

ASP.NET Web Application Development

Due Date: February 25th, 2021

Problem Synopsis:

Having graduated from College, you have recently been offered and accepted a position at a lucrative start-up company. The company consists of a number of experienced engineers and developers, who believe that hiring a new graduate, such as you, will result in an abundance of new and innovative ideas to the organization.

The company is situated in the Sporting sector, and the IT Director has identified the need to develop an application 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

To kick-off phase I of the project, your manager has the initial task of assembling a team capable of constructing the first phase of the project. As such, the IT Director, Management team and Architects all agree, that utilizing **ASP.NET (Visual Studio)**, **C#**, **IIS (IIS Express)** and **SQL Server (SQL Express)** are a mandatory requirement. The architects envision the solution should be delivered in **two separate releases**, specifically, phase I (assignment 1) and phase II (assignment 2 – to follow later in the course).

Your manager recalls during the interview, that you had highlighted your familiarity of ASP.NET, C# and web application development. Giving your experience of these technologies, you immediately appeal as an ideal candidate to be added to the project team. Additionally, by selecting you as part of the project initiative, your manager will conveniently be presented with the opportunity to evaluate your performance, gauging how you may possibly fit into future projects within the organization.

The phase I project deliverables have been scoped-out for by the Business Analysts. Wireframes have been constructed and serve to clarify the project deliverables that the solution must contain at a **minimum**.

Project Brief

The College Sporting 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.

The Database

The database for the application stores the data that's needed to track technical support incidents. You may build the database up incrementally, but ideally for the completed web application, the database should include tables for storing data about the company's products, technicians, customers, incidents, and registrations (you do not need all these tables for assignment 1, the recommendation is you develop only the elements of the schema you need for assignment 1, and defer the remaining tables/relationships for the final assignment, assignment 2).

The Table Entity Relationships

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 (join 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.

The Assignment

The description of the assignment includes one or more screen captures (wireframes) that conveys how the pages should conceptually appear when they're completed as well as specifications for how the assignment should be coded. **Please note however, the design of each page, and the overall application however, is ultimately left up to each team. A level of creativity and originality is much appreciated.**

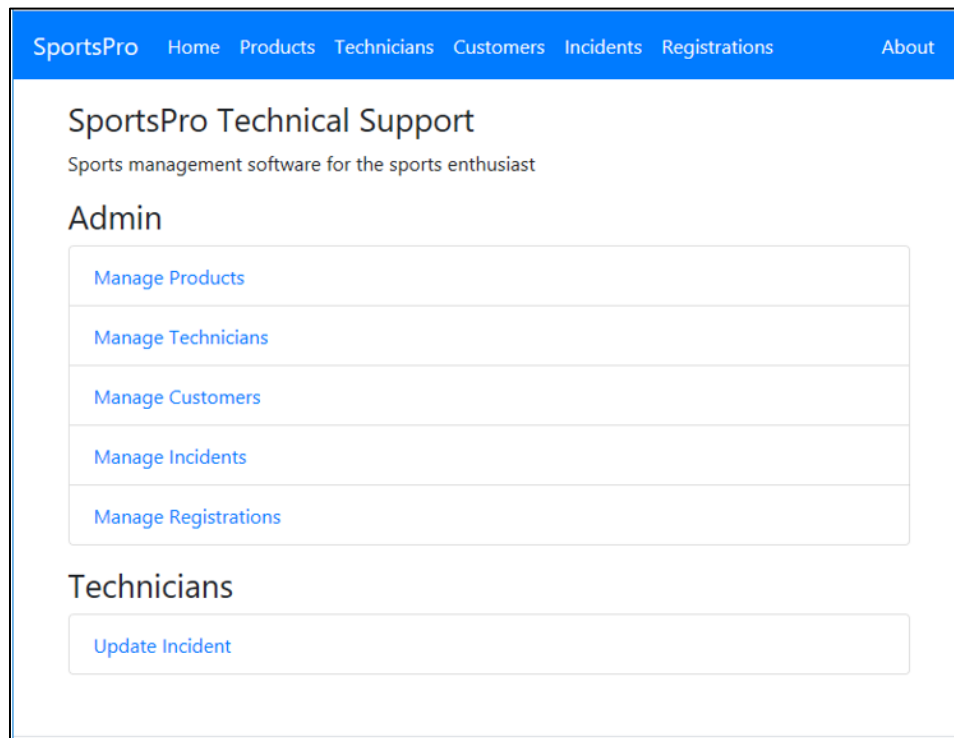
This information is detailed enough for you to complete the assignment. However, you'll need to use your best judgment on how to code many of the details. To do that, write the code in the way that you think is best.

