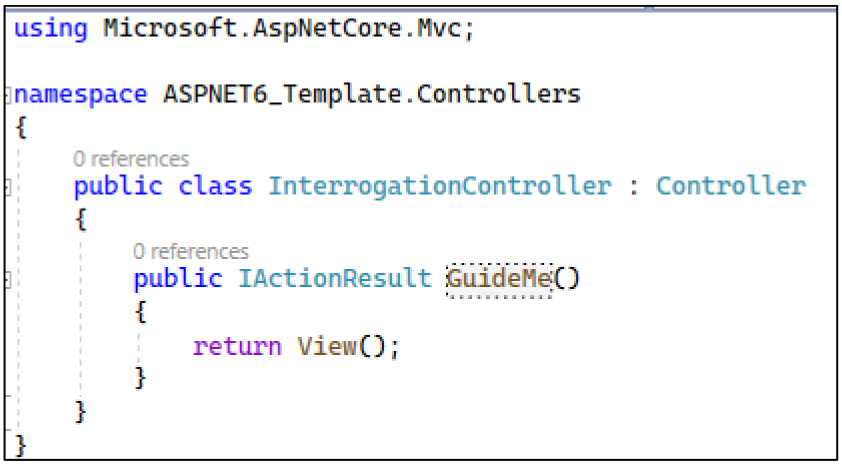
Scenario:

The requirement is to create an online onterrogation panel application using the Model-View-Controller (MVC) architecture that allows interrogators to create problem with four choices and solution for that through a form. Use partial views and model binding concepts to achieve it.

Exercise Steps:

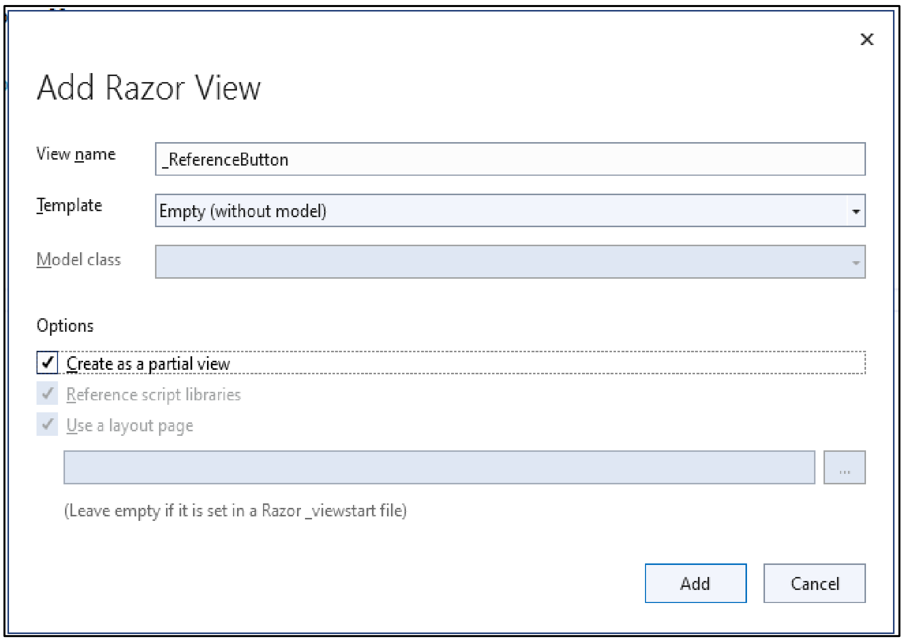
1. Create a new ASP.NET Core MVC project in Visual Studio.
2. Add a new controller called **"InterrogationController"** and create an **"GuideMe"** action method that returns a view.



```
using Microsoft.AspNetCore.Mvc;

namespace ASPNET6_Template.Controllers
{
    0 references
    public class InterrogationController : Controller
    {
        0 references
        public IActionResult GuideMe()
        {
            return View();
        }
    }
}
```

