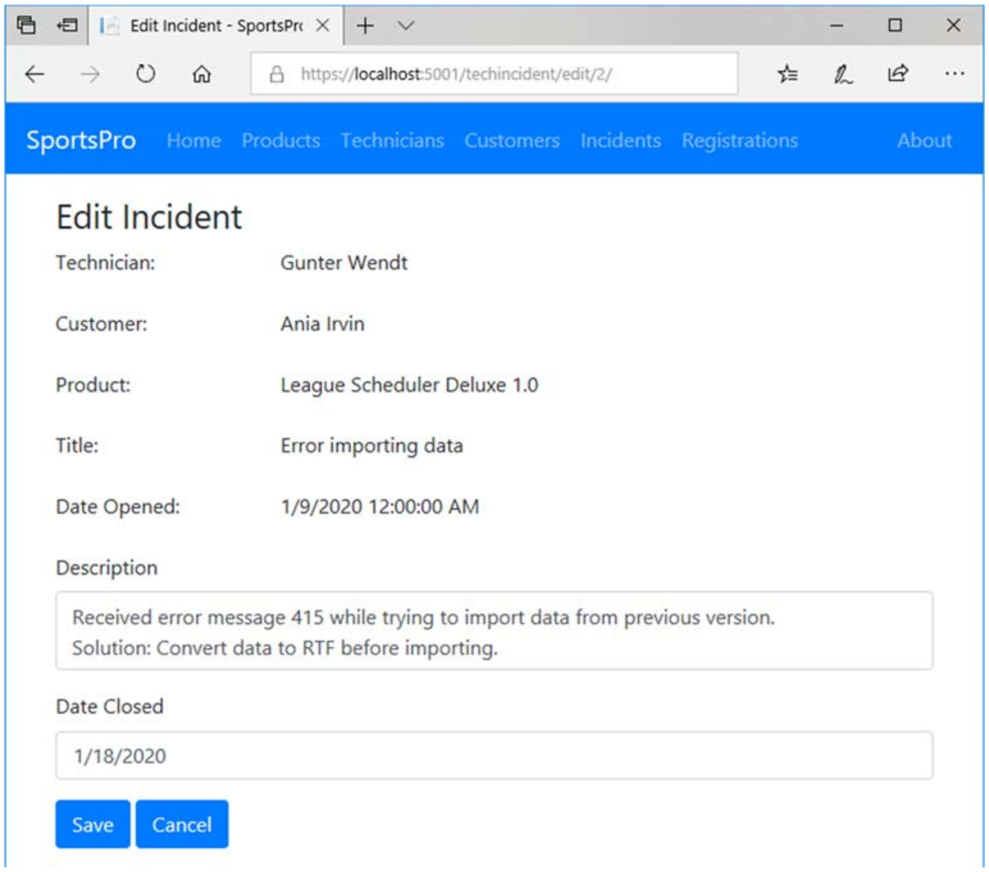
Assigned/Open Incidents

Title	Customer	Product	Date Opened	
Error importing data	Ania Irvin	League Scheduler Deluxe 1.0	1/9/2020	Edit

Specifications

- The Get Technician page is only displayed by the Update Incident link on the Home page for the website. There's no link for it in the navigation bar.
- The Get Technician page allows the user to select the technician name. If the user doesn't select a name, the app should display a message indicating that the user must select a technician.
- The Incidents by Technician page displays all open incidents that have been assigned to the selected technician. Or, if there are no open incidents for the current technician, this page displays a message that indicates that there are no open incidents.
- To switch technicians, the user can click on the Switch button. This displays the Get Technician page again.

The Edit Incident page



Edit Incident

Technician: Gunter Wendt

Customer: Ania Irvin

Product: League Scheduler Deluxe 1.0

Title: Error importing data

Date Opened: 1/9/2020 12:00:00 AM

Description

Received error message 415 while trying to import data from previous version.
Solution: Convert data to RTF before importing.