For some requirements, the assignment's specifications will require your team to perform a certain amount of **research**. For example, some requirements will require the study of technologies/enhancements related to the course material, but may perhaps have yet to be formally covered in the course lectures.

Home Page

The College Sports Application requires a Home Page. The Home Page includes a navigation bar and some menu options for alter assignments

The Business Analysts have provided a wireframe mockup of a potential home login page below:



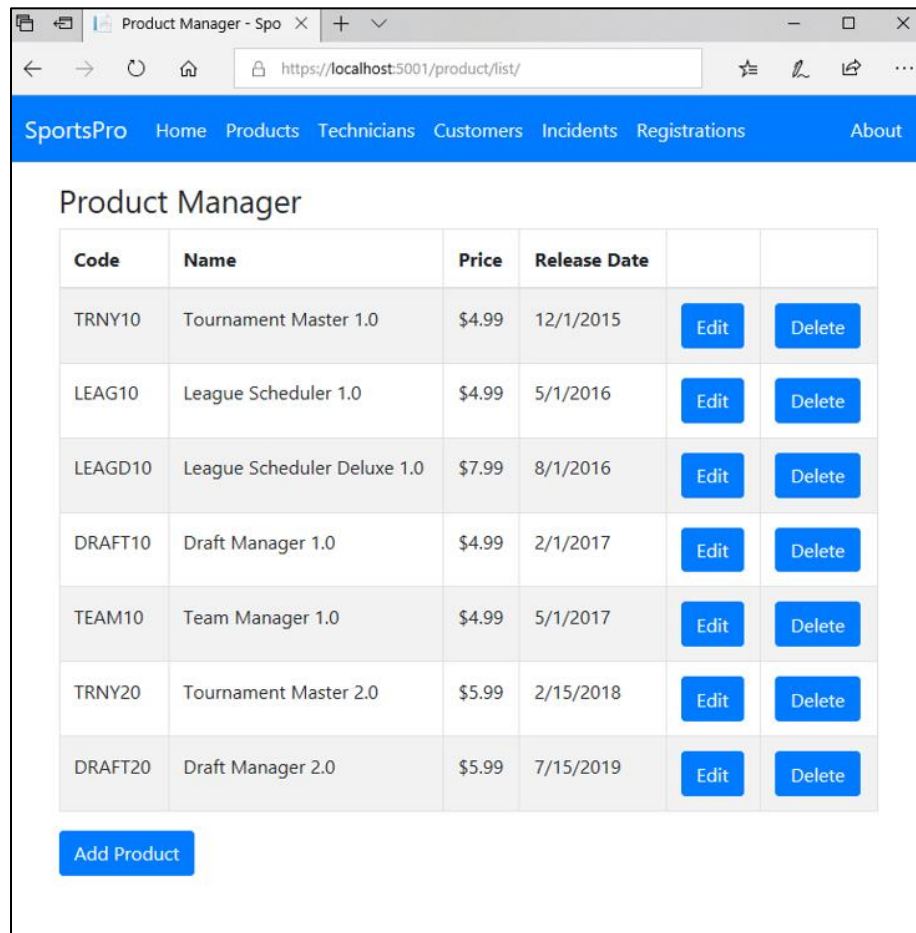
Home Page - Specifications

- Create an application that based on the MVC (Model-View-Design) pattern.
- 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 footer
- The recommendation is to use bootstrap for styling, but the design is left open to each individual group.
- Mark all navigation bar items as active.

Manage Products

The College Sports Pro application requires a Products page that allows admin users (and only admin users) to view, edit and delete existing products.

The Business Analysts have provided a wireframe mockup of a potential Products page below:



The Product Manager Page - Specifications

- When the user clicks on the Manage Products link on the Home Page, or the Products link in the navbar, the application should display the Product Manager page.
- When the user clicks the Add Product button, the application 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 application displays the Add/Edit Product page with the current data for the product.
- When the user clicks the Delete button for a product, the application should display a Delete Product page that confirms the deletion.