3. In **"Shared"** folder in the **"Views"** folder and create a new partial view called **"_ReferenceButton.cshtml"**.(In default MVC template you will have Shared and Views folder. Not necessary to create folder if already exist). Please refer the below image for creating partial views


Add Razor View

View name: _ReferenceButton

Template: Empty (without model)

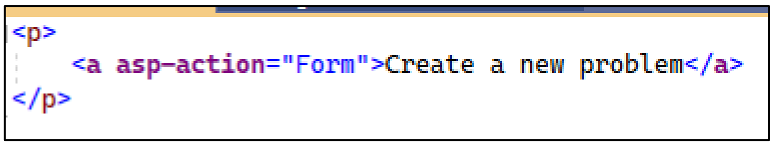
Model class:

Options

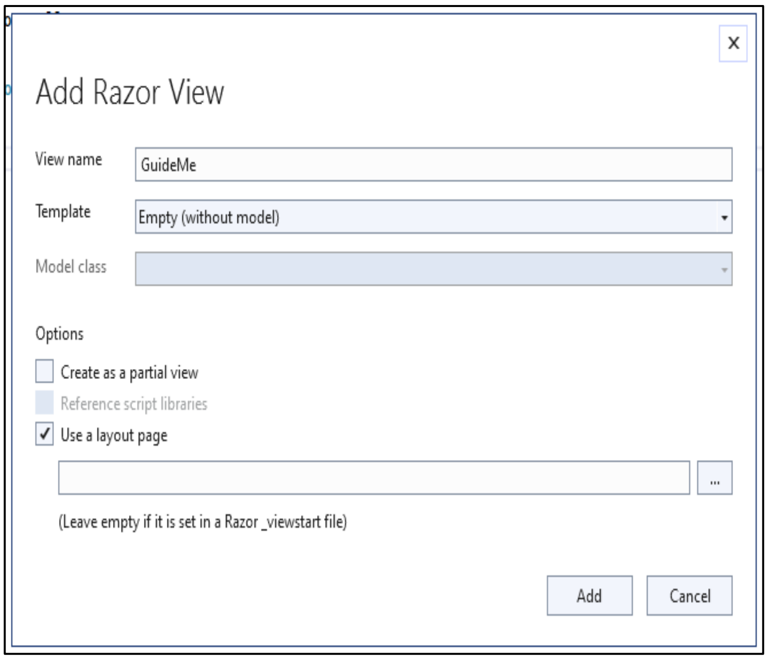
- [✓] Create as a partial view
- [✓] Reference script libraries
- [✓] Use a layout page

(Leave empty if it is set in a Razor _viewstart file)

4. In the **"_ReferenceButton.cshtml"** partial view, add HTML code to display a hyperlink. You can use an HTML anchor tag and display it as **Create Problem**. When the interrogator click the link, it will take to another page.


```
<p>
    <a asp-action="Form">Create a new problem</a>
</p>
```

5. In the **InterrogationController**, hover the cursor over the **GuideMe** action method and click the right button and choose Add View and name it as **GuideMe**.cshtml. Please refer the below image for creating views


Add Razor View

View name: GuideMe

Template: Empty (without model)

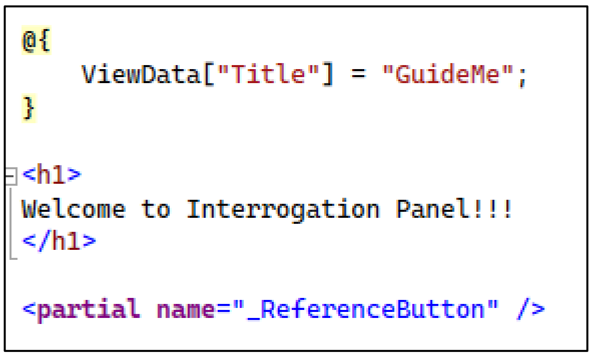
Model class:

Options

- [ ] Create as a partial view
- [ ] Reference script libraries
- [x] Use a layout page

(Leave empty if it is set in a Razor _viewstart file)

6. In the **"GuideMe.cshtml"** view of the **"InterrogationController"**, include the **"_ReferenceButton.cshtml"** partial view using the following code:

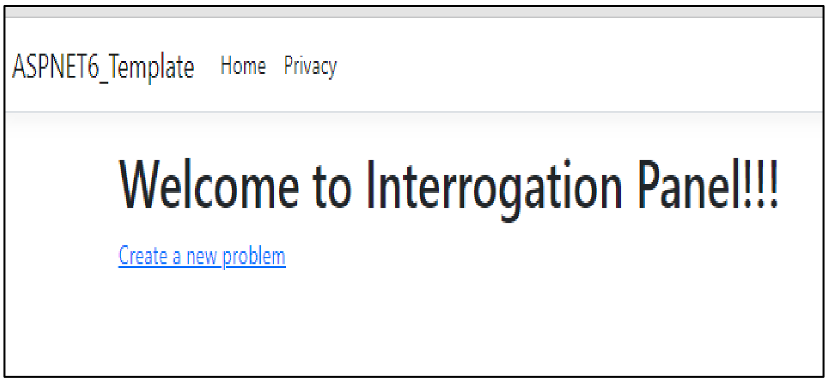

```
@{
    ViewData["Title"] = "GuideMe";
}

<h1>
Welcome to Interrogation Panel!!!
</h1>

<partial name="_ReferenceButton" />
```

7. Run the application and navigate to the **"GuideMe"** page. You should see the hyperlink named Create Question displayed from **"_ReferenceButton.cshtml"** partial view.

8. Let’s see the outcome of the above instructions,

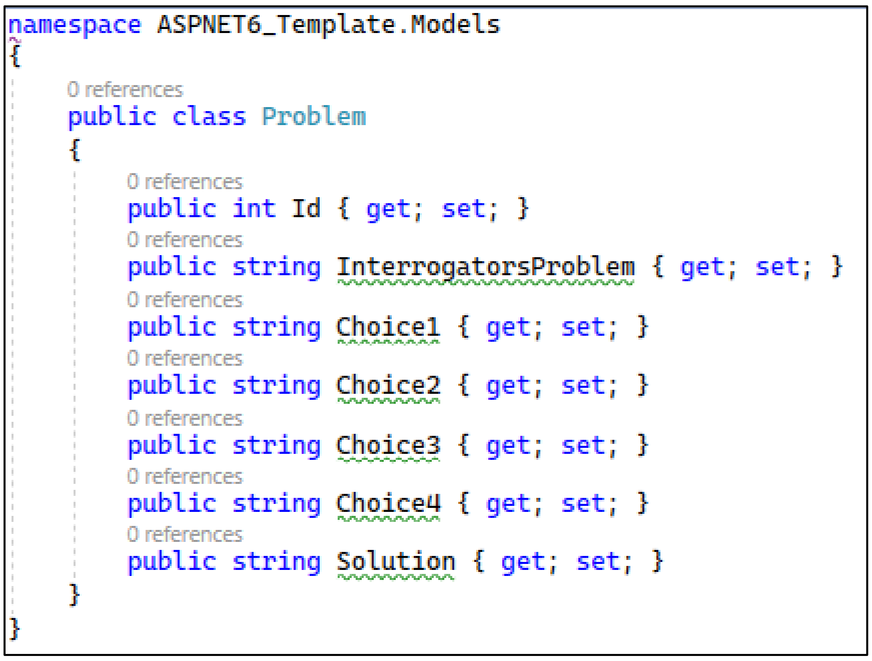


Welcome to Interrogation Panel!!!