Date Closed

1/18/2020

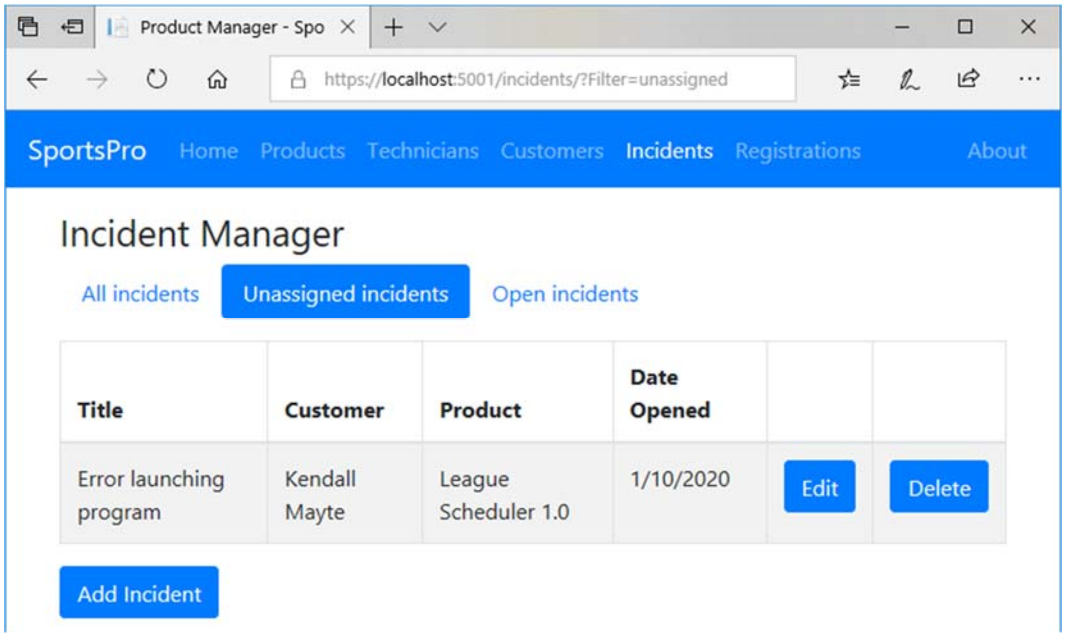
Specifications (continued)

- From the Incidents by Technician page, the user can click the Edit button for an incident to update that incident. This displays the Edit Incident page.
- Then Edit Incident page allows the technician to modify the description and optionally enter the date the incident was closed and click on the Save button.
- If the technician successfully updates an incident or cancels the update operation, the app should display the Incidents by Technician list again.
- The website should store the technician's ID in session state. That way, the website can "remember" the current technician.

Assignment 10-1: Add filtering to a page

For this assignment, you'll modify the SportsPro website so it allows an admin user to filter the incidents to view all incidents, unassigned incidents, or open incidents.

The Incident Manager page with filtering



Incident Manager

Title	Customer	Product	Date Opened		
Error launching program	Kendall Mayte	League Scheduler 1.0	1/10/2020	Edit	Delete