The Add/Edit Product Page

The Business Analysts have provided a wireframe mockup of a potential Add/Edit page below:

The wireframe mockup displays two browser windows side-by-side, illustrating the 'Add Product' and 'Edit Product' pages for the 'SportsPro' application.

Left Window (Add Product):

- URL: `https://localhost:5001/product/add/`
- Navigation: SportsPro, Home, Products, Technicians, Customers, Incidents, Registrations, About
- Form Fields:
 - Code:
 - Name:
 - Yearly Price:
 - Release Date:
- Buttons: Save, Cancel

Right Window (Edit Product):

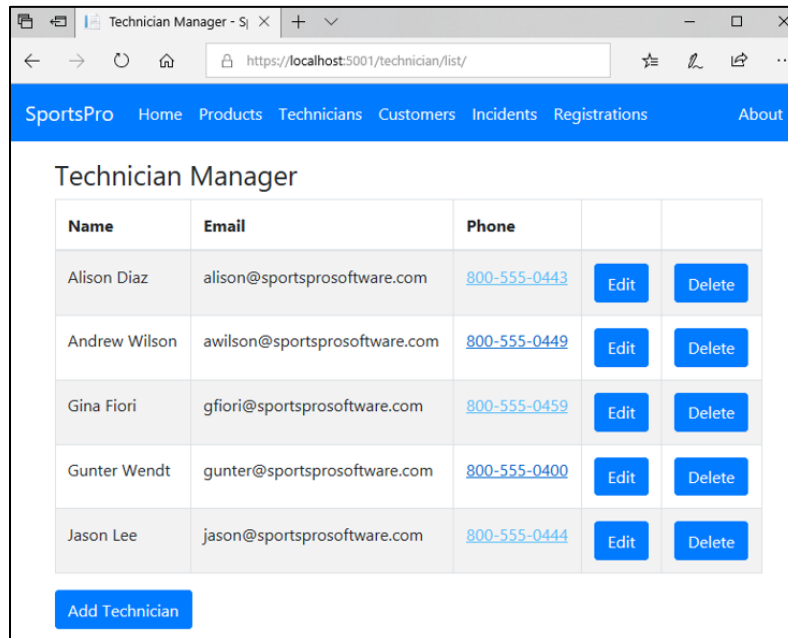
- URL: `https://localhost:5001/product/edit/6/`
- Navigation: SportsPro, Home, Products, Technicians, Customers, Incidents, Registrations, About
- Form Fields:
 - Code:
 - Name:
 - Yearly Price:
 - Release Date:
- Buttons: Save, Cancel

The Add/Edit Page - Specifications

- The application 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.
- Since they are so similar, use the same Razor view file to add and edit a product.

Manage Technicians

The Business Analysts have provided a wireframe mockup of a potential Manage Technicians page below:



The Manage Technician Page - Specifications

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

The Add/Edit Technician

The Business Analysts have provided a wireframe mockup of a potential Add/Edit Technicians page below:

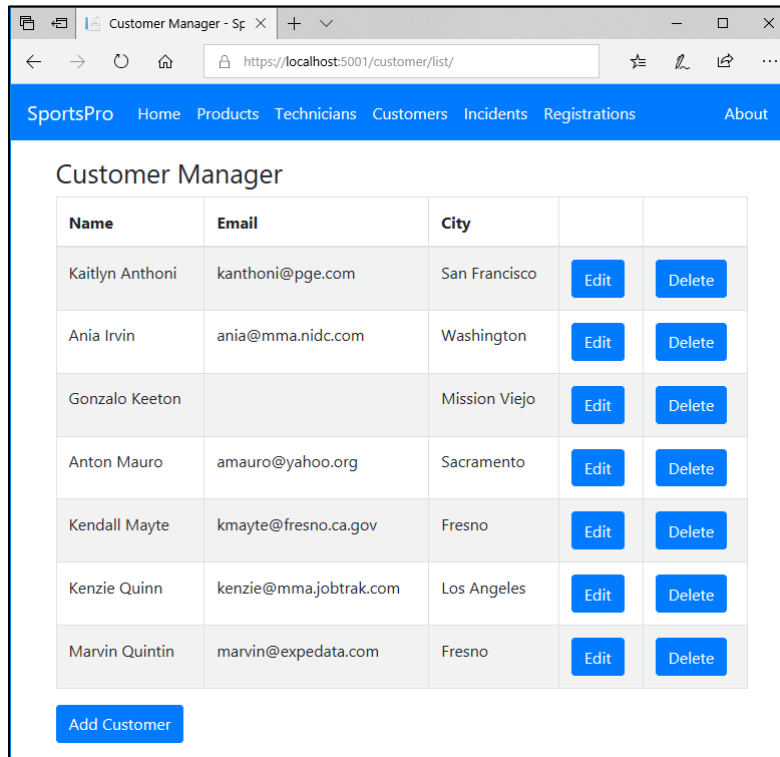
The wireframe mockup displays two browser windows. The background window, titled "Add Technician - Sports", shows a form with input fields for Name, Email, and Phone, and "Save" and "Cancel" buttons. The foreground window, titled "Edit Technician - Sports", shows a similar form with pre-filled data: Name "Alison Diaz", Email "alison@sportsprosoftware.com", and Phone "800-555-0443". Both windows feature a blue navigation bar with links: SportsPro, Home, Products, Technicians, Customers, Incidents, Registrations, and About. The foreground window's address bar shows the URL "https://localhost:5001/technician/edit/11/".

The Add/Edit Technician Page - Specifications

- The application 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 application 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.

Manage Customers

The Business Analysts have provided a wireframe mockup of a potential Manage Customers page below:

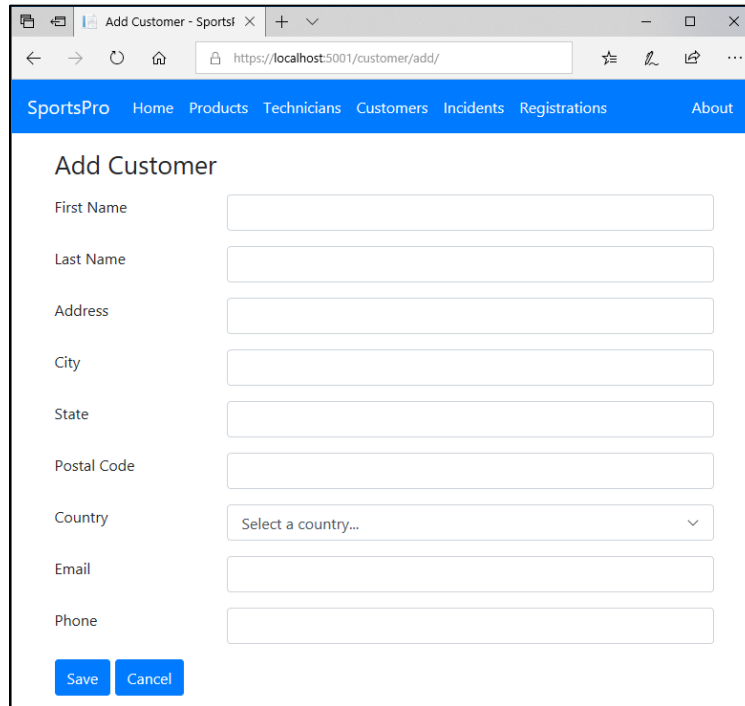


The Manage Customers Page - Specifications

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

The Add/Edit Customer Page

The Business Analysts have also provided a wireframe mockup of a potential Add/Edit Customer page below:



The wireframe shows a web browser window with the title 'Add Customer - SportsPro'. The address bar shows 'https://localhost:5001/customer/add/'. The navigation bar is blue with links: SportsPro, Home, Products, Technicians, Customers, Incidents, Registrations, and About. The main content area is titled 'Add Customer' and contains the following form fields:

- First Name
- Last Name
- Address
- City
- State
- Postal Code
- Country (drop-down menu with 'Select a country...' text)
- Email
- Phone