Create a new problem

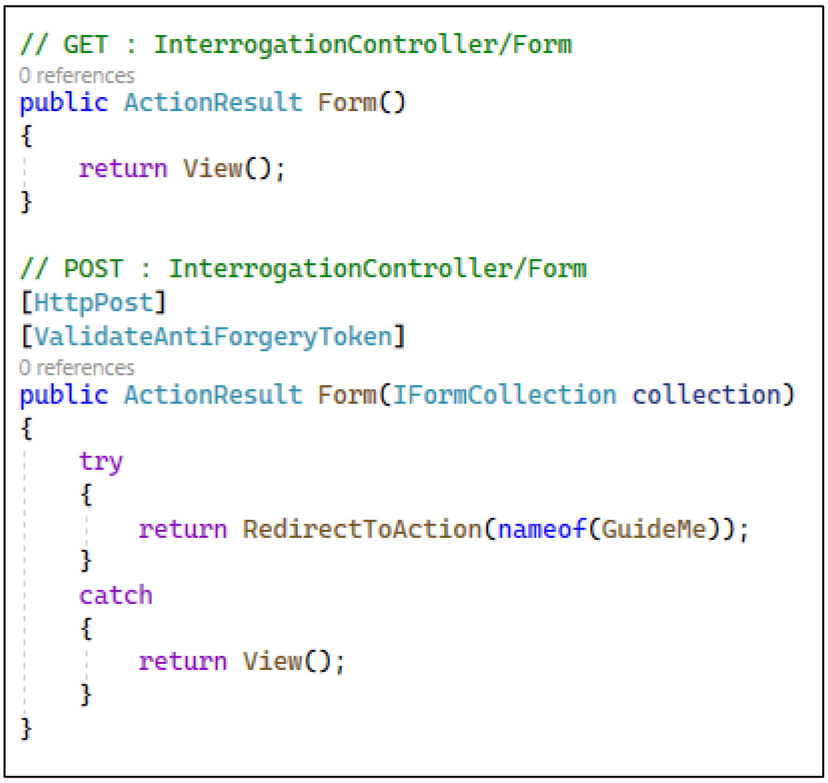
9. We have successfully created a partial view named as **"_ReferenceButton.cshtml"** and included this partial view in **"GuideMe.cshtml"** page. Now we should create another view for the **"_ReferenceButton.cshtml"** Where we need to create a form, which displays Id, Problem, Choices, and Solution.

10. In folder named **Models** and inside that folder create a class named **Problem.cs**. In this class, create the properties mentioned below,



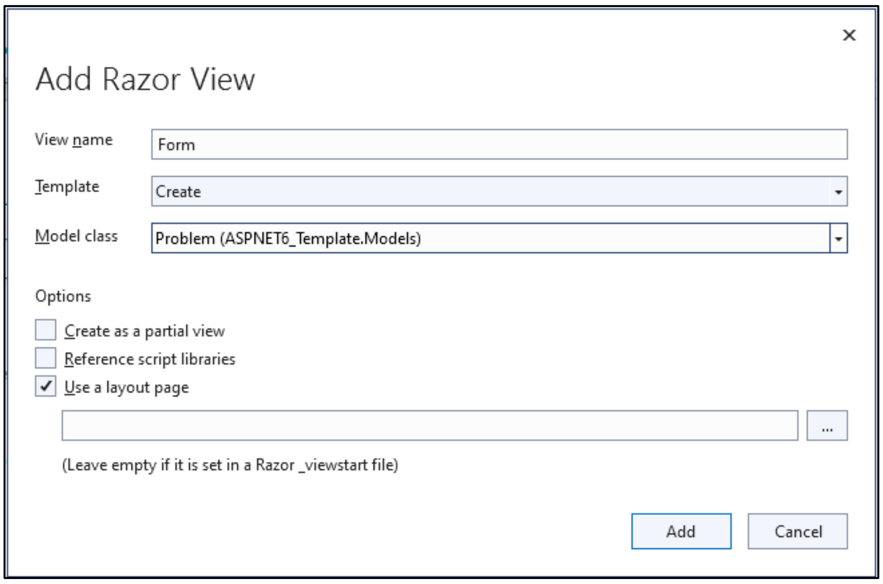
```
namespace ASPNET6_Template.Models
{
    0 references
    public class Problem
    {
        0 references
        public int Id { get; set; }
        0 references
        public string InterrogatorsProblem { get; set; }
        0 references
        public string Choice1 { get; set; }
        0 references
        public string Choice2 { get; set; }
        0 references
        public string Choice3 { get; set; }
        0 references
        public string Choice4 { get; set; }
        0 references
        public string Solution { get; set; }
    }
}
```