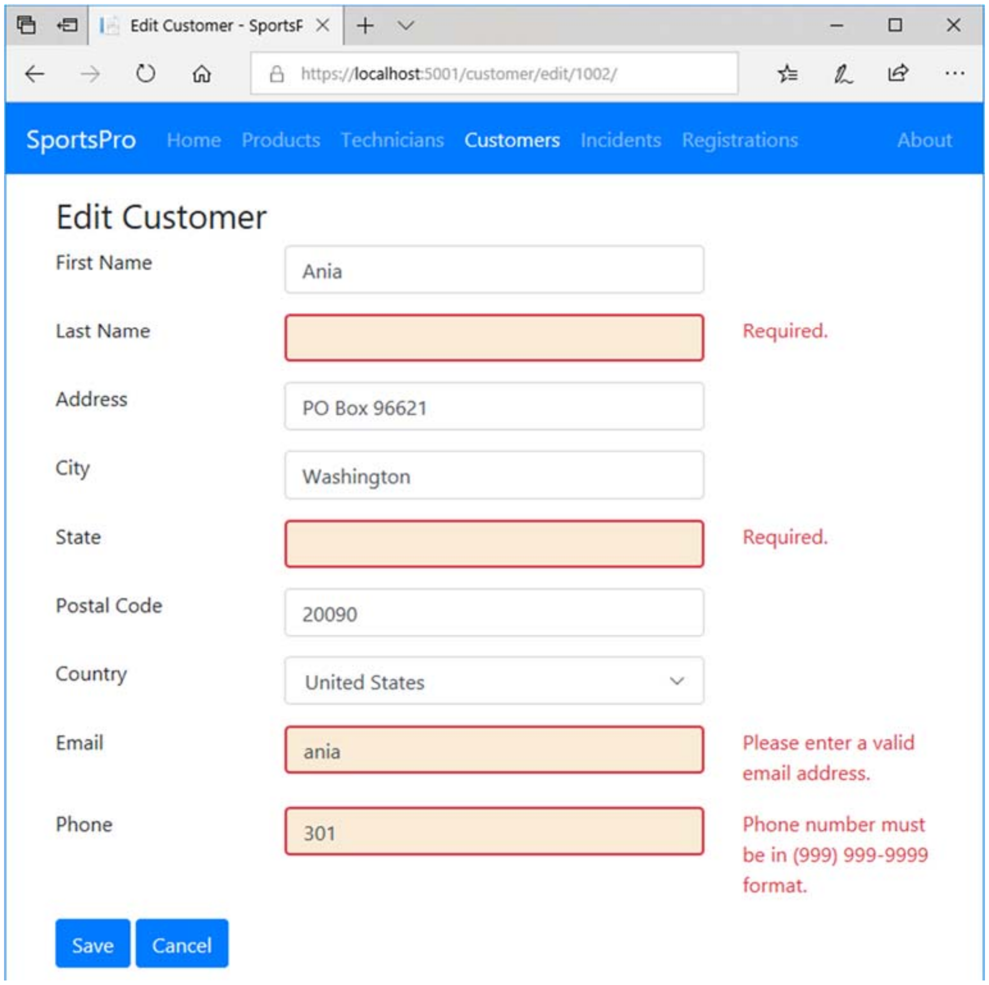
Specifications

- The Incident Manager page should include three links styled as Bootstrap pills that allow the user to filter the incidents by only displaying unassigned incidents (TechnicianID is null) or only displaying open incidents (DateClosed is null).

Assignment 11-1: Improve validation

For this assignment, you'll modify the SportsPro website so it improves the data validation for the Add/Edit Customer page.