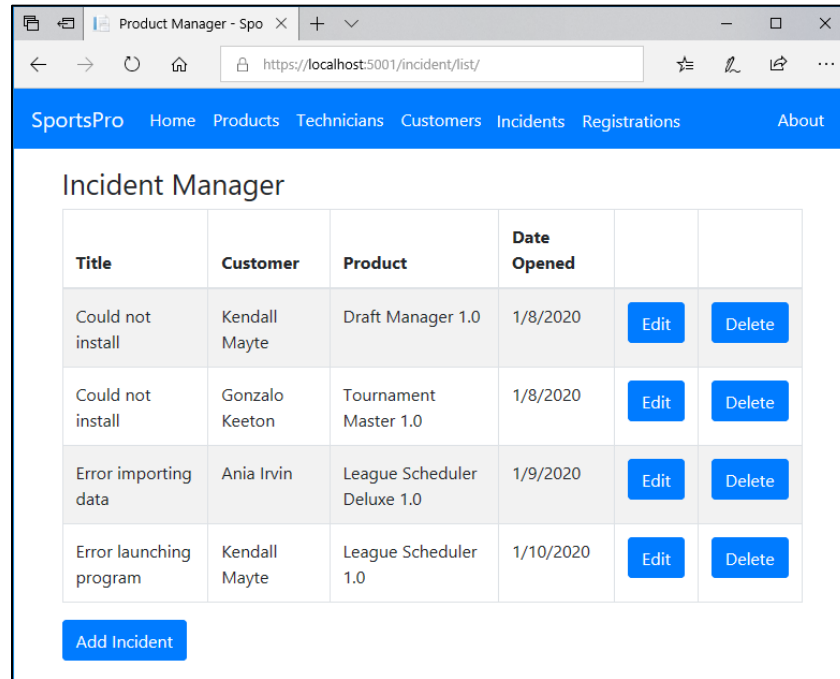
At the bottom of the form are two buttons: 'Save' and 'Cancel'.

The Add/Edit Customers Page - Specifications

- The application 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 (add some countries to your database).
- When the application displays the Edit Customer page, make sure to select the correct country for the specified customer.

The Manage Incident Page

The Business Analysts have also provided a wireframe mockup of a potential Manage Incidents page below:



The Manage Incidents Page - Specifications

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

The Add/Edit Incidents Page

The Business Analysts have also provided a wireframe mockup of a potential Add/Edit Incidents page below:

The Add/Edit Incidents Page - Specifications

- The application 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.

Final Word on Implementation

Lastly, the Business Analysts have decided to leave all **final** design (aesthetics, navigation etc...) decisions up to the developers to ultimately decide and convey. The wireframe provided are only provided to help communicate **the minimum requirements and NOT to establish the rule**, so creativity and interpretation is welcome.

Any missing or additional forms or pages are also welcome. Again, a requirement's document only conveys a perception of what is required, but each developer is free to interpret differently, so long as the underlying requirement is still met. Originality is always welcome and often marks are associated accordingly.

Assignment Submission Guidelines:

1. You must email your assignment to your manager (Professor)
2. **Steps to submit project**
 - a. Close your project and exit Visual Studio
 - b. Navigate to your project folder, and make a copy (recommended)
 - c. Remove and delete the following folders in your copied project directory:
 - i. .vs
 - ii. packages
 - iii. bin
 - iv. obj
 - d. Compressed (**.zip**) the copied project folder and name accordingly
 - e. Send email attachment from your **college** email account to the instructor.
3. All members in the project team must be cc'd on the assignment submission. Failure to do so will result in a mark of zero for those members not cc'd on the assignment submission email.
4. Within the body of the email, clarify course code, team name, team members and student numbers. Title the email accordingly - Assignment #.
Example:
Team Name: The Hackers
Team Members: John Smith - 1234567
Sally Jones - 7654321
Jane Wilson - 2342342
4. The uploaded compressed file must be in **.rar** or **.zip** format.
5. The .zip/.rar file naming convention as follows:
6. Your code should be modular and should show no signs of dry (don't repeat yourself) code.
7. You are required to devise and use a form of persistent data storage.
8. The technologies used for this assignment are ASP.NET, Visual Studio, IIS and SQL Server Express.
9. Be cautious **DO NOT** share your application with others. Complete failures will be assigned if code is shared. All assignments will be reviewed and analyzed strictly within these regards.
10. Late assignments are assigned a penalty of 25% per day.