11. In **InterrogationController.cs**, create an action named **Form** with **HttpGet**, which just returns a view, and another action named **Form** with **HttpPost**, which redirect to another action named **GuideMe** with View. The post action must have an input parameter **IFormCollection**.


```
// GET : InterrogationController/Form
0 references
public ActionResult Form()
{
    return View();
}

// POST : InterrogationController/Form
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public ActionResult Form(IFormCollection collection)
{
    try
    {
        return RedirectToAction(nameof(GuideMe));
    }
    catch
    {
        return View();
    }
}
```

12. Now let us create a view in **Views/Interrogation** for the **Form** action method by binding the **Models** in that view.


Add Razor View

View name: Form

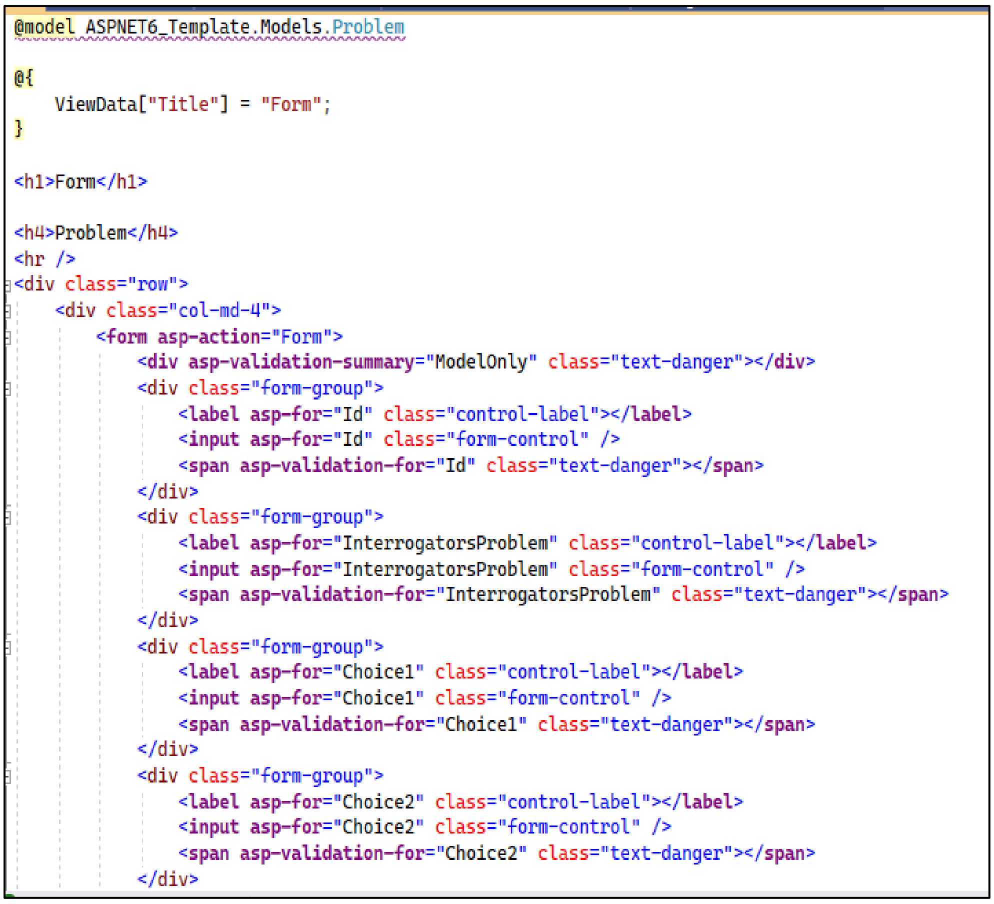
Template: Create

Model class: Problem (ASPNET6_Template.Models)