The Edit Customer page



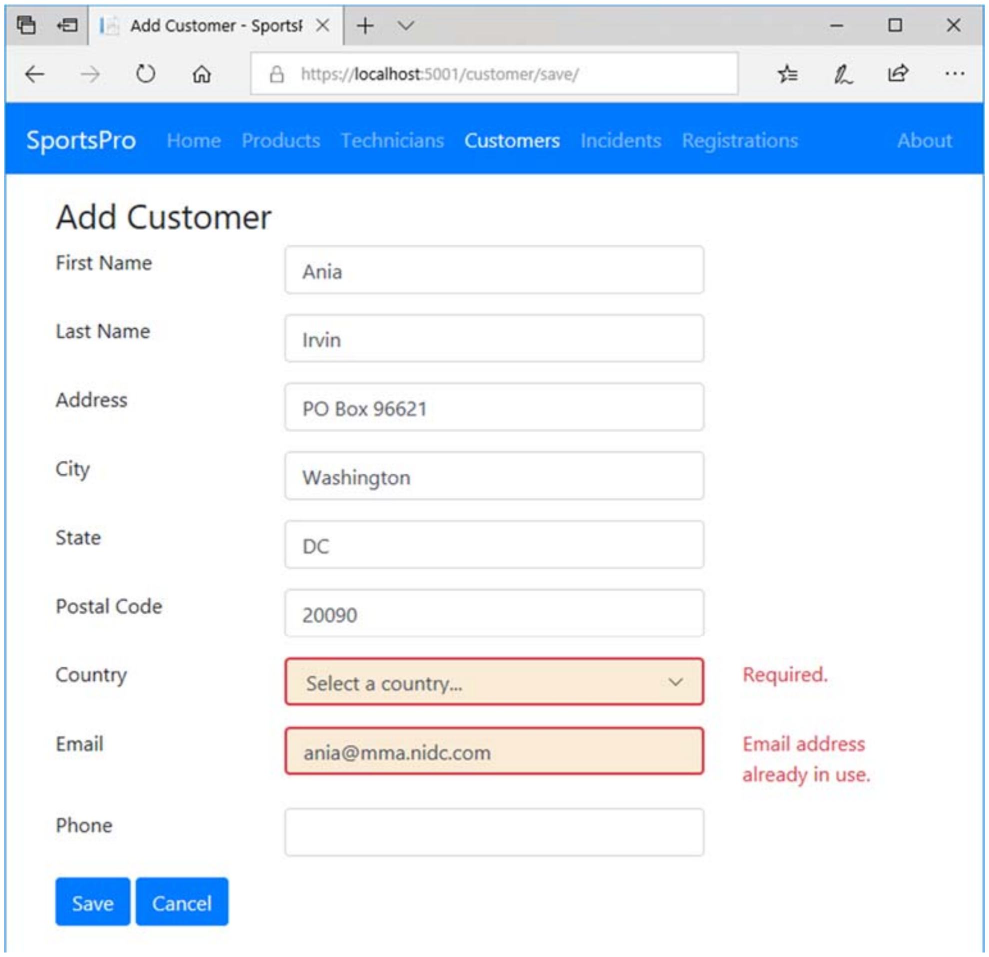
Edit Customer

First Name	Ania	
Last Name		Required.
Address	PO Box 96621	
City	Washington	
State		Required.
Postal Code	20090	
Country	United States	
Email	ania	Please enter a valid email address.
Phone	301	Phone number must be in (999) 999-9999 format.

Specifications

- Same as assignment 4-3 but with improved data validation for some fields on the Add/Edit Customer page.
- Here are the data validation requirements...
    - The first name, last name, address, city, state, and email are required and must have at least 1 and less than 51 characters.
    - The postal code is required and must have at least 1 and less than 21 characters.
    - The phone number is not required, but if the user enters one, it must be in the "(999) 999-9999" format. (You can search the Internet for an appropriate regular expression.)
    - The email address must be a valid email address. To do that, you can use the DataType validation attribute:
      `[DataType(DataType.EmailAddress)]`

The Add Customer page



Add Customer

First Name	Ania	
Last Name	Irvin	
Address	PO Box 96621	
City	Washington	
State	DC	
Postal Code	20090	
Country	Select a country...	Required.
Email	ania@mma.nidc.com	Email address already in use.
Phone		

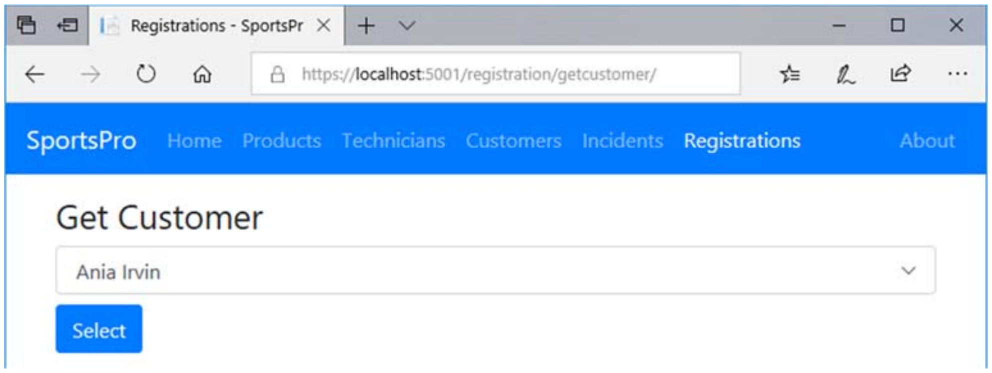
Specifications (continued)

- More data validation requirements...
    - The drop-down list must be set to a valid country (not the "Select a country" item).
    - When adding a new customer, the email address must not already be used by another customer.
- Add custom CSS that uses a different border and background for the <input> elements that contain invalid data.

Assignment 12-1: Manage registrations

For this assignment, you'll add some pages that let an admin user view the product registrations for a customer and add new product registrations as well.

