ASP.NET MVC Exercises

1) MVC Project

Create a new ASP.NET Web Application and give it the name FlightMVC. Choose a suitable location for the project (Desktop). On the New project dialog choose (Core) and set the authentication as Individual User Accounts.

Build and run.

Review the project structure within the Solution Explorer.

Review the project.json and startup.cs files.

Review the _ViewStart.cshtml file (Views folder) and _Layout.cshtml file.

2) Display PassengerDetails

Add a public class called PassengerDetails to the Models folder. To this class add Name (string) and Weight (int) properties.

Right click on the Controllers folder and add a Controller. Give this the name **Passengers**Controller (name is convention based). Within the 'Add Scaffold' dialog select to add the 'MVC Controller with read/write actions'.

To the controller add a static field called '_details' to hold a list of PassengerDetails. Within a static constructor populate the list with a number of PassengerDetails.

Implement the Index method to return the list of PassengerDetails.

```
public IActionResult Index()
{
    return View(_details);
}
```

3) Add Link to Passengers

Edit _Layout.cshtml to add a link to the Passengers view of the form:

<a asp-area="" asp-controller="Passengers" asp-action="Index">Passengers

Build and run. Note the error due to absence of view.

4) Add View for Passengers

Add a folder called Passengers (same name as Controller) to the Views folder. Right click on this folder and select 'Add View'. Within the 'Add View' dialog select View Name 'Index'; template 'List' and model class as 'PassengerDetails'. Leave 'Use a layout page' ticked.

Build and test.

5) Add Details View

Within the PassengersController implement the Details method to return a single passenger. Change the type of the parameter to string. Use this value to select the PassengersDetails from the list.

Add a view with model to display the PassengerDetails (Details view).

Edit the Index view to change the link to:

```
<a asp-action="Details" asp-route-id="@item.Name">Details</a>
```

Edit the Details view to change the link to:

```
<a asp-action="Edit" asp-route-id="@Model.Name">Edit</a>
```

Build and test.

6) Add Edit View (Get)

Within the PassengersController implement the Edit method to return a single passenger. Change the type of the parameter to string. Use this value to select the PassengersDetails from the list.

Add a view with model to display the PassengerDetails (Edit view).

Edit the Index view to change the link to:

```
<a asp-action="Edit" asp-route-id="@item.Name">Edit</a>
```

Build and test.

7) Implement Edit Method (Post)

Within the PassengersController implement the Edit method (post) to change the value of a PassengerDetails within the list. Change the type of the parameters to string and PassengerDetails. Check the ModelState is valid before using the data passed in to modify the data in the list.

Return the PassengerDetails if the ModelState is not valid.

Build and test.

8) Add Create View (Get)

Add a view with model to display the PassengerDetails (Create view).

Build and test.

9) Implement Create Method (Post)

Within the PassengersController implement the Create method (post) to add a PassengerDetails to the list. Change the type of the parameter to PassengerDetails. Check the ModelState is valid before using entered data to add to the list.

Return the PassengerDetails if the ModelState is not valid.

Build and test.

10) Add Delete View (Get)

Within the PassengersController implement the Delete method to return a single passenger. Change the type of the parameter to string. Use this value to select the PassengersDetails from the list.

Add a view with model to display the PassengerDetails (Delete view).

Edit the Index view to change the link to:

```
<a asp-action="Delete" asp-route-id="@item.Name">Delete</a>
```

Build and test.

11) Implement Delete View (Post)

Within the PassengersController implement the Delete method (post) to delete the PassengerDetails. Change the type of the parameter to string. Use this value to find the PassengersDetails within the list.

Build and test.

12) ViewData

Within the Index action add a statement to set a 'Heading' for the index view, using ViewData. Edit Index view to display the 'Heading'.

Build and test.

13) ViewBag

Within the Index action add a statement to set a 'Heading' for the index view, using ViewBag. Edit Index view to display the 'Heading'.

Build and test.

14) Validation (DataAnnotations)

Run the project and when editing try typing letters into the weight field. What happens?

Try typing a large number into the weight field.

To the PassengerDetails class add a Range DataAnnotation to the Weight property for a range [0-100].

Build and test.