Options

- [ ] Create as a partial view
- [ ] Reference script libraries
- [x] Use a layout page

(Leave empty if it is set in a Razor _viewstart file)

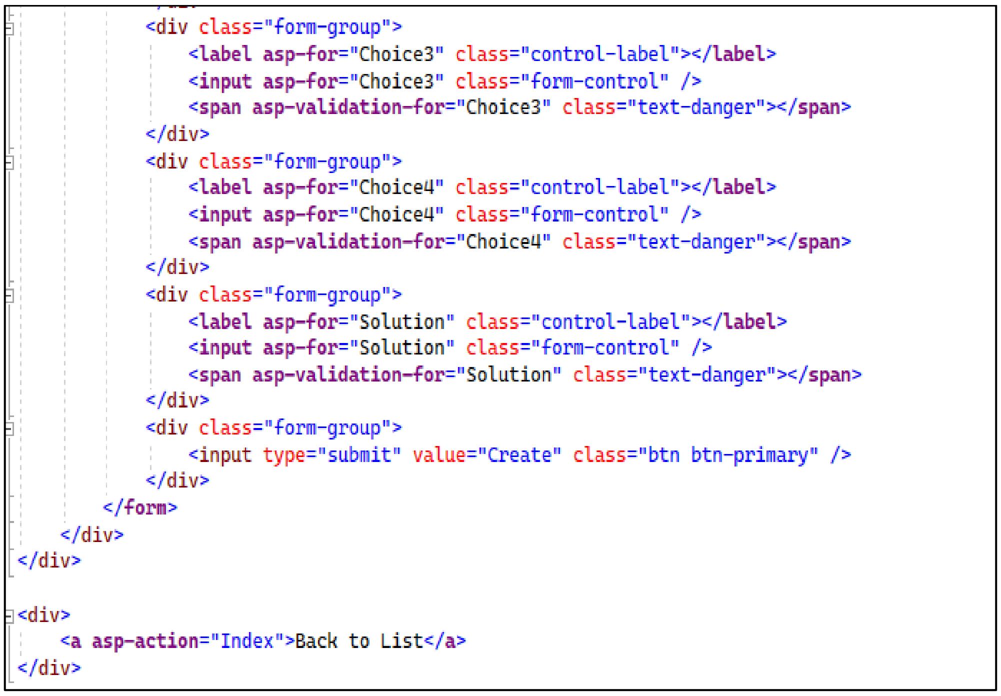


```
@model ASPNET6_Template.Models.Problem

@{
    ViewData["Title"] = "Form";
}

<h1>Form</h1>

<h4>Problem</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Form">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Id" class="control-label"></label>
                <input asp-for="Id" class="form-control" />
                <span asp-validation-for="Id" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="InterrogatorsProblem" class="control-label"></label>
                <input asp-for="InterrogatorsProblem" class="form-control" />
                <span asp-validation-for="InterrogatorsProblem" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Choice1" class="control-label"></label>
                <input asp-for="Choice1" class="form-control" />
                <span asp-validation-for="Choice1" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Choice2" class="control-label"></label>
                <input asp-for="Choice2" class="form-control" />
                <span asp-validation-for="Choice2" class="text-danger"></span>
            </div>
```