The Get Customer page



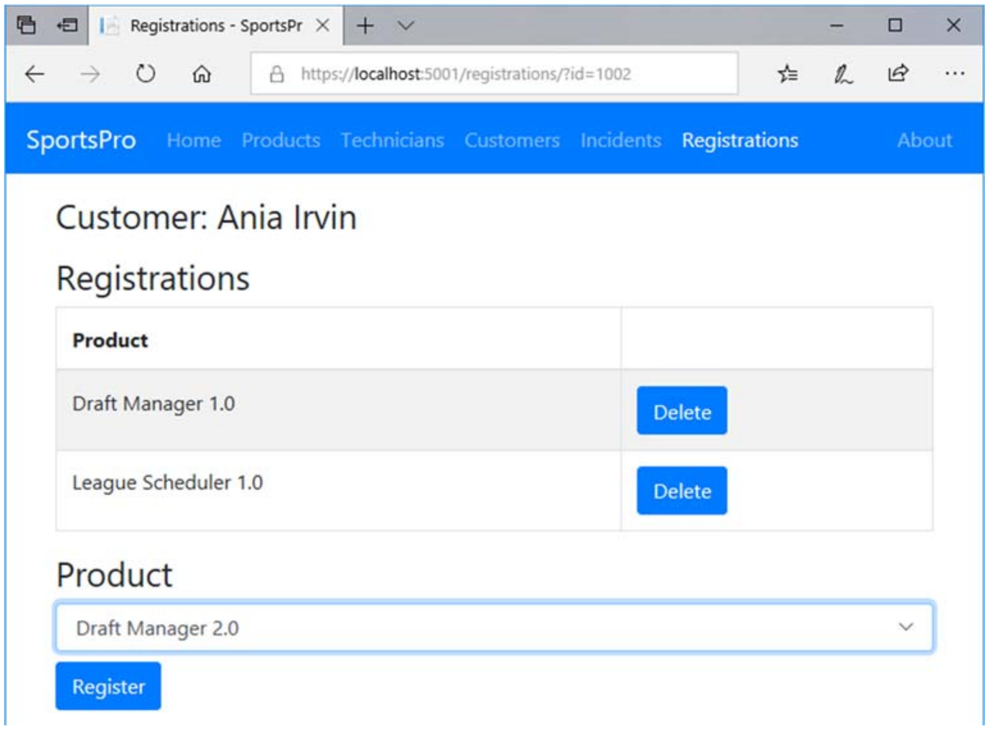
Get Customer

Ania Irvin

Specifications

- The Get Customer page is displayed by the Manage Registrations link on the Home page and by the Registrations link in the navigation bar.
- The Get Customer page allows the user to select the customer. If the user doesn't select a customer, the app should display a message indicating that the user must select a customer.

The Registrations page



Customer: Ania Irvin

Registrations

Product	
Draft Manager 1.0	Delete
League Scheduler 1.0	Delete

Product

Draft Manager 2.0

Specifications (continued)

- The Registrations page displays the name of the selected customer and all products that have been registered for that customer. Or, if there are no products registered for the selected customer, this page indicates that there are no products registered for the selected user.
- To register a product, the user can select a product from the Product drop-down list and click the Register button. This should add the product to the table of registered products for the selected customer.
- The website should store the customer's ID in session state. That way, the website can "remember" the current customer.
- To get these pages to work correctly, you should add a linking entity class named Registration that provides a many-to-many relationship between the Customer and Product entities.

Assignment 12-2: Encapsulate the data layer

For this assignment, you'll modify the SportsPro website so it uses the repository pattern and the unit of work pattern to encapsulate the data layer.

Specifications

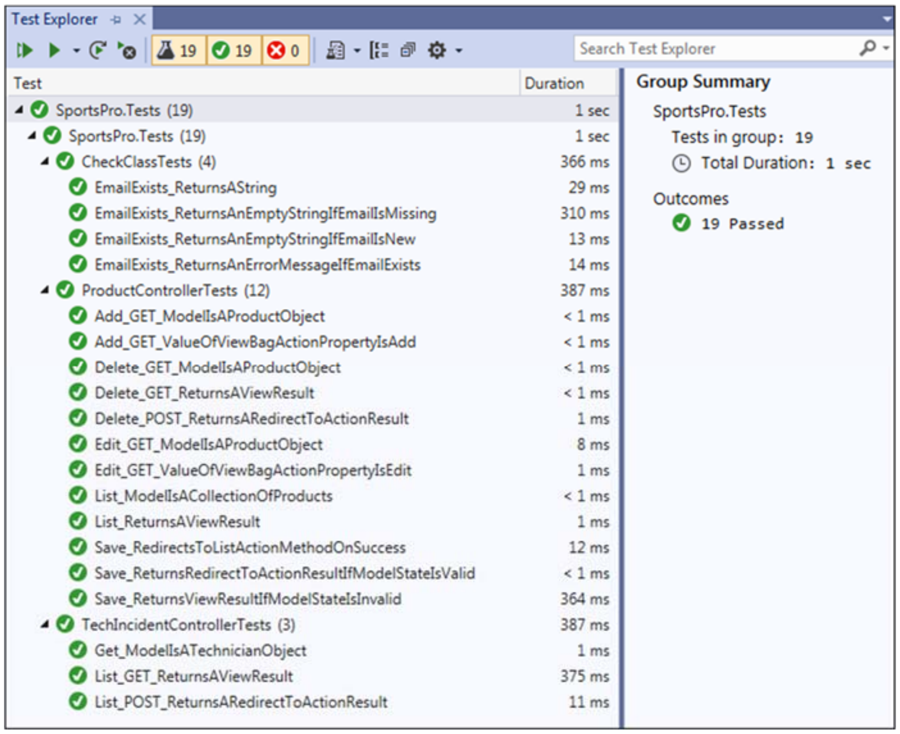
- All the code in the data layer should be stored it in its own folder (not in the Models, Views, or Controllers folder).
- The code that adds seed data should be moved out of the DB context class and into separate classes (one for each table).
- The code that configures the many-to-many relationship with the Registrations table as the linking table should be moved out of the DB context class and into a separate class.
- The data layer should use the IRepository<T> interface and the Repository<T> and QueryOptions<T> classes to implement the repository pattern.

  NOTE: Because one of the queries in the TechIncident controller has multiple WHERE clauses, you should use the Repository<T> and QueryOptions<T> classes presented in chapter 13. That's because these classes provide for multiple WHERE clauses, while the classes presented in chapter 12 only allow for provide for a single WHERE clause.
- The data layer should use an interface and a class to implement the unit of work pattern.
- The controllers should use the repository and unit of work classes instead of the DB context class.
- Controllers that only work with one DbSet can use a repository class. For instance, the Product controller only needs to work with Product objects, so it can use the Repository<Product> class.
- Controllers that work with multiple DbSets should use the unit of work class. For instance, the Incident controller works with Incident, Customer, Product, and Technician objects, so it should use the unit of work class.
- For the Customer controller to be able to use a repository or unit of work class, the Validation controller and the Check class must be updated, too.
- The TechIncident controller has a query with more than one WHERE clause. You can refer to the text in figure 13-4 for an example of how this query should look after it's updated to use the WhereClauses property of the QueryOptions class.

Assignment 14-1: Automate testing

For this assignment, you will automate some of the testing for the app. In order to facilitate that, you'll update the controllers to use dependency injection (DI).

The Test Explorer window



Test	Duration
SportsPro.Tests (19)	1 sec
SportsPro.Tests (19)	1 sec
CheckClassTests (4)	366 ms
EmailExists_ReturnsAString	29 ms
EmailExists_ReturnsAnEmptyStringIfEmailIsMissing	310 ms
EmailExists_ReturnsAnEmptyStringIfEmailIsNew	13 ms
EmailExists_ReturnsAnErrorMessageIfEmailExists	14 ms
ProductControllerTests (12)	387 ms
Add_GET_ModelIsAProductObject	< 1 ms
Add_GET_ValueOfViewBagActionPropertyIsAdd	< 1 ms
Delete_GET_ModelIsAProductObject	< 1 ms
Delete_GET_ReturnsAViewResult	< 1 ms
Delete_POST_ReturnsARedirectToActionResult	1 ms
Edit_GET_ModelIsAProductObject	8 ms
Edit_GET_ValueOfViewBagActionPropertyIsEdit	1 ms
List_ModelIsACollectionOfProducts	< 1 ms
List_ReturnsAViewResult	1 ms
Save_RedirectsToListActionMethodOnSuccess	12 ms
Save_ReturnsRedirectToActionResultIfModelStateIsValid	< 1 ms
Save_ReturnsViewResultIfModelStateIsInvalid	364 ms
TechIncidentControllerTests (3)	387 ms
Get_ModelIsATechnicianObject	1 ms
List_GET_ReturnsAViewResult	375 ms
List_POST_ReturnsARedirectToActionResult	11 ms