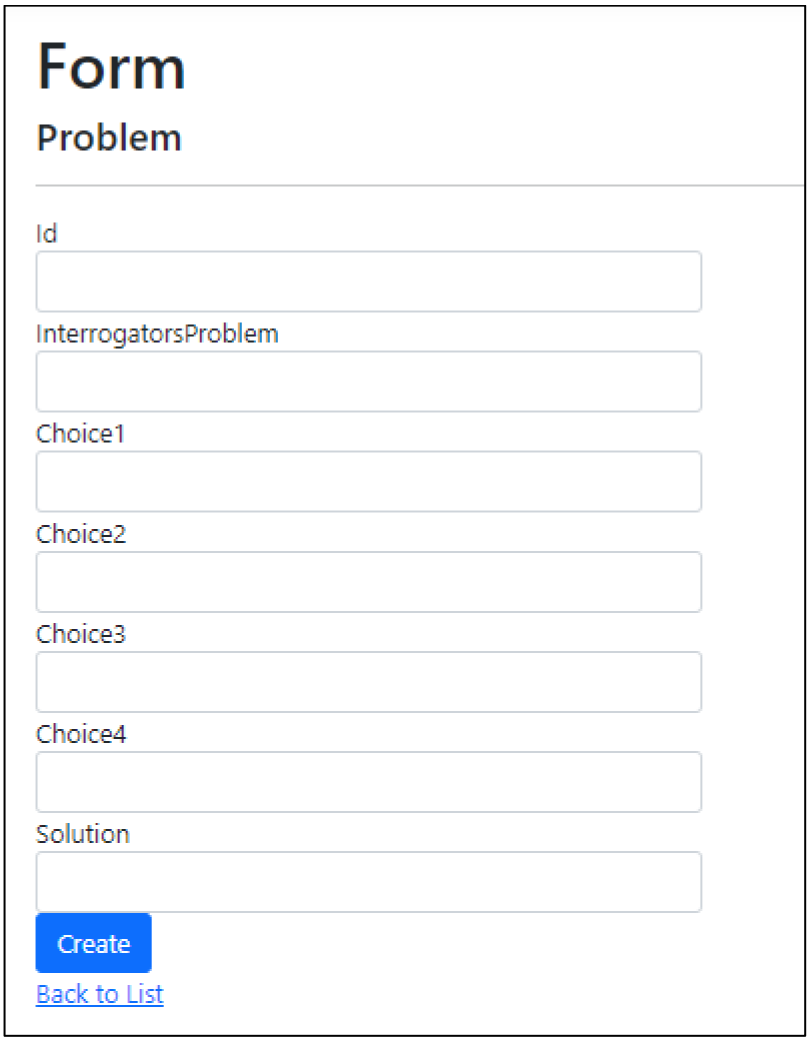


```
            <div class="form-group">
                <label asp-for="Choice3" class="control-label"></label>
                <input asp-for="Choice3" class="form-control" />
                <span asp-validation-for="Choice3" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Choice4" class="control-label"></label>
                <input asp-for="Choice4" class="form-control" />
                <span asp-validation-for="Choice4" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Solution" class="control-label"></label>
                <input asp-for="Solution" class="form-control" />
                <span asp-validation-for="Solution" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Create" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>
```

13. Run the application and navigate to the **"GuideMe"** page. You should see the hyperlink named Create Question displayed from

"_ReferenceButton.cshtml" partial view. When you click that hyperlink it would take you to the **Create.cshtml**. The page would look like this,



Form

Problem

Id
InterrogatorsProblem
Choice1
Choice2
Choice3
Choice4
Solution

14. When the form is filled and when you hit the create button it button it would redirect you to the action method named GuideMe.

To summarize,

This exercise demonstrates how partial views can be used to reuse code across multiple views in an ASP.NET Core application. By creating a single partial view named **"_LinkButton.cshtml"** we can use this partial view in any new or existing views. We also learnt about model binding by getting the values from the user and setting those values to the properties we declared.