Group Summary
SportsPro.Tests
Tests in group: 19
Total Duration: 1 sec

Outcomes
19 Passed

Specifications

- The controllers should receive repository or unit of work objects by dependency injection.
- The dependencies should be configured with the transient life cycle.
- To facilitate testing, the controllers should work with interfaces, not concrete objects. In addition, the return types in the unit of work interface and class should be interfaces.
- The TechIncident controller should receive an HttpContextAccessor object by DI, in addition to a unit of work object. Then, it should access data in session using the ISession object from that HttpContextAccessor object, rather than from the controller's HttpContext property. This makes the TechIncident controller easier to test.
- The Validation controller should use action method injection.
- The solution should have a unit test project that contains tests for the controllers of the main web app. The unit test project should also contain tests for the static Check class.
- The unit test project should use Moq to create fake versions of the repository, unit of work, and HttpContextAccessor objects the controllers and/or model classes receive via DI.

Assignment 15-1: Use tag helpers, partial views, and view components

For this assignment, you will use tag helpers, partial views, and view components to reduce code duplication in your Razor views and layouts, and to simplify your controller and view model code.

Specifications

- Use a partial view to store the jQuery libraries for data validation.
- Use a tag helper to display a temporary message. The layout should no longer use Razor code to do this.
- Use a view component to display the copyright info. The layout should no longer use Razor code to do this.
- Use a tag helper to mark a current nav link as active. This should work for nav links in the navbar in the layout and for the nav pills for filtering incidents. The nav links should no longer use any Razor code to do this.
- Use view components for the options of the <select> elements for technicians, customers, countries, and products. Don't include the <select> element itself in the partial view for a view component. That way, the <select> element can still use the asp-for attribute for model binding and data validation. Here's an example of how a <select> element with a view component for options should look:

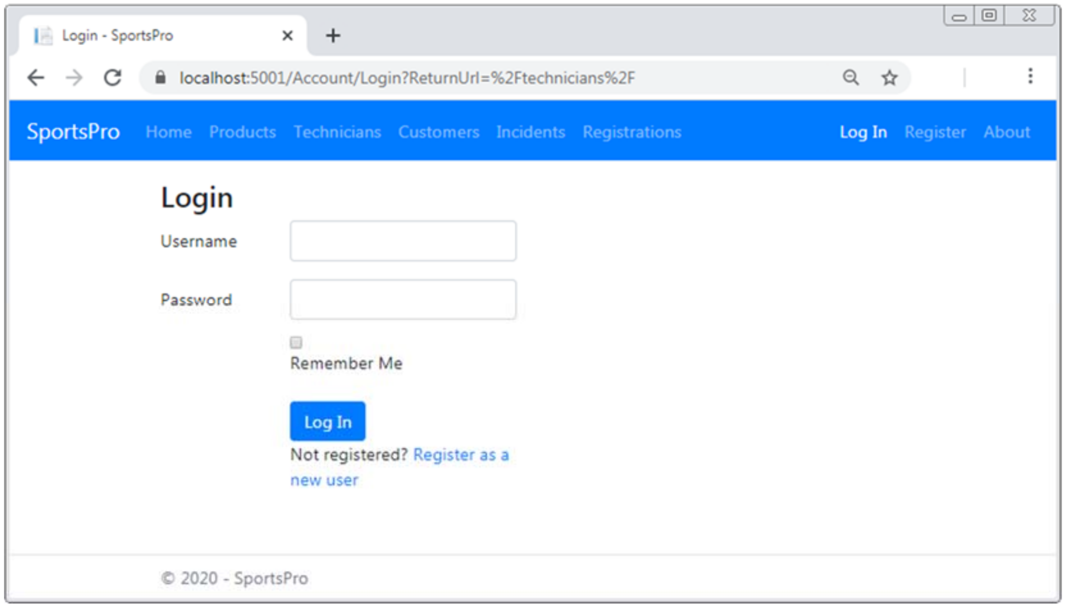
  ```
  <select asp-for="Incident.ProductID" class="form-control">
      <vc:product-drop-down-options
          value="@Model.Incident.ProductID"
          default-value=""
          default-text="Select a product..." />
  </select>
  ```

  These view components should handle getting the data for the drop downs. The controllers should no longer use the ViewBag property or view model objects to pass that data to the views. If a controller previously used a unit of work object but can now use a single repository object, it should be updated to use that repository object.
- If you have a unit test project, run the tests when you make changes to the controllers to make sure those changes don't cause your tests to fail. If you encounter errors, fix the website or unit tests as needed.

Assignment 16-1: Authenticate and authorize users

For this assignment, the admin functions will only be available to users who have logged in as an admin user.

The Login page



Login

Username

Password

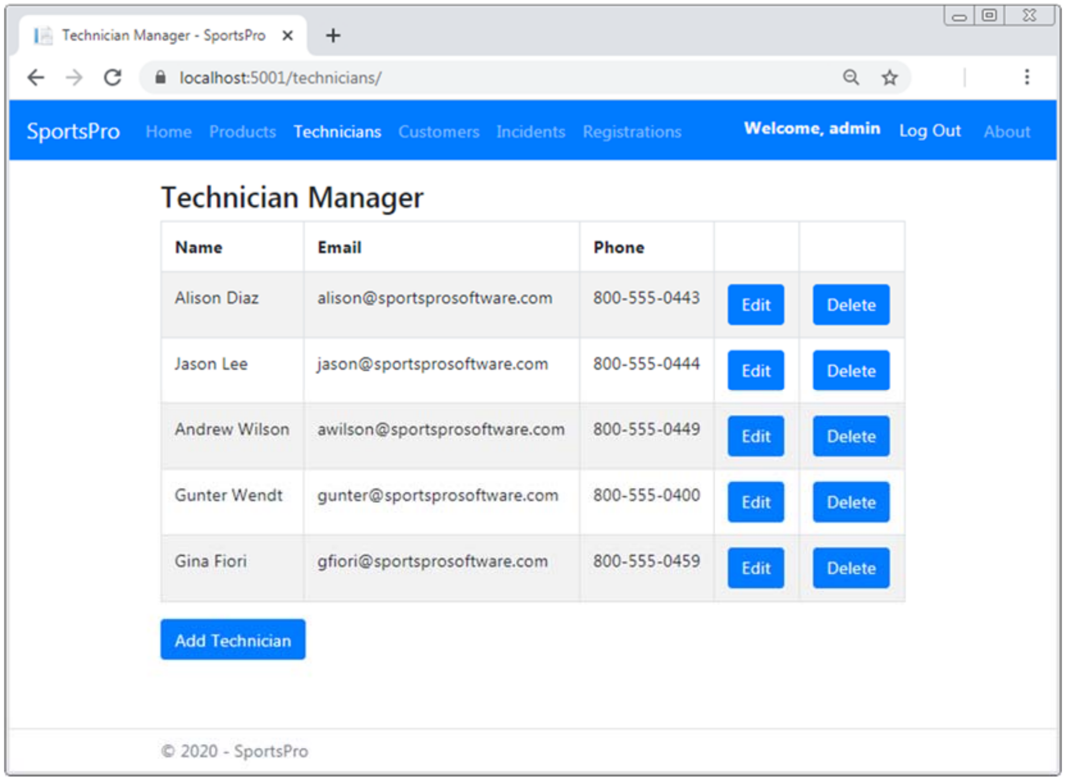
Remember Me

Log In

Not registered? Register as a new user

© 2020 - SportsPro

The Technician page



Technician Manager

Name	Email	Phone		
Alison Diaz	alison@sportsprosoftware.com	800-555-0443	Edit	Delete
Jason Lee	jason@sportsprosoftware.com	800-555-0444	Edit	Delete
Andrew Wilson	awilson@sportsprosoftware.com	800-555-0449	Edit	Delete
Gunter Wendt	gunter@sportsprosoftware.com	800-555-0400	Edit	Delete
Gina Fiori	gfiori@sportsprosoftware.com	800-555-0459	Edit	Delete

Add Technician

© 2020 - SportsPro

Specifications

- Anonymous users can only view the Home page, the About page, the Register page, and the Login page. If an anonymous user attempts to access any other page, the app displays the Login page.
- If the user enters a valid username and password into the Login page, the website redirects to the page the user originally clicked, if the user is authorized to view that page. Otherwise, it displays the Access Denied page.
- The navbar should display Log In and Register links to anonymous users, and a Log Out link with a welcome message to authenticated users.
- Only a user in the Admin role should be able to access the admin pages. For example, only a user that's logged in and is in the Admin role should be able to access the Product Manager page.
- Any authenticated user should be able to access the Update Incident page that's designed for technicians.
- On startup, the app should add a user with a username of "admin" and a password of "P@ssw0rd" to the Admin role.